How to Communicate with the Admin

Standing communication preferences captured by Cowork. Read once at the start of any session about this project.

Owner: Matt Evans (Admin) | Maintained by: Claude Cowork | First version: 2026-05-04

1. Why this document exists

Each new Cowork session about the StartupFinancialPlan.ai project starts fresh with no memory of prior conversations. Communication preferences captured in chat get lost when the session ends. This document is the single source of truth for HOW to communicate with the Admin — separate from WHAT to communicate (which lives in the SOP and Backlog Tracker).

Add a new entry whenever the Admin gives feedback that should not be forgotten. Each entry should name the preference, give the reason, and include a concrete before/after example.

2. Bottom Line Up Front (BLUF) — start at the top of the string

When reporting status, results, or analysis, lead with the headline / outcome / what-it-means. Do NOT bury the conclusion at the bottom of a long technical recap. The Admin is a CPA and PM-style reader — he wants the answer first, supporting detail second.

Source: Admin direction 2026-05-04 — "start at the top of the string, not at the bottom — this is how functional people see the world from a technical level."

Anti-example: "SA delivered four fixes. Fix 1 covered Apps 1-6 with sequential paste prompts; Fix 2 used a 30-second visibility flip; Fix 3 had a multi-step deletion sequence; Fix 4 added a nav button to the Hub. … In conclusion: the rework failed."

Correct: "Bottom line: the rework failed. Round 3 needed. Detail below."

3. Number every list item — never use bullets

When giving the Admin a list of things to do or check, use numbered items (1, 2, 3, …) rather than bullets. This lets the Admin reply by number — "1 PASS, 2 FAIL, 3 N/A" — instead of having to retype what each bullet was.

Source: Admin direction 2026-05-04 — "When tasking me, please number my task so I can refer back by number not bullet point."

Apply to: punch lists, verification checklists, options, multi-step instructions. Inside running prose, plain comma-separated lists are still fine.

4. Full path from https:// down to the line — never drop the Admin into mid-path

When asking the Admin to navigate to a code file, settings panel, deployed page, or any technical location, give the FULL path starting from where he is (typically a URL like https://… or a folder like C:\Users\…). Do not start with the deepest part of the path ("line 127 of pages/AgentsHub") and assume the Admin knows the upstream steps to get there.

Source: Admin direction 2026-05-04 — "When you reference a line number to a person who is functional and not a technical person, you must start with the upstream steps to get down to the line number + your path comes across as incomplete; people like me start with the beginning of the path such as https://"

Anti-example: "Code spot-check at pages/AgentsHub line 127 — confirm the handler binding."

Correct: "Open https://app.base44.com → workspace dashboard → click the Hub app → click 'Edit code' → file tree on left → expand 'pages' folder → open 'AgentsHub.jsx' → scroll to line 127. Confirm it shows: <nav onClick={handleNavClick}>."

If the Admin is not technical and likely cannot do the spot-check anyway, mark it Optional and don't ask.

5. Use file formats the Admin is familiar with — never .md (markdown)

When sending the Admin a file (response, report, log, deliverable of any kind), use formats the Admin opens in his normal Office workflow: .docx, .xlsx, .pdf. Never .md (markdown). Markdown is an AI/developer convention; professional users — especially CPAs and PMs working in Office — typically have no rendering tool for .md files and see the raw asterisks/hashes/pipes as visual noise. This rule applies to ALL AI agents on this project (Cowork AND Super Agent), not just to chat communication.

Source: Admin direction 2026-05-04 — "Communicate with files that the Admin is familiar with such as .doc, xls, etc. The AI Agents kept sending me .md documents (markdown) which most professional people never deal with." Earlier related: SOP v1.9 (2026-05-03) codified "do not post .md files" with Super Agent — promoting now to this communication log so the rule applies universally across all AI agents, not only SA, and so it survives across sessions.

Anti-example: SA's CO-001 round-2 response (2026-05-04) said: "Response doc saved at 09_Admin_Submissions/_Backlog/SA_Responses/CO-001_response_rework_2026-05-04.md" — referencing a .md path even after the SOP v1.9 rule. This is exactly the violation pattern: AI agents naturally default back to .md unless explicitly redirected each time.

Correct: All response files saved as .docx (matching the SA_Responses/CO-001_response_2026_05_04.docx pattern Admin actually uses). All standing logs as .docx. All trackers as .xlsx. All formal deliverables (reports, scope docs, plans) as .docx or .pdf. The only acceptable use of

.md is inside SA chat code blocks where SA is showing literal markdown syntax — never as a saved file extension.

Apply to: any saved-file deliverable. Cowork enforces this on its own outputs AND must include the rule in any SA prompt that asks SA to save a response or any other artifact. When drafting an SA prompt that involves file output, include explicitly: "Save as .docx (NOT .md)."

6. Top-down workflow instructions for functional (non-technical) users

When asking the Admin to navigate the Base44 platform (or any external tool), give instructions that start from the OUTERMOST starting point — typically the homepage URL — and drill DOWN step-by-step to the target destination. Do NOT start at the target ('open getSession in functions folder') and assume the Admin can navigate to it. Functional users execute the steps in the order given; if the first step doesn't match what they see, they get stuck. Each instruction should match the literal label the Admin sees in the UI, not the label you remember or expect.

Source: Admin direction 2026-05-04 — "I need directions from the top down — starting with Preview, Dashboard or Code which is my drill down workflow" and "Start at the top and work down when defining workflows for functional people." Triggered by 5 navigation hickups during CO-002 SA session: SA called the menu 'Backend Functions' but the actual label was 'Functions'; SA expected getSession in the Functions list but it wasn't there (because of a failed redeploy state); Admin couldn't find a 'New Function' button; the AI Agent Helper link resolved to a chat instead of an editor; and AI Agent Helper turned out to be under Superagents in the sidebar, not under Apps. Each hickup individually was small; cumulatively they caused Admin to park the CO.

Anti-example: "Open Backend Functions → getSession → click Deploy." — Skips the upstream navigation. If the Admin is on his app dashboard, he doesn't know which app to open, where Backend Functions is, or whether the menu is even called that. Three implicit assumptions, three potential hickups.

Correct: "Step 1: open https://app.base44.com in your browser. Step 2: in the left sidebar, click 'Superagents' (NOT 'Apps' — Superagents is a separate section, second item in the left nav). Step 3: click the three-dot menu (⋯) next to 'AI Agent Helper' → choose 'Edit'. Step 4: this opens the editor view. In the left sidebar of the editor, click 'Code' to expand. Step 5: under Code, click 'Functions' (the actual menu label may render slightly differently — adapt and report what you see). Step 6: in the Functions list, look for 'getSession.ts'. If present, click it. If absent, click the '+' button to add a new file and name it exactly 'getSession.ts'. Step 7: paste this code: [code block]. Step 8: click 'Deploy' (top-right corner). Step 9: confirm the success message appears. Tell me when done and I'll run a verification test."

Apply to: any instruction set asking the Admin to navigate a platform he doesn't fluently use. This includes Base44, GoDaddy, Stripe, any cloud console, any IDE, etc. Cover from URL bar to target click, with literal UI labels. Adapt-and-report mechanism: when you can't be sure of a literal UI label, tell the Admin 'the menu may be called X or Y or similar — tell me what you see and I'll adjust.' This converts a hickup-prone instruction into a self-correcting one.

Document hickups when they happen: Per Admin direction, every navigation hickup gets documented in the Action_Items 'Notes' column of the active CO AND surfaced in this log if it represents a generalizable pattern. The CO-002 hickups (5 documented) were captured in CO-002.04 in the Action_Items tab. This log Entry #6 captures the underlying RULE that prevents the pattern from recurring, separate from the CO-002 case study.

File-path corollary: The top-down rule applies to FILES the same way it applies to platform UI. When asking the Admin to open or view a file, give the FULL absolute path (e.g., C:\Users\mevan\OneDrive\Documents\Claude Project\09_Admin_Submissions_Backlog\SOP-001_Base44_Platform_Navigation.docx) OR provide a clickable computer:// link in the same message — preferably both. Never say 'open SOP-001' or 'view the tracker' without telling Admin where the file lives. If Admin uses File Explorer instead of clicking links, the absolute path is what he'll navigate. If Admin clicks links, the computer:// is what works. Both formats future-proof the instruction.

Source: Admin direction 2026-05-04 — "When you issue an action - open SOP, I have to start at the top - what folder or attach it to your communication to fool-proof your communication." Triggered when Cowork wrote 'Open SOP-001 v1.0' as a numbered instruction without specifying which folder it lived in.

Anti-example: "Open SOP-001 v1.0 in one window." — File name only; no path. Admin has to figure out where it lives.

Correct: "Open the file at C:\Users\mevan\OneDrive\Documents\Claude Project\09_Admin_Submissions_Backlog\SOP-001_Base44_Platform_Navigation.docx — or click this link: [Open SOP-001](computer://C:\Users\mevan\OneDrive\Documents\Claude Project\09_Admin_Submissions_Backlog\SOP-001_Base44_Platform_Navigation.docx). If the link doesn't work, navigate in File Explorer: OneDrive → Documents → Claude Project → 09_Admin_Submissions → _Backlog."

Apply to: every reference to a file Admin needs to open. Includes deliverables Cowork creates, source documents Admin authored, tracker files, logs, anything saved on disk. The clickable computer:// link is provided alongside, not as a substitute for, the literal path.

7. Use the Admin's product vocabulary, not implementation jargon

When describing the user-facing surface area of an app, use the words the Admin uses, not the implementation terms. For this project the standing vocabulary is:

1. **Survey** = the beta sign-up form (Full Name, Email, Role, Company Size, Company Type) that overlays the Hub on click
2. **App** = the actual app content loaded (e.g., the Step 1 LaunchCost UI, the Step 5 IncomeStatement Pro UI, etc.)
3. **Workspace** = the Workspace login page (Base44 platform-level access wall — "Welcome to LaunchCost / Sign in to continue")

Avoid: "modal", "intercept", "auth guard", "BetaGateModal", "DOM event delegation", "preventDefault", etc. — use them only when explicitly required (e.g., inside an SA prompt).

Source: Admin direction 2026-05-04 — "Did the survey load, did the app load or did the workspace load — this is how a dummy like me looks at the world."

If a fourth user-facing thing shows up that needs a name, ask the Admin what to call it BEFORE writing about it.

State translation (corollary): The vocabulary rule extends from UI elements to system STATE. When describing whether the suite is up/down/working/broken, describe what Admin can OBSERVE — login + app loads / login wall appears / nothing loads — not internal mechanisms (backend service status, function deployments, cache states, etc.). Admin doesn't need to know which backend service implements the suite's data layer; he needs to know whether he can use the apps.

Source: Admin direction 2026-05-04 — "Don't say getSession - just say login and see if you can access the app so we can see the suite is up and running." Captured during CO-030 setup after Cowork repeatedly used the technical service name in a verification instruction.

Anti-example: "Open any app to verify the getSession backend is online — if it errors, getSession may be offline." — Uses internal service name; tells Admin nothing about how to interpret success vs failure in his own terms.

Correct: "Open any app and try to log in. If the app loads with all its content, the suite is up. If the app errors, won't load, or shows broken forms, the suite is down." — Pure observable behavior. Admin doesn't need to know the implementation.

Apply to: all status verification, debugging, and "is this working?" questions. Translate technical-system-state to user-observable-state BEFORE writing the instruction. If the underlying mechanism truly matters (e.g., a specific bug requires checking a specific service), surface that ONLY after first describing the user-observable test.

8. How a status update should be structured

Combining the four preferences above, every status update / verification report / rework summary should follow this shape:

4. **Headline:** one sentence — what happened, what it means, what's next.
5. **Numbered punch list of action items:** written in product vocabulary (Survey/App/Workspace), with full paths where navigation is required, no jargon.
6. **Supporting detail (optional):** only if the Admin is likely to want to drill in. If skipped, leave a one-liner pointer ("detail in CO-XXX Notes column if needed").

9. Maintenance

Add an entry whenever the Admin gives feedback that meets all three of: (a) it's a HOW-to-communicate preference, not a WHAT-to-do request; (b) it should apply to all future communications,

not just the current one; (c) a future Cowork session would have no way to learn it without this document.

Each entry should: name the preference (heading), give a one-paragraph explanation, cite the Admin quote that triggered it ("Source: Admin direction YYYY-MM-DD"), and include a concrete anti-example + correct-example pair when the preference can be misapplied.

10. Revision history

v1.0 — 2026-05-04 — Initial version. Captured 4 preferences from CO-001 rework cycle: BLUF (top-of-string), numbered lists, full paths from https://, and Survey/App/Workspace product vocabulary. Source: Admin response to CO-001.08 verification.

v1.1 — 2026-05-04 — Added Entry #5: "Use file formats Admin is familiar with — never .md (markdown)." Triggered by Admin direction citing SA's repeated use of .md file paths in chat (most recent: CO-001 round-2 response). Promotes the SOP v1.9 rule from SA-only to universal across all AI agents on the project. Renumbered §5-§8 to §6-§9 to make room for Entry #5 in numerical order.

v1.2 — 2026-05-04 — Added Entry #6: "Top-down workflow instructions for functional (non-technical) users." Triggered by 5 navigation hickups during CO-002 SA session (Functions vs Backend Functions, missing function in list, missing New Function button, mis-resolving app link, Superagents vs Apps section). Renumbered §6-§9 to §7-§10 to make room for Entry #6 in numerical order.

v1.3 — 2026-05-04 — Extended Entry #7 (product vocabulary) with state-translation corollary. When describing system STATE, use observable behavior (login + app loads) not internal mechanisms (backend service status). Triggered by Admin direction during CO-030 setup.

v1.4 — 2026-05-04 — Extended Entry #6 (top-down workflow) with file-path corollary. When asking Admin to open or view a file, give the FULL absolute path AND a clickable computer:// link in the same message. Triggered by Admin direction during CO-030 SOP-001 v1.0 delivery.