Cyana: Security and Control

A technical white paper for administrators, security reviewers and procurement stakeholders evaluating Cyana, SimpleHelp's AI assistant for remote support and remote monitoring and management.

Version: 1.0. Applies to SimpleHelp 6.0 and later.

---

1. Executive Summary

Cyana is SimpleHelp's AI assistant for remote support and RMM. It runs inside the SimpleHelp platform organisations already use, and gives technicians a natural language interface to the fleet, to individual live sessions, and to the SimpleHelp Toolbox.

With Cyana, a technician can ask one question across a group of machines and get an answer that used to cost minutes or even hours of dashboard filtering, manual searches and on-device investigation. They can investigate a failing service inside a live session without switching windows, and draft or adjust Toolbox scripts by describing what they want. The aim is not to replace the technician, but to ensure that more of the technician's day is spent deciding and acting, and less of it gathering the information needed to decide.

The same capability that makes AI useful in RMM is what makes it dangerous if it is not controlled. As such, Cyana is built on one principle: controls do not depend on the model behaving well. Every control is enforced in server code, and the layers deliberately do different jobs. Administrator group permissions are a primary security control: they define what Cyana can ever be asked to do for a team, and a capability that is not granted is never offered to the model at all. Technician approval is the attention control: every chat starts in a mode which requires maximum approval from the technician, and allows the model minimum autonomous capability. Every meaningful action waits for an explicit decision, and any trust the technician grants is per category, per chat, and gone when the chat is cleared. An independent reviewer model with no tools of its own rates every higher risk action with a traffic light system, a red rating forces explicit confirmation even over the technician's own auto approve, and a review layer that cannot answer fails closed into requiring approval. Behind these sit outbound scanning for protected server configuration values, an isolated agent for web research, and a full audit record of every prompt, proposal, rating, approval and result in the standard SimpleHelp history.

Cyana runs against a model endpoint the organisation chooses, whether a frontier provider or a fully self hosted model on the organisation's own infrastructure, so the data path, cost and capability of the AI are decisions the organisation makes and can change with minimal cost. Chat and review can use different models from different providers.

The intended readers for this document are IT managers, MSP owners, security reviewers and procurement stakeholders. The paper describes how each control works, what evidence exists for it in the product today, and how to roll Cyana out, starting safely and expanding as trust in the specific model builds. It is written to be checked: where a claim depends on configuration, that is called out, and the limits of every control are stated plainly in section 12.

---

2. Why AI Powered RMM Needs Control

Remote support and management has always been about visibility and action. Technicians watch the fleet, notice the machine that needs attention, and do something about it. The work is critical to business operations, but a lot of it is slow, filtering dashboards, checking one endpoint at a time, and writing the same commands again and again.

AI changes the interface, not the technician's expertise or responsibility. Instead of hunting through screens, a technician can ask a question across a group of machines and get an answer, and can review a proposed action before anything runs, bringing to bear the full context of the entire environment and business.

The concern here is obvious. To provide this capability, the endpoint tools that SimpleHelp provides are powerful, and the machines they reach often hold customer data, production workloads or regulated information, so letting a model act broadly across that estate is not something most IT teams or MSPs are willing to accept, nor something they should be willing to accept. The question that matters from a security perspective therefore is not whether AI can help with RMM, but whether AI can help with RMM without giving up control.

Cyana is designed around that question. Rather than relying on the model behaving well, it combines four independent control layers: administrator enablement, technician group permissions, per action technician approval, and automated review by a separate agent for higher risk actions. Each layer can be tightened independently, and each is enforced in code rather than in prompts.

---

3. How Cyana Fits Into SimpleHelp

3.1 Platform Integration

Cyana is not an integration or a separate product. It is a layer built into SimpleHelp and deeply embedded within it. It runs on the same SimpleHelp server, uses the same remote access service on each machine, and honours the same technician group permission model as every other SimpleHelp feature.

That has three practical consequences:

First, Cyana can only reach machines through SimpleHelp's existing platform, and has no independent connection to endpoints. If a machine is offline Cyana cannot reach it, if a technician cannot see a machine then Cyana operating for that technician cannot see it either, and if a customer has not approved a machine Cyana is refused just as any other technician transaction would be.

Second, Cyana honours SimpleHelp's existing authorisation checks. Every tool the model is offered is filtered against the technician's permissions at the moment the tool list is built, and every tool call the model makes is again checked on the server at execution time. The permission set applied to a running conversation is the snapshot taken when the conversation started. A permission change made by an administrator takes effect at the start of the technician's next Cyana conversation. Cancelling and restarting the conversation applies the new permissions immediately.

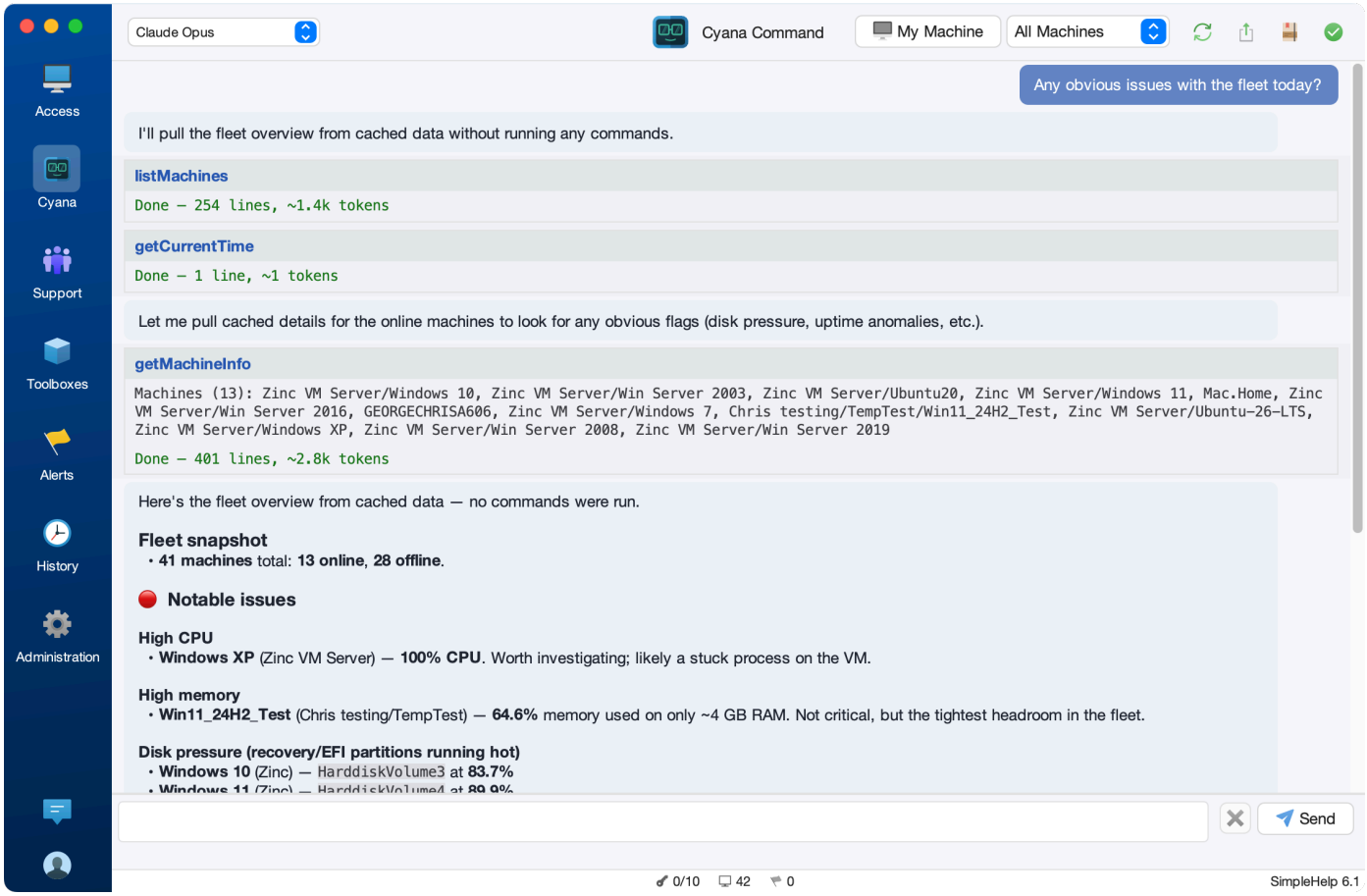
Third, Cyana's activity is captured in the same history and reporting surface as normal support and remote work activity, and is subject to the same view permissions.

3.2 What Cyana Speeds Up

The easiest way to understand the benefit of Cyana is to picture the everyday work it makes faster. With scope narrowed and approval required, a technician can:

- Ask Cyana to review endpoint health across a customer site or a small machine group before starting a support session.
- Find machines with low disk space, failed services or missing updates across the fleet, and look into the ones that matter.
- Investigate a failed service inside a live session by asking Cyana to gather details, list running processes, and propose a fix for review.
- Collect and summarise logs from a set of endpoints during an incident, and produce a structured report to hand to another team.
- Compare configuration or patch state across a customer site.
- Draft, review and adjust a Toolbox script from a natural language description, without leaving the Toolbox editor.

Each of these starts with reading rather than changing, and any action that would cause change is proposed for review. The technician sees what Cyana found, decides what to do, and has a record of it.



The Cyana Command panel showing a request for a fleet overview. Cyana proposes tool calls, executes those that are approved, and returns a structured summary. The scope selectors along the top control which machines Cyana may reach.

3.3 Where Cyana Appears

To the technician, Cyana appears as a contextual chat area in the SimpleHelp technician client. The context adjusts to the technician's current activity, and the operating mode is chosen automatically by where the technician opens the panel. Section 4 covers the three modes in detail.

---

4. Cyana Operating Modes

Cyana has three operating modes. Each mode has a different tool set, and each is enforced by a separate orchestrator on the server. A technician cannot cause a tool from one mode to run in another mode, as the orchestrator itself refuses to advertise a tool the mode was not built to include.

4.1 Cyana Command

Cyana Command sits alongside the access tab and is used for broader fleet and platform work. It is the mode a technician opens to review the state of many machines, to look for anomalies across a customer site, or to ask questions that span more than one endpoint.

Scope options. The technician picks the scope for each conversation from a combo box at the top of the panel. Options include:

- All machines the technician is permitted to see.
- A specific machine group the technician selects.
- A subset the technician chooses with a picker.
- The technician's own machine, controlled by a separate "My Machine" toggle. When enabled, Cyana can also address tools at the technician's local machine.

The technician can also enter a text based technical context that is attached to every Cyana Command prompt from that technician. This can be used to make Cyana aware of the organisation's conventions, policies and priorities. Because the context is persistent and technician controlled, two facts about it matter for a security review. It is stored server side in the technician's user preferences file rather than in the client, so it survives client restarts and follows the technician's account. It is included in every Cyana Command prompt, and is therefore recorded in every persisted AI history entry for that technician's Cyana Command activity, and included in the recent technician messages section of the automated review payload described in section 6.3. There is no dedicated administrator UI for viewing or resetting a technician's standing context today; administrators who want to audit it can do so through the AI history in the SimpleHelp history panel or by inspecting the technician's preferences file on the server. The client warns the technician if the context grows past a reasonable length.

Tool universe. In Cyana Command, Cyana can list machines, read details about them, read files, list directories, search inside files, read process and service state, run commands, write files, modify files, restart or stop services, kill processes, view and assign tags, and take screenshots. Each of these categories is separately permissioned. See section 5.2 for the exact list.

Web research. In Cyana Command, as in every other mode, the web research tool is available if the technician's group has been granted the web research permission. To help prevent injection from third party websites, web research is not executed by Cyana directly. Instead it runs in a separate isolated agent that has no other tools available to it, and cannot see the parent conversation. Section 8.2 describes this compartmentalisation.

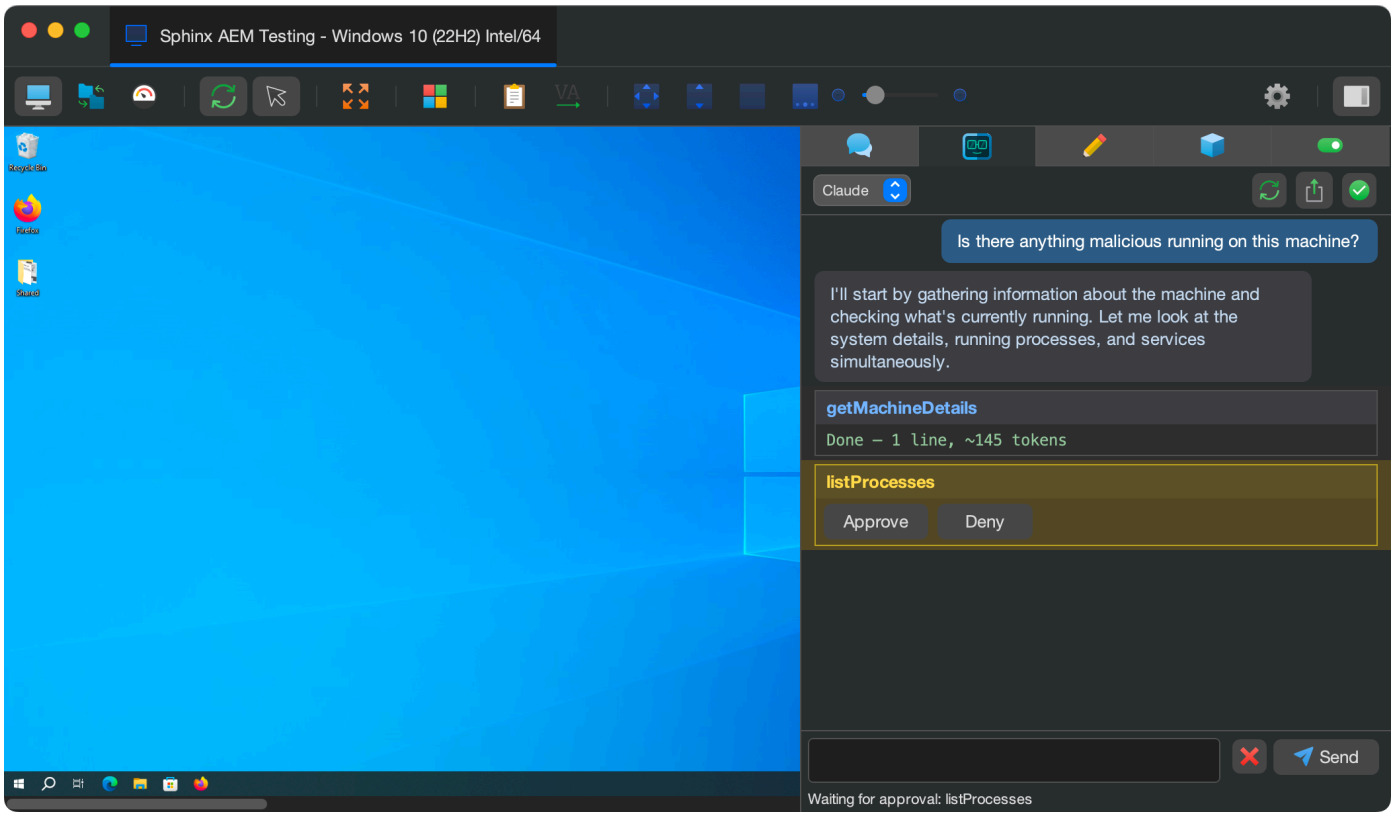
4.2 Cyana In Session

Cyana In Session is used inside a live support or access session to a single remote machine. When a technician is already connected to a machine via a SimpleHelp session, Cyana is present and waiting to help and shares their view of that specific machine.

Scope. Cyana In Session is scoped to the machine of the current live session, and no other. Its tools do not include machine discovery and instead are tailored to the current machine. The conversation is owned by the technician who started it, so a second technician cannot pick up the same conversation and act on the same session.

Computer control. Cyana In Session is the only mode in which Cyana can control mouse and keyboard on a remote machine. Computer control is off by default and in each session requires the technician to explicitly enable it with a toggle on the Cyana panel. The toggle itself is only enabled if the technician's group has the "Control in-session mouse and keyboard" permission. Enabling the toggle raises a confirmation dialog on the first click of the session. The grant is per conversation and per technician client instance, and is cleared when the chat is cleared.

Screenshots inside the live view. In session, the `getScreenshot` tool allows Cyana to view the remote machine as the technician does. Note that this requires a backend model endpoint which has this capability. Some models are text-only whereas others have 'vision' and can see and interpret images and computer screens.



Cyana In Session, in dark mode, panelled beside a live Windows session. Cyana has completed the read only `getMachineDetails` step automatically. Its next proposed step, `listProcesses`, is presented for the technician to Approve or Deny before it runs.

4.3 Cyana Toolbox

Cyana Toolbox is used inside the SimpleHelp Toolbox script editor. It helps the technician read, adjust and rewrite the currently open script.

Scope. Cyana Toolbox has no access to machines, files on disk, commands or services, cannot browse the fleet, and cannot run scripts. Its entire tool universe is a short list of methods that read the current script, search within it, replace its contents, or replace a specific line range, and nothing else is offered.

That narrow universe is enforced by the orchestrator for this mode registering only the Toolbox script tools. There is no ability for Cyana to escape the mode from within a Toolbox conversation.

Data loss prevention still applies. Even in Toolbox mode, the outbound scanner runs on every proposed script write. If a technician were to ask Cyana to embed a value that matches a piece of the server's protected configuration, the write is blocked. Section 8.4 describes what is scanned.

---

5. Permissions, Scope and Technician Approval

Cyana's control model has four layers that all have to line up before Cyana can take an action on behalf of a technician. This section describes each layer.

5.1 Enablement Layers

Cyana is only available when all of the following are true:

1. **Feature is present.** Cyana is a supported feature of SimpleHelp 6.0.
2. **Cyana is licensed.** An AI licence must be present in the SimpleHelp account linked to the SimpleHelp server. Without one, Cyana is disabled in the technician client, and no Cyana transaction can be executed. Administrators can request a trial from the licence panel if their SimpleHelp account is linked.
3. **The server has AI enabled.** A server wide "Use Cyana with this server" checkbox, in the Core Features section of the server settings, controls whether Cyana is enabled at all. This is on by default in supported builds. Turning it off disables Cyana at the server, regardless of licence and regardless of per group settings.
4. **The technician's group has Cyana permissions.** A group has to have the "Allow Cyana" permission enabled at minimum. Each of the more specific Cyana capabilities (file read, file modify, command execution, and so on) is a separate permission that is only meaningful if "Allow Cyana" is enabled.
5. **At least one AI endpoint is configured.** Administrators configure endpoints in the Administration section. An endpoint has a name, URL, model, provider, API key, and a per endpoint system prompt. Endpoints and their keys are stored on the SimpleHelp server; keys are encrypted at rest with the same mechanism used for other server configuration secrets. A technician who opens Cyana with no endpoints configured sees a "not configured" panel rather than a chat.

Cyana is only reachable for a technician when the feature is present, licensed, enabled at the server, permitted by their group, and pointed at a configured endpoint.

Default posture. Once licensed, Cyana defaults to on so that technicians can open the panel and converse with Cyana. Every category of capability, from file read to command execution to session control, is controlled by a separate Cyana group permission that the administrator can independently disable. These group permissions are the outer envelope: they define what Cyana is permitted to be asked to do for that group. They do not authorise any specific action. Every proposed tool call is still subject to the technician approval gate described in section 5.4, which starts fresh at the beginning of every chat with no auto

approval granted for anything except a small set of clearly non destructive enumeration and display tools. Administrators who want a stricter posture do not have to disable Cyana at all: they can leave the chat available and turn off the specific capabilities they do not want the group to have. Turning off "Modify files and folders", for example, removes every file writing tool from the tool universe Cyana ever sees for that group.

5.2 Fine Grained Cyana Permissions

These permissions form the outer envelope of what Cyana can be asked to do for a group. Grants at this level authorise possibility, not execution: every tool call still passes through the per call approval gate in section 5.4, and every tool call still passes through the automated review in section 6 if it is in the higher risk set.

Beyond the master "Allow Cyana" permission, Cyana exposes ten additional permissions on the technician group, each of which gates a category of tools. Every permission is separately controllable in the Cyana section of the administrator's group permissions panel. The categories are:

Permission	Category	Example tools
"Use web research" (`canCyanaUseWebResearch`)	Outbound	`webResearch`, `fetchWebPage` inside the sandboxed research agent
"View machine details and metrics" (`canCyanaViewMachineDetails`)	Read only inspection	`getMachineDetails`; also gates detailed fields on `getMachineInfo`
"View screenshots" (`canCyanaViewScreenshots`)	Read only inspection	`getScreenshot`
"Read files and folders" (`canCyanaReadFiles`)	Read only inspection	`readFile`, `readFileLines`, `readBinaryFile`, `listDirectory`, `listDirectoryDetailed`, `searchInFiles`, `findFiles`, `pathExists`, `resolvePath`, `getPathInfo`
"Modify files and folders" (`canCyanaModifyFiles`)	Machine affecting	`writeFile`, `writeBinaryFile`, `createDirectory`, `deletePath`, `movePath`, `copyFile`, `searchAndReplace`, `applyTextEdit`
"Run commands and scripts" (`canCyanaRunCommands`)	Machine affecting	`runCommand`, `runGroovyScript`, `getEnvironmentVariables`
"Manage processes and services" (`canCyanaManageProcessesAndServices`)	Machine affecting	`listProcesses`, `killProcess`, `listServices`, `startService`, `stopService`, `restartService`
"View machine tags" (`canCyanaViewTags`)	Read only inspection	`getTags`
"Assign machine tags" (`canCyanaAssignTags`)	Machine affecting (requires "View machine tags")	`addTag`, `removeTag`, `setTags`
"Control in-session mouse and keyboard" (`canCyanaControlSession`)	Session affecting (only in Cyana In Session)	`typeText`, `pressKeys`, `moveMouse`, `clickMouse`, `scrollMouse`, `dragMouse`

Permissions are quoted as they appear in the Cyana section of the technician group permissions panel. The name in brackets is the internal identifier used where group settings are serialised into server configuration, included here for administrators who audit group settings on the server rather than through the UI.

This design has two practical consequences.

First, a permission that has not been granted is not merely refused at execution time, but is also excluded from the list of tools that Cyana is told about. This means Cyana cannot propose or try a tool it has no permission for, because it does not know that tool exists in the current conversation.

Second, a technician's own group permissions can be merged with per machine group permissions. This is a SimpleHelp platform behaviour that Cyana honours. A technician who has broad Cyana permissions in general can still be restricted from destructive tools on a specific machine group by the administrator's normal group and tag permission settings, and Cyana's per call authorisation check enforces the merged result.

5.3 Scope Selection and Enforcement

Command scope. In Cyana Command, scope is chosen by the technician from a combo box on the conversation pane. The choice is transmitted to the server as an encoded path of the selected machine group. On the server, every proposed tool call is checked against that scope: the target machine must exist, must live within the selected group path (with the exact group, or a subgroup, allowed), and must be visible to the technician under normal SimpleHelp visibility rules. A tool call that violates any of these fails on the server before dispatch. Discovery tools like `listMachines` are also filtered by the same scope.

Session scope. In Cyana In Session, scope is fixed to the machine of the current session. There is no machine selection in the panel and no discovery tool. The conversation itself is owned by the technician: poll, submit and cancel operations from any other technician are refused on the server. Machine control grants (mouse and keyboard) are additionally bound to the specific client instance that requested them, so a second client instance signed in as the same technician cannot inherit an in flight machine control grant.

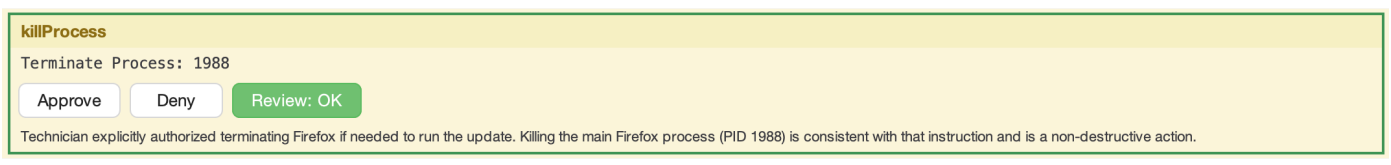
Toolbox scope. In Cyana Toolbox, there is no machine scope. Cyana's tools operate only on the currently open script.

Per conversation ownership. Every Cyana conversation is owned by the technician who started it. Poll, submit and cancel operations from any other technician are refused on the server. This prevents another logged in technician from taking over a Cyana conversation to approve or deny an action on the original technician's behalf.

5.4 Technician Approval

Every Cyana chat starts in maximum required approval mode (minimum autonomous capability mode). Only a small set of clearly non destructive tools, described below, run without a per call technician prompt. Every other tool the model proposes waits for the technician's explicit Approve or Deny before it runs.

The tools that run without a prompt are limited to fleet enumeration and high level machine information (for example listing machines, summarising machine info, retrieving hardware and OS details), tag reads, chat rendering helpers, time and wait utilities, and inspection of Cyana's own stored tool results. They produce information Cyana needs to answer questions and cannot on their own change anything on a machine. Every other tool the model proposes waits for the technician's Approve or Deny before it runs: file reads on a specific path, directory listings, file writes, deletes, moves, copies, commands, scripts, process and service control, screenshots, tag writes, and any session mouse or keyboard action.



A `killProcess` approval prompt, showing the process Cyana proposes to terminate, explicit Approve / Deny buttons, and the automated review's OK rating with its reasoning: the proposal matches what the technician authorised earlier in the conversation.

Approval timeout. If a technician does not approve within two minutes, the tool call times out. The model sees an ordinary error result for the tool call and can decide what to do next, which is usually to ask a clarifying question or try a different approach. Nothing runs during that wait.

Denials do not stop the conversation. When a technician denies a tool call, the model sees a structured denial and can continue the conversation. It can revise its approach, propose a different tool, or ask the technician for guidance. Denial is treated as feedback rather than as termination, but the model has no autonomous way to circumvent the denial.

5.5 Managing Approval Load

Approval fatigue is a genuine problem. When a technician is investigating a fleet issue that requires many file reads across many machines, prompting for each read is not useful. Cyana's approach is to make the technician's trust decisions bounded, explicit and reversible, so that the technician can loosen approval where they judge it safe and appropriate for the environment they are working within without the platform assuming that judgement for them.

Auto approve dialog. A technician can open an auto approve dialog that offers seven tool categories: Screenshots, Search results, Read (file contents), Write (create, delete, move, copy), Commands, Scripts (Groovy, apply text edit, search and replace), and Process (list, kill, start, stop, restart of processes and services). A separate list on the same dialog manages trusted domains for `fetchWebPage` calls that occur inside the sandboxed web research agent. Selecting a category means Cyana can run tools in that category for the rest of the current chat, without a per call prompt.

Four bounds on that grant.

- **Warning acknowledgement.** Every category checkbox is disabled until the technician ticks a warning acknowledgement in the same dialog. Unticking the acknowledgement deselects every category. There is no way to grant auto approve without the technician engaging with the warning that Cyana will be able to take autonomous actions based on the explicit grants. The acknowledgement is not a one time dismissal, it applies per chat.
- **Session scope.** Grants live only for the current chat. Nothing about auto approve is persisted to the server, no setting carries between chats, and closing the panel or clearing the chat resets every selection. The next chat, next session, or next customer begins again at maximum approval / minimal autonomous capability.
- **Envelope bound.** A technician cannot grant more than the administrator has permitted the group. If "Run commands and scripts" is off for the group, ticking the Commands category has no effect: the tools are not in Cyana's tool universe at all for that group.
- **Red review overrides.** Even for a category the technician has auto approved for the current chat, a red rating from the automated review agent forces an explicit approval prompt for that specific tool call. Section 6 describes how the review works.

This model makes approval load workable without collapsing the point at which a human decision is made. The default is safe, loosening is deliberate, and the scope of any loosening is small, bounded and revocable, while the automated review remains a second signal that the platform can use to override the technician's own working posture when a specific action looks wrong.

Why forcing all manual approval is not automatically more secure. It is tempting to assume that requiring a technician to approve every tool call produces a more careful technician. Decades of evidence from security human factors research show the opposite. When a person is asked to approve too many things too often, the approval collapses into a reflex click and the attention that was meant to sit behind it disappears.

Windows UAC is a broadly known and well established case. When Vista prompted for consent on almost every privileged action, users habituated to clicking Yes so quickly that malware could trigger UAC and ride on the reflex. Microsoft reduced the prompt frequency substantially in Windows 7 largely for this reason. Users learn quickly that the base rate of genuinely dangerous prompts is low, and rationally stop reading them. Browser TLS warnings have shown the same at scale, with click through rates measured between 33 and 70 percent depending on browser implementation and warning design (Akhawe and Felt 2013), enough to make them close to worthless as a safety mechanism. In medicine, the Joint Commission classifies alarm fatigue as one of the leading contributors to adverse events in hospitals (Sentinel Event Alert 50, 2013). The same pattern has been documented in automated driving, aviation cockpit warnings, industrial control room alarms, and clinical decision support.

The common thread is that a human's attention budget is finite, and a system that spends that budget on routine confirmations has none left for the case that actually matters. In security human factors research this is closer to a settled result than a debate.

Cyana's design assumes the same principle. A technician who has consciously granted auto approve for Search results and Read in the current chat, after choosing which categories to trust and ticking the warning acknowledgement, is more likely to notice something odd in a `runCommand` proposal than a technician who is clicking Approve on every routine listing in the same investigation. A technician managing Cyana permission grants appropriate to the environment and context is not a weakening of the platform safety, it is platform safety working correctly and as designed. Cyana's design directs the technician's attention to where it matters, keeps the admin permission envelope as an outer bound, and keeps the automated review as a second signal that can override the technician's working posture on any specific proposal. Organisations that want a stricter posture should tighten the admin permission envelope rather than force every action through a manual prompt, because the manual prompt collapses to a reflex under load and the envelope does not.

Permissions are security control; approval is attention control. The two layers of Cyana's model do different jobs. The Cyana permission envelope decides what Cyana can ever be asked to do for a technician group, and it binds every principal that reaches Cyana: tired technicians, junior technicians, and misunderstanding or misbehaving models. It also handles the compromised technician account case. An attacker in possession of a technician's credentials already holds native SimpleHelp remote access and can run commands directly, without Cyana being involved, so Cyana grants that attacker no capability beyond what the platform permission model already gave the account. Because the Cyana envelope merges with those platform permissions (see section 5.2), the layer that bounds legitimate Cyana activity also bounds the attacker's Cyana activity, and the platform permission model beneath it bounds the attacker's native activity for the same reason. Per call approval and automated review decide where a legitimate technician's finite attention gets spent inside that envelope, catching model error or injection in front of the one person positioned to notice.

Trust differences between teams belong in the envelope too, not in differential prompting. If team A is trusted with Cyana running commands and team B is not, the correct answer is not to force team B through more manual clicks in the hope that the extra clicks catch a bad proposal, but to give team B "Allow Cyana" plus a read only set of Cyana permissions, so that Cyana becomes an advisor that drafts and suggests while a technician on team B executes through normal SimpleHelp tooling. That changes what is possible for team B, rather than how often they click.

---

6. Automated Review For Higher Risk Actions

Automated review is the second safety signal Cyana uses. Automated review is conducted by a separate agent in a customised context, distinct from the acting agent, with no tools of its own, and its recommendation is surfaced to the technician alongside the approval prompt. Where the review is red, the recommendation is enforced: no auto approve setting can skip the technician prompt.

6.1 What Review Is

When Cyana proposes to run a higher risk tool, and at least one endpoint is configured for review, the server calls out to that endpoint with a small, structured payload that describes the proposed action and its context. The endpoint returns a JSON rating of green, yellow or red, together with a short comment. The technician client displays that rating and comment inline with the approval prompt for the action.

The reviewer does not execute any tools, never sees prior tool results, and has no ability to change execution behaviour beyond the traffic light rating it produces.

6.2 Which Actions Are Reviewed

Review only runs for a defined list of higher risk tools:

`runCommand`, `runGroovyScript`, `writeFile`, `writeBinaryFile`, `applyTextEdit`, `searchAndReplace`, `deletePath`, `movePath`, `copyFile`, `createDirectory`, `killProcess`, `startService`, `stopService`, `restartService`, `webResearch`, `fetchWebPage`.

Read only inspection tools (file reads, listings, machine details, screenshots, tag reads, process listings) do not go through review. Read only tools are already subject to technician approval and to the Cyana permission model, and are considered low enough risk that the additional review cost is not justified.

Review applies whatever machine context Cyana is running in: in Cyana Command scoped to any machine set, in session, and when running tools on the technician's own machine.

6.3 What The Reviewer Sees

The review payload is built from a small number of well defined sections. Every field is wrapped inside a `<review_input untrusted="true">` element, and any occurrence of the same closing tag inside the content is escaped, so untrusted content cannot forge the terminator of its own container. The sections are:

- **Execution context.** The Cyana mode, the selected machine identifier if any, the group filter, and whether a live session is attached.
- **Recent technician messages.** The most recent technician prompts, oldest to newest, so the reviewer can see what the technician asked for.

- **Recent Cyana messages.** The most recent text or completion messages from Cyana, oldest to newest, so the reviewer can see what Cyana said it was going to do.
- **Tool name.**
- **Tool arguments as shown to the technician.** Machine identifiers are resolved to friendly names in this section.
- **Raw tool arguments.**

The reviewer is asked to return a JSON object of the form `{"rating": "green|yellow|red", "comment": "..."}`. It is not asked for anything else, and its response is parsed strictly.

The following are deliberately excluded from the review payload: prior tool results and their contents, machine information payloads, screenshots, secrets, and the full transcript. The reviewer sees the summarised execution context and the last few conversational turns only.

Why the reviewer is deliberately blind. This exclusion is not a limitation but a design decision, made for two reasons.

First, tool results are exactly where hostile content may live. A malicious page fetched during web research, or an attacker authored file returned by a `readFile` call, is untrusted content that has crossed into the conversation. Untrusted markers and model development help to keep the primary agent model from trusting or obeying such content, but they are not a guarantee. Passing those same contents to the reviewer would put the reviewer within reach of the same injection surface Cyana itself has to handle. A reviewer that reads a hostile tool result and rates the next action green because the tool result talked it into doing so is a reviewer that has become compromised by the same injection rather than a defence against it.

Second, the reviewer does not need the injected content in order to do its job. The reviewer's job is a consistency check between what the technician asked Cyana to do and what Cyana is about to do. An injection driven action typically manifests as exactly that inconsistency: the technician asked for a disk space summary and Cyana is proposing to `runCommand` something that has nothing to do with disk, so the reviewer can catch the symptom without ever needing to see the cause.

There is one caveat to note here. The raw tool arguments Cyana proposes are model authored and are in the payload. The reviewer is therefore not entirely sealed from potential adversarial content: it can still see whatever Cyana produced as tool arguments. Two controls address this. The `<review_input untrusted="true">` wrapping described above marks the arguments as untrusted evidence, and the reviewer's response is parsed strictly against a small JSON shape that only allows a rating and a comment. There is no path from an argument value through the reviewer back into an executable instruction. More importantly, the rating's only enforcement power is to add friction: a red forces a manual prompt, while a green changes nothing. A reviewer that was talked into a green rating therefore degrades to the same state as no review at all, with the permission envelope, scope enforcement and technician approval still in force. Manipulating the reviewer can remove a second signal, but it cannot grant a capability or approve an action.

6.4 Traffic Light Outcomes And Effect On Execution

The three ratings are displayed to the technician as OK, Caution and Warning. The internal codes are green, yellow and red.

- **Green (OK).** The action looks consistent with the request and low risk in the current context. No behaviour change on the server side. The technician still sees the ordinary approval prompt, and any

auto approve setting the technician has for the category still applies.

- **Yellow (Caution).** The action may be reasonable, but the technician should look at the details carefully. The rating and comment are displayed prominently on the approval prompt. No enforcement change.
- **Red (Warning).** The action looks potentially destructive, too broad, or inconsistent with what was asked. Any auto approve setting for the category is disregarded, and the technician has to explicitly approve or deny the action. The red rating is designed as a stop signal that the technician has to actively override.

The reviewer's short comment is displayed alongside the rating so that the technician can see why the action was flagged, and can decide whether the reviewer's concern applies to the case in front of them.

6.5 What Happens When Review Fails

Administrators configure endpoints in the same panel used for chat endpoints, and may mark them as review endpoints using a checkbox next to each endpoint. The checkbox is on by default for new endpoints, so administrators who add an endpoint without changing settings automatically enable review on it. If multiple endpoints are marked for review, Cyana tries them in the order the administrator has arranged, and stops at the first endpoint that returns a valid rating.

Review runs with a 30 second timeout. If an endpoint is unavailable, returns invalid output, or times out, Cyana moves to the next configured review endpoint. If every review endpoint fails, Cyana falls back to a synthetic red rating with a comment of the form `Review failed: ...`. That means when review is configured but cannot answer, Cyana takes the conservative path and requires manual approval. The behaviour is fail closed by design.

If no review endpoint is configured at all, higher risk tools still require the ordinary technician approval, but no automated review runs and no traffic light indicator is displayed. Cyana does not silently skip approval when review is unavailable.

---

7. Data Flow And Model Endpoints

This section describes the lifecycle of a Cyana request. Every step is enforced in server code. Where a step is configuration dependent, that is called out.

1. **Technician opens a Cyana mode.** The technician client picks the operating mode based on the surface the technician opened (main technician window for Command, session panel for In Session, Toolbox editor for Toolbox). The server starts a conversation owned by the technician, and if the mode requires it, records the scope of the conversation.
2. **The SimpleHelp server builds context from the selected scope and administrator settings.** For Cyana Command, the scope path is retained on the server and used to filter every machine discovery and validate every tool call. For Cyana In Session, the session identifier and the technician's client instance are recorded, and used to bind the conversation to that specific session and client. For Cyana Toolbox, only the current script is in scope.
3. **The selected model endpoint receives the prompt and the allowed context.** The technician picks the endpoint from a dropdown in the chat panel. The server sends the technician's prompt, the system prompt, and the tool definitions permitted for the mode and permission set. Tool definitions

that the technician's group does not permit are not sent, so the model cannot propose a tool it is not permitted to run.

4. **Cyana proposes tool calls when it needs product or machine access.** Model responses that include tool calls are routed to the corresponding orchestrator on the server.
5. **Permission and technician approval gates apply before execution.** Every tool call is authorised against the mode's tool universe, against the technician's permissions merged with the machine's group permissions where applicable, against the scope of the conversation, and against ownership by the technician. Tools that require approval trigger an approval prompt in the technician's client. Tools that are annotated as not requiring approval are auto approved for the current step only.
6. **Higher risk actions may be reviewed by a separate review endpoint.** For the higher risk tool list in section 6.2, the review payload is built and dispatched. The result of the review is attached to the approval prompt if one is being shown, and enforces manual approval on a red rating.
7. **Approved tool calls run through SimpleHelp's existing platform access.** Cyana Command uses SimpleHelp's usual remote messaging channel to reach the target machine. Cyana In Session uses the same live session the technician is inside. Cyana Toolbox operates on the script the technician has open. Cyana has no independent path to any endpoint.
8. **Tool output is returned to the conversation with untrusted input handling.** The tool result is placed into the conversation as a message wrapped in an `<tool_result tool="..." untrusted="true">...</tool_result>` element. Any occurrence of the same closing tag inside the content is escaped so untrusted content cannot forge its own terminator. This framing is a signal to the model that the content is untrusted data, not additional instructions. Section 8.1 discusses the limits of that control.
9. **Outbound capable actions are checked for protected configuration data.** Any tool call whose method is marked as outbound capable is scanned for known protected server configuration values before the call is dispatched or relayed. A match blocks the call before it reaches the target, and writes an entry into the administrator activity log. See section 8.4.
10. **Relevant actions, decisions and outputs are logged.** Every message and every tool call flows through the conversation update record, and on terminal states (completion, cancellation or error) the transcript is persisted for later inspection. In session Cyana events are interleaved with the parent session's history. Standalone Cyana conversations are stored as their own history session. Section 10 discusses what is captured and how it can be reviewed.

Cyana supports frontier providers natively and any OpenAI compatible endpoint, including self hosted ones. Section 9 covers the endpoint model in full.

---

8. Input Handling, Output Filtering And Sensitive Data Controls

This section explains what Cyana actually does at each control boundary, and what it does not do. Each control is described by what it *does*, not by what it *prevents*, unless code makes a strict guarantee of this.

8.1 Marking Tool Output As Untrusted

Every tool result that flows back into the conversation is wrapped by the server inside an `<tool_result tool="..." untrusted="true">...</tool_result>` element, and any occurrence of the same closing tag inside the content is escaped as `<\/tool_result>` so untrusted content cannot forge the terminator of its own container.

The purpose of this framing is to make the trust boundary explicit and machine visible. It is a signal to the model and a defence in depth control rather than a parser level guarantee. A model that ignores the label can still act on injected instructions inside a tool result. In line with that reality, Cyana relies on the additional controls in the rest of this section, on the approval gate in section 5.4, and on the automated review in section 6.

8.2 Sandboxed Web Research

Web research is not executed in the same conversation as the technician's chat. When Cyana calls `webResearch`, a separate research orchestrator is created for that call. The research orchestrator has one tool available to it, `fetchWebPage`, and nothing else. It has no history from the parent conversation. It has no access to any machine, any file, any command or any of the SimpleHelp platform.

Two additional controls apply.

- The research orchestrator's system prompt tells it to treat pages as information only, not to follow instructions in them, and to end its reply with a specific canary line that is randomly generated for the request. On return, the server validates that the canary appears exactly once and as the final line. If it does not, the returned research is discarded and Cyana receives a synthetic error result instead of the research agent's response.
- The `fetchWebPage` executor enforces a strict URL policy pre connect and again on redirect. Only `http://` and `https://` schemes are permitted. URLs with user info, opaque authority, control characters, or unusual encodings are rejected. IPv4 and IPv6 loopback, link local, RFC 1918, cloud metadata addresses (including `169.254.169.254`), and other reserved ranges are blocked. Hostnames that DNS resolve to any blocked address are rejected. Redirects are capped at five and each destination is re validated. Response bodies are read with a total character cap.

Together, these controls compartmentalise web research: if a page tries to influence Cyana through injection, the affected agent has no destructive tools of its own, and its output either passes a syntactic freshness check or is discarded. This is best effort compartmentalisation rather than a proof of injection resistance. A model that continues to append the canary while nonetheless carrying influence in the summary text cannot be caught by the canary check alone. The rest of Cyana's controls are the reason this compartmentalisation is sufficient in practice.

8.3 HTML And SVG Rendering Safety

Cyana can return two kinds of visual output to the technician's chat panel: an HTML report, and an SVG image such as a chart, diagram or network map. Each has its own rendering path with its own controls, and both are sanitised before they are written or rendered.

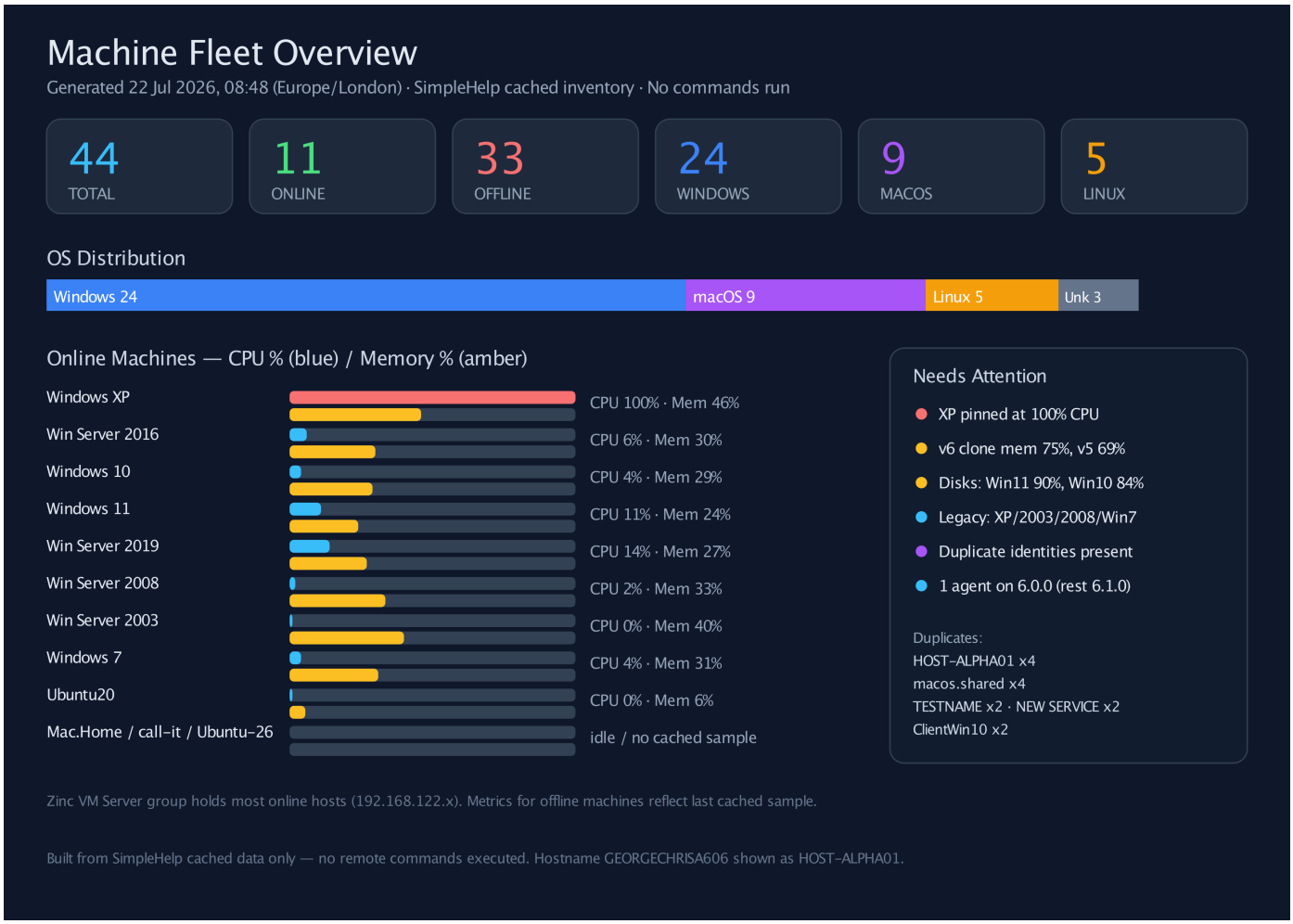
HTML reports. Sanitisation applies a strict tag and attribute allowlist that keeps report style content (headings, tables, lists, code blocks, links, images and static inline SVG drawings) and removes active content, event handlers, styles, external resources, forms and inputs, together with `<math>`, `<canvas>`, `<iframe>`, `<object>` and `<embed>`. Anchor `href` values are permitted only if empty or a `#` fragment.

Image `src` values are permitted only if they are `...` URIs for raster formats (PNG, JPEG, GIF, WebP); SVG data URIs are not accepted. All `on*` event handlers are stripped. The whole result is wrapped in a document with a restrictive Content Security Policy that disallows script, external images, external fonts, external styles and form actions. The sanitiser runs when the server prepares a report for delivery, and again on the client side when the report is rendered into the panel, providing defence in depth.

Inline SVG inside reports. Within an `<svg>` subtree the sanitiser switches to a static drawing allowlist: shapes, paths, lines, text and gradients, plus grouping and definition elements. `script`, `style` and `foreignObject` elements are removed together with their entire contents; animation, filter, `image` and `use` elements are also removed. SVG attributes are allowlisted in the same way, and an attribute value is rejected outright if it contains a URL scheme or path separator, which blocks external references in any form; paint server references are permitted only in same document `url(#id)` form. An `<svg>` element left unclosed is closed by the sanitiser, so following report content cannot be pulled inside it.

SVG images. SVG delivered to the chat panel is never handed to a browser engine for display. Before rendering, sanitisation strips `<script>` elements and their contents and removes all `on*` event handler attributes from the markup. The stripped SVG is then rasterised by an embedded SVG renderer into a static bitmap, and that bitmap is what appears in the chat panel. The renderer contains no script engine, so script, animation or interaction content has no execution environment in the panel — the displayed artefact is a plain image. The tool that produces SVG also instructs the model to generate static markup without animations, filters or `foreignObject` elements, though that instruction is guidance to the model rather than an enforced control. The technician can export the stripped SVG markup or a PNG rasterisation of it. A view option can also open the image in the technician's browser: that path reduces the SVG to the same static drawing allowlist used for reports and wraps it in the same restrictive Content Security Policy document.

This is a rendering safety control: it protects the technician's rendering environment from active content the model tried to emit inside a report or image, but it does not attempt to detect malicious intent inside the text of a report or the labels of a chart.



An example SVG report generated by Cyana. SVG output is stripped of script and event handler content, then rasterised to a static image by an embedded renderer before display in the chat panel; HTML reports are sanitised on the server and again on the client, and wrapped in a restrictive Content Security Policy.

8.4 Outbound Data Loss Prevention

Every tool call that is capable of exfiltrating data outside the SimpleHelp environment carries an `OutboundTool` marker. Before any such call is dispatched or relayed, the server scans its arguments against a set of known protected server configuration values. Detections block the call, and log an entry to the administrator activity log.

The set of protected values is derived from live SimpleHelp server configuration and includes:

- The master static salt used for key derivation.
- The server queue password.
- The DNS authentication token used by the SimpleHelp DNS integration.
- Encrypted representations of the Remote Work site API token and site registration email.
- Encrypted representations of the SSL keystore store password and key password.
- Per technician hashed passwords and encrypted TOTP keys.

Values shorter than twelve characters are excluded from the scanning set to avoid false positives on trivial substrings. Detection uses exact substring matching against the raw JSON arguments of the outbound tool call.

When a detection fires, Cyana returns a generic block result to the model of the form `Blocked by server content policy: tool arguments contain sensitive server configuration data that cannot be sent out. Rephrase your request without including that data.` The message deliberately does not disclose which value matched. An entry of the form `AI tool call '<name>' blocked by data-loss-prevention scan (conversation <id>)` is written to the administrator activity log. The technician sees the block message in the chat.

The outbound tool set is enforced by a dedicated test that requires the highest risk exfiltration methods (`runCommand` and `runGroovyScript` on every applicable interface) to carry the marker. This test is a backstop that fails the build if a refactor accidentally drops the annotation from a tool method.

Limits. This control is precise about what it does. It scans arguments to outbound capable tools, not tool results returning to Cyana and not Cyana's free text replies to the technician. It matches exact substrings of live server configuration, not general secrets, credit card numbers, PII, customer credentials, or anything the technician typed into the chat. The `getEnvironmentVariables` tool has its own separate defence: an allowlist of safe variable names on the machine that runs the tool, with other values redacted at source before they enter the tool response. Section 8.5 covers a separate log level control.

One layer among several, deliberately. Cyana's outbound scan is defence in depth, and this is a general pattern across SimpleHelp: layered controls so that if one boundary fails, another still blocks. The outbound scan sits alongside per call technician approval, admin envelope permissions, scope enforcement and automated review, and the strongest thing to say about it is that a value the technician would not want to leak has to get past all of these to reach the outside. Constraining an LLM by output scanning alone is not a solvable problem, and this paper deliberately does not claim otherwise. Exact substring matching does not detect encoded, transformed, chunked or paraphrased forms of the same value, and it is not intended to. Its job is to stop the straightforward case cheaply, log the block for an administrator to see, and let the other controls carry the adversarial cases.

8.5 Diagnostic Log Hygiene

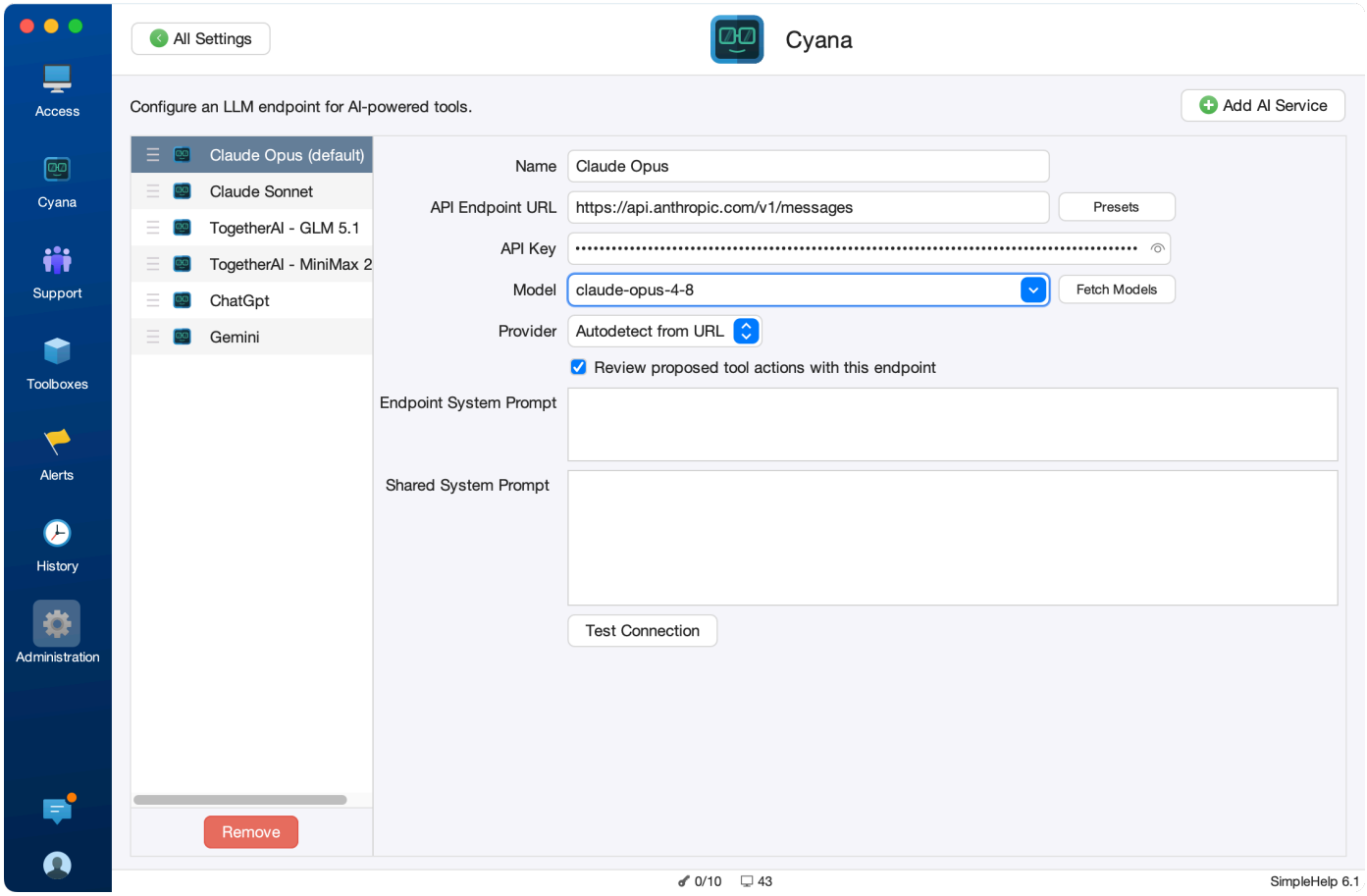
Server side diagnostic logs written by the Cyana subsystem never include prompt bodies, tool arguments, tool results, or provider response bodies. A diagnostic log sanitiser is used at every point where those payloads might otherwise be captured, and it emits size metadata only. Provider errors have their body content restricted to a small allowlisted subset of error message, error type, error param and error code fields, with a length cap and collapsed whitespace. Exception messages are replaced by the exception class name. URLs are stripped of `key=`, `api_key=` and `access_token=` query parameters before logging.

This is log hygiene rather than a runtime security control. It stops the server's own debug logs from containing message content that would otherwise be sensitive.

---

9. Self Hosted And Customer Selected Models

Cyana works with a model endpoint you choose. There is built in support for the frontier model providers and their APIs. There is also broad support for self hosted endpoints and for custom endpoints reached over OpenAI compatible protocols.



The AI endpoint configuration panel in the Administration section. Administrators list, add, remove and edit AI endpoints. The API key is masked in the UI, held encrypted on the server, and never returned to technician clients.

Frontier providers. Anthropic (Claude), OpenAI (GPT) and Google (Gemini) are supported as first class integrations using each provider's native API. Anthropic uses the Messages API with `x-api-key` authentication. Google Gemini uses the generative language API with URL parameter authentication and content parts. OpenAI is auto detected between Chat Completions and Responses API depending on the URL. In each case, keys are stored encrypted on the SimpleHelp server and are never returned to technician clients. Technicians whose group has "Allow Cyana" see the endpoint list without keys so they can pick which endpoint a conversation uses.

Azure OpenAI, Mistral and Cohere. These are configurable as their own provider values and are handled through the OpenAI compatible chat completions path.

OpenAI compatible endpoints. A generic OpenAI compatible option accepts any endpoint that speaks Chat Completions. This includes self hosted stacks (Ollama, llama.cpp, LM Studio, vLLM, Krasis and similar), enterprise inference gateways, and OpenAI compatible cloud providers. Krasis is designed to run very large models on consumer grade GPU hardware, and is a route to a fully local Cyana backend without datacentre equipment. The URL normalisation logic is forgiving: base URLs without a path are treated as if they end in `/v1/chat/completions`, and a wrong path 404 triggers an automatic retry against the standard path.

Hosted OpenAI compatible providers. Cyana recognises common hosted providers such as Together, Fireworks, Anyscale, Groq, Perplexity, DeepInfra, OpenRouter, xAI, DeepSeek and Cerebras by URL, and picks sensible defaults for each.

Auto detect. An `Autodetect from URL` mode identifies the provider by the endpoint URL. Administrators who want to pick a provider explicitly can do so from a dropdown.

Data residency. For organisations that need data to stay inside their own infrastructure, the same configuration surface accepts local URLs (including localhost) and internal DNS names. Cyana does not require SimpleHelp cloud contact for its normal operation. As part of the standard SimpleHelp version check that every server performs, aggregate Cyana usage counters (total prompt count, per mode prompt counts, number of unique technicians who have used Cyana, and number of configured LLM endpoints) are included in the posted payload. No prompt content, tool arguments, tool results or model responses are included, and no Cyana traffic goes to the cloud outside that periodic version check.

Endpoint routing by use case. Cyana distinguishes between chat endpoints and review endpoints. The same physical endpoint can serve both, or an administrator can configure a higher grade model to be used only for review by ticking its review checkbox and unticking others. In practice, many organisations use a strong general purpose model for chat and a strong instruction following model for review.

What is not supported today. Cyana does not currently support mutual TLS or custom authentication headers beyond the provider standards (`x-api-key`, `Authorization: Bearer`, `key=` query parameter). Deployments that need mTLS should place a reverse proxy in front of the endpoint.

---

10. Audit, Accountability And Operational Review

Every Cyana conversation is captured in the SimpleHelp history. Administrators and technicians with the appropriate view permission can inspect what Cyana was asked, what it decided to do, what the technician approved, what tools ran, what they returned, and how the conversation ended.

What is captured. For each conversation, the persisted record includes:

- The technician who owned the conversation, with username, display name and IP address.
- Start time and duration.
- The transcript, consisting of the ordered list of conversation updates: technician prompts, Cyana text replies, tool call proposals, review outcomes attached to tool call proposals, tool call results, and completion or error status.
- For in session conversations, the associated session identifier so that the AI events sit inside the parent session record.
- For standalone conversations, a dedicated AI session record is registered in the history.

Review outcomes (green, yellow or red) and reviewer comments are recorded on the tool call proposal record, so an operational review can see which actions were flagged and how the technician handled the flag.

What is not captured. The persisted schema does not, at the time of writing, include a per turn record of which endpoint or model was in use. Administrators who want an audit story that includes which model produced which decision should currently treat that as a per conversation approximation (technicians pick an endpoint at the top of the panel).

Where it lives. The history is stored on the SimpleHelp server, alongside every other kind of session record. The Cyana chat panel has a PDF export for the current conversation, useful for producing an evidence pack for a specific incident.

Who can see it. History access follows the standard SimpleHelp view permissions: view own, view group, view all. There is no separate view permission for AI events at present.

Administrator activity log. Independently of the conversation transcript, an entry is written to the administrator activity log when the outbound content policy blocks a tool call. That entry names the tool and conversation identifier and is reviewable by administrators who have access to the activity log.

---

11. Recommended Rollout Approach

Cyana is a productivity tool. Its value shows up when technicians actually use it, and its safety story is designed to make that use comfortable rather than to make it slow. The approach below reflects that principle. Enable Cyana, keep the technician in full approval mode at the start, and expand what Cyana is trusted to do without a prompt as the team's confidence in the specific model they are using builds up. Trust is earned, per model and per environment, by watching the model behave in real workflows.

11.1 Enable And Start In Full Approval

1. Configure one approved model endpoint. Prefer a well understood model for the first rollout, for example a frontier model from a leading provider.
2. Decide whether cloud or self hosted is appropriate for the environment. Where data residency matters, prefer a self hosted or customer controlled endpoints. For self hosted models assess public benchmarks not just on intelligence and capability but on instruction following, using benchmarks such as IFBench to ensure the model is geared toward correctly obeying instructions and not deviating.
3. Enable Cyana for a small group of experienced technicians. Experienced technicians are best placed to notice when Cyana is proposing something odd.
4. Leave the group's Cyana permissions at their supported defaults. What Cyana can actually do at this stage is still bounded by the per call approval gate.
5. Instruct technicians that during the initial period they should approve every proposed action rather than granting session level auto approve for any category. The friction at this stage is deliberate, as every approval prompt is a chance to notice that Cyana is proposing something the technician did not intend.
6. Ask technicians to start with lightweight read only workflows. Fleet health checks, low disk space finds, inventory summaries, service state investigations. These give the team a real feel for how the specific model chooses to investigate a problem in their environment, without Cyana changing anything on a machine.
7. Leave the "Review proposed tool actions with this endpoint" checkbox ticked on every configured endpoint so that the automated review is running from day one.

11.2 Expand As Trust In The Model Builds

Trust in an AI tool is real, it accumulates with observation, and it is specific to the model doing the work. As technicians see Cyana propose sensible tool calls, ask sensible clarifying questions and produce useful summaries, they build up an intuition for how the specific model behaves in their environment. That intuition is the signal for loosening what Cyana runs without a per call prompt. Expansion should follow that

confidence rather than a calendar.

- When a technician finds themselves approving many read only tool calls in a row for the same investigation, this is the moment to consider granting Search results or Read auto approve for the current chat. Do not persist the trust across chats. Grant it when it is useful and let it reset with the chat.
- When technicians report that Cyana handles Write, Command or Script scenarios well and the automated review has never had to catch a bad proposal, machine affecting categories start to be reasonable candidates for auto approve in specific chats. Even at this stage, machine affecting auto approve should only ever be per chat, and only after the technician has read the warning notice.
- Instruct technicians to always consider the environment they are working within. When they reach for the auto approve dialog, technicians should build a habit of considering the sensitivity of the environment, the blast radius of the machine set they are working with, and the tool use they are approving.
- Start Cyana usage in lower sensitivity environments until trust has accrued sufficiently in the model. Having Cyana take action on a newly installed VM is a fundamentally different position from having it run commands on a mission critical database server. In the latter case it will likely never be appropriate to have Cyana operate without full approval of all potentially damaging tool use. In the former Cyana can be given leeway to operate without the same concerns and the technician team can learn about the model by doing so.
- Broaden the admin envelope only after the operational envelope has been well exercised. If a group has been using "Read files and folders" and "View machine details and metrics" productively for weeks and has been blocked by the envelope in a workflow the technician genuinely wanted to run, that is when granting "Run commands and scripts" becomes a sensible next step, rather than granting everything at once at the start.
- **Trust in Cyana is trust in a specific model, in a specific environment.** If the organisation swaps its Cyana endpoint for a different model, treat the trust the team has built as partially reset. Model behaviour differs between providers and between generations, and Cyana's persisted schema does not currently record which model produced which decision. Start again in full approval mode on the new endpoint until the team has seen it behave well in their environment.

11.3 Higher Sensitivity Environments

Organisations that operate in regulated or high risk environments should tighten the baseline further:

- Prefer self hosted or customer controlled endpoints where data residency matters. SimpleHelp can provide advice on self hosting models.
- Use narrow scopes. Do not enable "All Machines" during a pilot.
- Require explicit approval for every meaningful tool call. Do not enable auto approve for any category during the pilot period.
- Avoid broad fleet actions until the team has validated Cyana's behaviour in narrower ones.
- Configure two review endpoints of the strongest available quality, and use different providers for the two so a systemic error in one provider does not silently propagate through the review layer.
- Review retained conversation and tool history as part of normal operational oversight, in the same

way session records are reviewed.

11.4 What To Observe

There are four signals worth attending to as a rollout matures. Two of them are directly observable in the SimpleHelp administrator surface. The other two rely on team feedback and normal audit review.

- **Red review overrides.** Every review outcome is recorded on the tool call proposal in the conversation transcript, and the technician's subsequent Approve or Deny is recorded in the same transcript. A team that finds itself frequently approving red rated actions should treat that as a signal to re-check the model's suitability for their workflows, review Cyana's system prompt, and consider tightening admin envelopes. There is no built in override rate metric today. This signal is observable by reviewing the AI history in the SimpleHelp history panel.
- **Blocked outbound activity log.** The administrator activity log receives an entry every time the outbound content policy blocks a tool call. That log is directly viewable in the administrator UI. An empty log is normal. A non empty log deserves a look to check whether the block corresponds to a genuine attempt to embed protected values, or to something that warrants further investigation.
- **Time saved on the workflows Cyana was intended to speed up.** This is qualitative and comes from the technicians. Ask them at review intervals.
- **Whether admin envelopes have been broadened by mistake.** Changes to Cyana permissions on technician groups flow through the standard SimpleHelp administrator activity log for group edits, and are auditable there.

---

12. Limitations And Admin Responsibilities

This section is included to prevent the paper from overstating what Cyana provides. Cyana's controls are carefully thought out, layered, and built to provide a high degree of safety, but there are boundaries that administrators should understand.

- **The untrusted tool result framing is a signal, not a guarantee.** Models that ignore the framing can still act on injected instructions inside a tool result. Cyana relies on the approval gate, on scope enforcement, on outbound data loss prevention and on automated review as its enforcement layers, not on the label alone.
- **The outbound content policy is precise, not exhaustive.** It scans arguments to outbound capable tools for known server configuration secrets, using exact substring match. It does not detect general secrets, customer credentials, credit card numbers, PII or content in tool results. Administrators who want broader outbound content controls should apply them at the network egress boundary.
- **Values shorter than twelve characters are not scanned.** The policy applies a length floor to avoid trivial false positives. Administrators who choose short server secrets weaken the policy for those specific values.
- **Web research is compartmentalised, not proven safe.** The sandboxed research agent has no tools other than `fetchWebPage`, no access to prior context and a canary check on its output. It is not a proof that a web page cannot influence the agent, only a demonstration that the web research agent has a very small blast radius if it does.
- **Automated review is a signal, not a decision maker.** Green and yellow ratings are advisory; only

red enforces manual approval. Administrators who want stricter behaviour should not disable review and should not encourage technicians to turn off auto approve overrides by ignoring red ratings.

- **Auto approve is bounded by the admin envelope and by the chat session; there is no server side lockout, by design.** A technician can grant Cyana auto approve for the categories offered in the dialog, after ticking the warning acknowledgement. That grant is scoped to the current chat and is cleared when the chat is cleared. It cannot exceed what the administrator has permitted for the group. Cyana does not expose a server side switch that turns off auto approve for a specific technician group, because doing so would mischaracterise the approval layer as an enforcement layer. For organisations working through a controls assessment such as SOC 2, ISO 27001, HITRUST or CIS Controls, the enforceable, auditable, centrally administered control in Cyana is the admin envelope, which in turn is the layer that maps to a "technically enforced" requirement: turning off "Modify files and folders" for a group makes it impossible for Cyana to write files for any technician in that group regardless of what happens in the chat, and the same holds for every other Cyana capability. Section 5.5 sets out the reasoning for why the approval layer is deliberately a judgment control rather than an enforcement control, and covers the compromised account case at the platform permission layer for the same structural reason.
- **The persisted transcript does not include a per turn record of which endpoint or model was used.** The endpoint is chosen per conversation. Organisations that need a per turn record should treat this as future work.
- **Mutual TLS and custom auth headers are not supported.** Endpoints that require mTLS or non standard headers need a reverse proxy.
- **Cyana In Session computer control is a granted, session bound capability.** It is off by default, only usable in In Session mode, only available if the group has the permission, and requires a confirmation dialog to enable. Administrators should still make sure technicians understand that once enabled, and until the chat is cleared, Cyana can move mouse and keyboard on the remote machine when in an active turn.
- **The per technician Cyana Command standing context is technician controlled and lacks a dedicated administrator surface.** A technician can save a free text context that is server persisted and prepended to every Cyana Command prompt they make. It is included in the persisted AI history transcript and in the automated review payload, so an administrator can inspect it through those routes, but there is no dedicated administrator UI for viewing, resetting or approving a technician's standing context today. Organisations that consider a technician account a sensitive credential should include the standing context in the same policy that governs technician account handling. Administrators do have the ability to specify endpoint wide context of their own via the endpoint administration UI, and as such can make models aware of company policies and priorities, including referencing specific contexts the model may find itself in (for example, acting as a reviewer).

---

13. Conclusion

AI powered RMM does not have to mean AI operated RMM. Cyana is designed on the assumption that the technician is in the driving seat and that the platform's role is to make that seat simultaneously powerful and safe.

The controls that make that possible are stated openly. Cyana cannot reach a machine that the technician cannot reach, cannot run a tool the group does not permit, and cannot run a tool call the technician has not approved unless the tool is one of a small set that is deliberately non destructive. Higher risk actions are reviewed by a separate agent whose only output is a rating and a comment, and a red rating from that reviewer forces explicit technician confirmation. Outbound tool calls have their arguments scanned for known protected server configuration values before dispatch, and detections both block the call and log to the administrator activity log. HTML reports and SVG images the model produces are sanitised before display, and SVG is rasterised to a static image rather than rendered in a browser. Web research runs in a separate isolated agent with no tools other than the single web fetch.

None of these controls depends on the model behaving well, and each is enforced on the SimpleHelp server and can be tightened independently by the administrator.

For teams that want the productivity of an AI assistant that understands their fleet, but who do not wish to hand over control of that fleet, Cyana is designed to make that trade off unnecessary.

Appendix A: Change History

- v1.0. Initial paper release, covering SimpleHelp 6.0 and 6.1 implementation.