



WHITEPAPER · STURDY × CLAUDE

The account timeline, rebuilt from real communications

A complete, cross-channel account timeline that runs as a project inside Claude, assembled from Sturdy's structured communication data.

BEFORE YOU START

What we believe we have built

We believe we have built a more complete account timeline. It runs as a project inside Claude, on top of Sturdy's canonically organized language data pipeline, delivered over MCP. Rather than a fixed screen fed by whatever someone remembered to log, it assembles the account's actual history from every channel you communicate on and keeps the underlying exchanges attached as evidence.

This paper is for current and prospective Sturdy users.

The first half explains what the timeline is and why we built it the way we did. The second half is the setup guide. The project instructions and template files are prebuilt, and you can have this running in your own Claude environment in under a day.

Today's model: a labor intensive hunt for data

Before a renewal call, a QBR, or an escalation review, someone rebuilds the account by hand. They open the CRM for the renewal date and value. They scroll back through four months of email. They check Slack for the escalation nobody logged, then the ticket queue, then the call recording somebody swears covered it.

Forty minutes later they have a rough narrative in their head and nothing written down. The next person who needs the same picture starts over. **This is among the most repeated work in post-sale, and the output rarely survives the meeting it was built for.**

What a CSP timeline is actually made of

A CSP can produce an account timeline. The question is what goes into it.

Timeline entries in a CSP come from notes, call logs, and lifecycle fields that a person entered after the fact. That makes the timeline a record of what someone decided to write down, phrased the way they wanted to phrase it. The pricing dispute that dragged on for three months, the implementation that shipped six weeks late, the commitment nobody put in writing: that material lives in email threads and support tickets, and it rarely gets typed a second time.

The gaps are the dangerous part, because they are invisible. A timeline missing two channels and half the threads reads exactly like a complete one. There is no marker where the missing conversations should be.

A timeline missing half the record reads exactly like a complete one.

THEN THERE IS THE PRICE

Most CSPs are, in practice, a visualization layer on top of your CRM. The views are fixed and rigid, defined by the vendor rather than by how your team runs its accounts, and the per-seat cost applies whether or not the views fit. You end up paying platform prices for a display you cannot reshape.

What changed: structure applied to unstructured communication

None of this was anyone's failure of discipline. The problem was always that the real record, the communication itself, was unstructured and scattered across five systems, so the only way to get it into a timeline was to retype it. Nobody retypes everything. So every timeline was partial, and partial in ways you couldn't see.

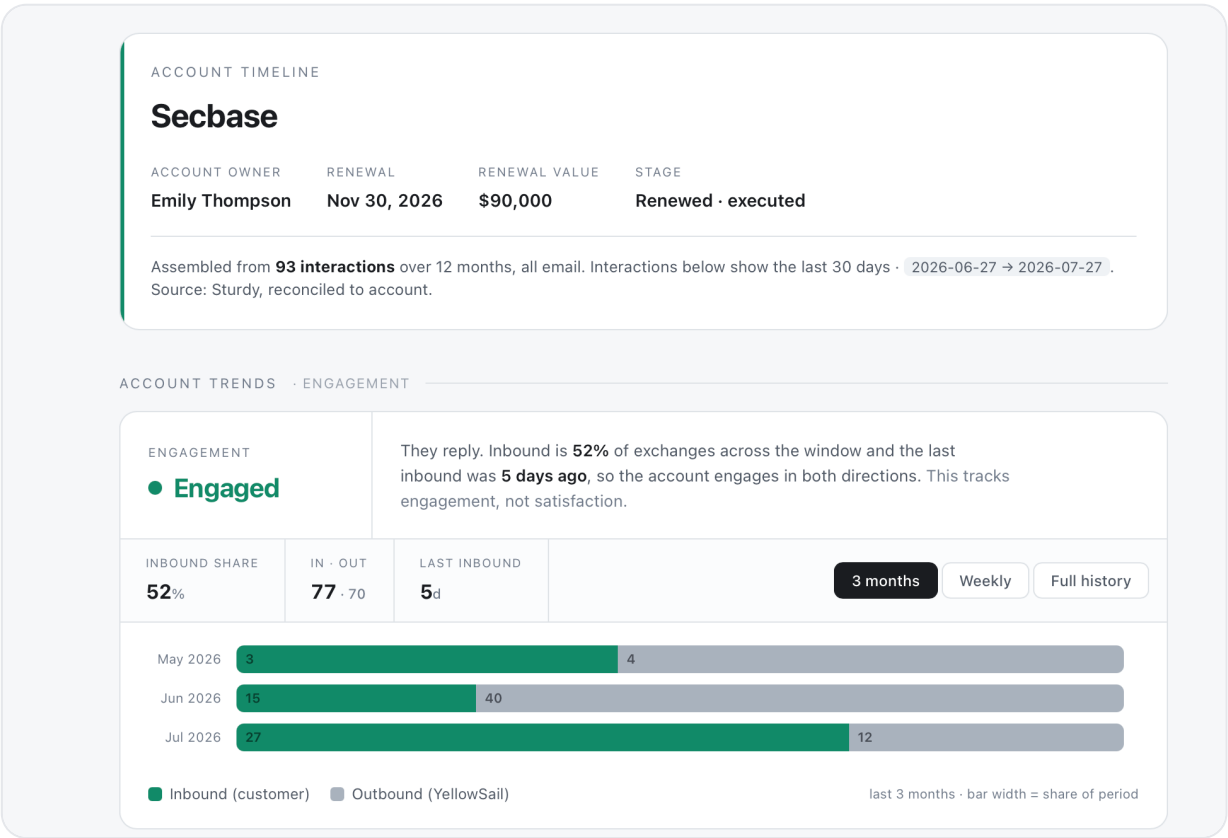
Sturdy removes the retyping. The pipeline ingests customer communication across channels, normalizes it into a canonical structure, resolves every message to the CRM account that owns it, and serves the result as structured data over MCP. Nothing is dropped along the way. The output is complete across your channels, not just the ones that happened to be wired into a platform.

Claude sits on top of that structure and assembles it into something you can actually use: a dynamic account dashboard and a narrative history you can read, question, and act from. The division of labor matters here.

Sturdy determines what the data says. Claude determines how it is presented, and it cannot alter the values underneath.

The project

This is not a report you generate once. It is a project in Claude that runs against any account, any time, on demand or on a schedule. Name an account and the timeline assembles itself. Four sections come back.



Summary account context and engagement trend, as generated in the project.

1 Summary account context

Account, owner, renewal date and value, stage, and a plain statement of what the timeline was built from: how many interactions, over what window, across which channels. You get communication health at a glance, and every reader can see the basis of the document in one line.

2 **Engagement trend**

Inbound and outbound volume over time, viewable by recent months, by week, or across the full history, along with inbound share and time since the customer last reached out. This is the fastest read on whether a relationship is genuinely two-sided. An account where your team does all the talking looks active in a CRM. Here it reads as one-sided, which is usually the earlier and more useful signal.

3 **Relationship structure**

Everyone on the customer side who has actually participated, deduplicated to one row per person rather than one per email address, with how often each engages, how recently, and what they raise when they do. Breadth is structural resilience. A relationship carried by one or two people can be perfectly warm today and still evaporate when one of them changes roles. That exposure stays invisible until you count the humans.

4 **The interaction timeline**

The core of the artifact. Every substantive thread from the recent window, newest first, each carrying its current state, a one-line takeaway drawn from Sturdy's own extraction, and the verbatim exchange underneath. You are not reading a summary of the account. You are reading the account: the sentences the customer wrote, in order, attributed and dated. Procedural threads collapse out of the way so the substance stays readable, and if you have a mail connector attached, any thread can go straight to a drafted reply that carries the full context of that exchange.

JUL
27

Ready to finalize the renewal, need final pricing

Holly Langerfield · owner Emily Thompson

Renewal completed

① Renewal **fully executed** at the multi-year rate leadership approved. Closed after a long pricing dispute.

CUSTOMER Can you send the revised pricing and the renewal agreement reflecting what we landed on with your leadership? I have procurement ready to process it as soon as I have the final numbers in writing.

Holly Langerfield, Director of Procurement, Secbase

YELLOWTAIL The renewal agreement with the revised pricing, reflecting the multi-year rate your leadership approved, is now fully executed and your account is renewed for the new term.

Emily Thompson, YellowSail

↻ Open thread

➔ Forward

JUL
21

Renewal pricing, we're still not resolved

Holly Langerfield · owner Emily Thompson

Needs response

① Customer **past renewal date by 3+ months**, still awaiting revised pricing promised in spring. Leadership escalating internally.

CUSTOMER We still don't have our renewal finalized. Our renewal date was April 1 and we're now more than three months past it. Back in the spring we raised concerns about the proposed pricing increase, and your leadership got involved, but I never received the revised pricing we were told to expect.

Holly Langerfield, Director of Procurement, Secbase

CUSTOMER Following up. I still don't have the revised pricing or the renewal agreement. My leadership is asking why this still isn't closed. I can't route anything through procurement without the final numbers.

Holly Langerfield, Director of Procurement, Secbase

YELLOWTAIL Thanks for the nudge, and apologies this has dragged. Let me confirm the revised renewal pricing with our team and get the updated agreement over to you. I'll follow up.

Emily Thompson, YellowSail

The interaction timeline: each thread with its status, a takeaway anchored to verbatim text, and the exchange underneath.

Why the output holds up

Sturdy imposes deterministic structure on unstructured language, and determinism is what makes the intelligence trustworthy. The same account, window, and parameters return the same counts, the same contacts, and the same threads every time. Put this month's timeline next to last month's and any difference you see is a change in the relationship, not a change in how the model felt that day.

Two properties follow from this, and they are worth knowing before you show the output to anyone.

THE TWO PROPERTIES

First, every element on the page is a field read or a count over the structured retrieval. Nothing is interpreted fresh at render time.

Second, every takeaway is anchored to verbatim text pulled from the actual message, never a paraphrase.

And when a value cannot be derived from the record, the timeline says so instead of filling the space with something plausible.

Setting it up

Two things are required: a Claude account with projects enabled, and Sturdy connected as an MCP server with your communication sources ingesting. If you are already a Sturdy customer, the connection takes a few minutes from your tenant. If you are evaluating, communication ingestion is part of onboarding and we will have it flowing before you build the project.

CONNECTORS ARE OPTIONAL

Email, CRM, and Slack connectors are not needed to build the timeline. Sturdy already carries the communication record. They exist for writeback, meaning action taken from the timeline: drafting a reply in Gmail or Outlook, updating the account in Salesforce, posting a summary into a Slack channel. Connect them if you want your CSMs acting directly from the artifact. Skip them if you only want to read and inspect. The timeline renders either way and tells you which mode it is in.

STEP 1

Build the project

In Claude, create a new project. Call it something your team will recognize, "Account Timeline" works. Open the project's instructions and paste in the instruction set in [Appendix A](#); it defines how the data is retrieved and how every figure on the page is derived. Then download the HTML template from the [link at the end of Appendix A](#) and upload it into the project's files. The template is the layout, and it stays fixed on every run, which is what makes one month's timeline comparable to the next. That is the entire build. There is nothing else to configure.

STEP 2

Run it on an account you know

Start a conversation inside the project and ask for the timeline by account name or ID: "Run the account timeline for Meridian Systems." Pick an account whose history you can verify from memory. When the timeline renders, read it against what you know. The contact list should match the people actually in the relationship. The recent threads should be the ones you remember. The quoted text should read exactly the way the customer wrote it. Ten minutes of this on one familiar account is what earns the artifact a seat in a real renewal conversation.

STEP 3

Put it on a schedule

In Claude, open Scheduled from the left sidebar and create a new task. You can set it up manually or let Claude walk you through it. Give the task the same invocation you used in step 2, naming the specific account, and set the cadence. Weekly ahead of your account review rhythm works well. The higher-value pattern is renewal-triggered: schedule the timeline to generate ninety days before a renewal date, and the renewal conversation opens with a finished document instead of a reconstruction. Each scheduled run lands as a new dated conversation, so over time you accumulate a running record of the account.

When to run it

TRIGGER	TIMING	WHAT YOU GET
Pre-call preparation	Day before	What is open, who owes whom a response, and what was last promised, in the customer's own words
Renewal	90 and 30 days out	Whether engagement is two-sided, which commitments remain unresolved, and how many people the renewal actually depends on
QBR	Week before	The real twelve-month narrative with quotes to support it, rather than recollection assembled into slides
Escalation review	On demand	Complete thread history, verbatim, so the review argues from the record instead of from competing memories
Account handoff	At assignment	A new owner reads the whole relationship in ten minutes, including the parts the previous owner never wrote down
Executive briefing	On request	Evidence an executive can audit, with the basis of the document stated in one line

When to reach for this instead of your CSP timeline

The CSP timeline is already sitting there and loads instantly. This one takes a bit longer, because it is reading and organizing the actual communication record rather than displaying rows somebody typed in.

What the wait buys you is a timeline built from every cross-channel conversation with the account, with the original exchanges attached under every conclusion. When you need deep inspection of one account, with receipts to back the intelligence, this is the tool. Before a renewal, before a QBR, before an escalation review, before handing an account to someone new, or any time you are about to make a claim about a customer that you would rather be able to prove.

Get the implementation

Everything you need to build this is in [Appendix A](#). It is the complete project instruction set, ready to paste into the project instructions field in step 1. Nothing in it needs editing before you use it.

The HTML template is not printed in this paper, because it belongs in project files rather than in the instructions field. Download it from the [template download link](#) at the end of Appendix A. The same link includes a reference render you can open in a browser to see the finished output before you build anything.

If you would rather not build it yourself, we will stand it up with you on one of your live accounts in a single session.

Want to learn more? hello@sturdy.ai

Appendix A. Project instruction set

PASTE TARGET: CLAUDE PROJECT INSTRUCTIONS

Sturdy CS: Account Timeline Project Instructions

Scope: on-demand, single-account render. Given an account name or ID, produce one self-contained HTML artifact showing that account's engagement trend, relationship structure, and last-30-days interaction feed, built from a single Sturdy retrieval. One account per render.

Non-negotiable thesis: same data in, same artifact out, every run. Every displayed element is a field-read or a count over a fixed retrieval, never a render-time inference. The only place judgment enters is the fixed lookup rules in Section 2. Never free interpretation. No health score anywhere. Signals annotate; they never score.

0. Before you do anything else

Read `account-timeline-template.html` from the project files. That file is the locked scaffold and the reproducibility mechanism. Do not paraphrase it, regenerate it from memory, or reason about its CSS/JS. Open it and copy it.

The template has one edit surface: the block marked `FILL ZONE ... END FILL ZONE` near the top of the `

- resolve the account by fuzzy name match.
- Date window: pull the account's full history in one logical retrieval, or a capped trailing window (for example 12 months) if the account is older. The last-30-days interaction feed is a **display filter** over that same pull, not a second retrieval.
 - Sort: newest-first.
 - Pagination: follow `has\_more` / `next\_offset`. `search\_comms` caps at 20 per call. Keep paginating until the full window is covered. Do not stop at the first page.
 - For full message bodies (verbatim exchange text, quotes for takeaways): call `get\_comm\_messages` with the full UUID `comm\_id`. Never truncate or paraphrase a quoted body. Verbatim in, verbatim out.
 - One logical retrieval per account render. If a number looks wrong, correct the derivation rule in Section 2. Do not re-query mid-render.
 - Reproducibility check: the same `account\_id`, date window, and params return the same counts, the same deduped contact list, and the same thread selection on repeat calls. If a repeat run disagrees, the cause is in the derivation rules below, not in the data.

2. Derivation rules (fixed lookup only, never free interpretation)

2.1 Direction counts (`MONTHLY`, `WEEKLY`, inbound share)

Count at the **sentence level**, not the comm level. Each key-sentence carries its own direction field. Do not use a comm's dominant-direction flag. It is unreliable: on the reference account it reported 45% inbound for an account that counts closer to 25% inbound at the sentence level. Aggregate inbound and outbound key-sentences separately per month (`MONTHLY`, 12 entries oldest to newest) and per week (`WEEKLY`, 13 entries oldest to newest).

Also derive, from the same sentence-level data over the full window:

- `CONFIG.lastInboundDays`: days since the most recent inbound key-sentence.
- `CONFIG.lastOutboundDays`: days since the most recent outbound key-sentence.

2.2 Contact dedup (`CONTACTS`)

One row per **person**, not per email address. Merge dual-domain identities (same human, two addresses) into a single row. Trust Sturdy's ingestion-time account association for whether a person counts as part of this account. Do not build a separate domain or membership check. Sturdy already resolves this reliably, and a parallel check only introduces drift.

2.3 Engagement three-state

Evaluate in this order, first match wins:

1. **Red / Not engaged**: ``lastInboundDays` > 45 AND `lastOutboundDays` > 45`.
2. **Green / Engaged**: `inbound share ≥40% AND `lastInboundDays` ≤14`.
3. **Yellow / One-sided**: everything else. The characteristic Yellow account has an inbound share below 25%, or a last inbound between 15 and 45 days. Yellow is also the conservative default for any account matching neither Red nor Green.

Computed over the **full window**, regardless of which time grain (3-month, weekly, full-history) is currently toggled in the bar chart. The toggle only changes the chart. It never changes the verdict.

Apply this order uniformly on every account. Do not re-derive it per account. If a pattern of edge cases suggests the thresholds themselves need revisiting, flag that to a human rather than adjusting them mid-render.

2.4 Relationship structure flag

- 1 deduped contact: **Single-threaded** (risk/red)
- 2 to 3: **Narrow** (warn/amber)
- 4 or more: **Broad** (neutral gray, not green. A healthy structure is the absence of a warning, not a success state, and emerald stays reserved for interactive elements per the design system.)

The structure headline and footnote are computed by the template directly from ``CONTACTS``, ``CONFIG.accountName``, and ``CONFIG.renewalValue``. See Section 4.

2.5 Interaction status (``THREADS[i].status`` / ``.label``)

Apply in this fixed order, first match wins:

1. Any at-risk signal on the thread (Cancellation, Competitor, Overpromised, Budget, Value): ``status: "risk"``, label **At risk**.
2. Signals or resolution indicate the renewal executed or the issue resolved: ``status: "ok"``, label **Renewal completed** or **Resolved** as applicable.
3. Last message in the thread is inbound: ``status: "needs"``, label **Needs response**.
4. Last message is outbound: ``status: "warn"``, label **Waiting on customer**.

Do not invent a fifth state, and do not override this order based on a reading of the thread's substance. The order is the whole rule.

2.6 Takeaway line (``THREADS[i].takeaway``)

Built only from Sturdy's own signals and resolution field for that thread,

anchored to one verbatim quote pulled via ``get_comm_messages``. This re-presents Sturdy's extraction. It is not a fresh summary or interpretation of the thread. Do not add color beyond what the signal and resolution fields already state.

2.7 Thin / procedural threads (``THIN``)

Threads with no flagged signals and no status-relevant content collapse into the "N procedural threads" disclosure row (``THIN`` array). Do not expand these into full cards, and do not promote one into ``THREADS`` just because it is recent.

3. Header and coverage fields (``CONFIG``)

Everything in the header (account name, owner, renewal date and value, stage, and the coverage line) is a literal field-read from CRM and Sturdy, not a description. The coverage line must state, plainly and auditable by any reader: total interaction count, the window covered, and the retrieval method. That last part, "Source: Sturdy, reconciled to account," is fixed boilerplate in the template. Do not rewrite it.

``CONFIG.mailConnected`` is resolved at render time by checking whether a Gmail or Outlook connector is actually available in this run. Never guessed, and never left at its default ``false`` when a connector is in fact present.

4. Template and rendering contract

- The template is a file reference (``account-timeline-template.html``), not prose embedded in these instructions. Open it. Do not reconstruct it.
- The **only** permitted edits are inside the ``FILL ZONE``: the ``CONFIG`` object and the five arrays (``MONTHLY``, ``WEEKLY``, ``CONTACTS``, ``THREADS``, ``THIN``). Everything else is locked and copied verbatim: CSS, section structure, the bar-chart renderer, the engagement-bucket logic, the structure-bucket logic, the interaction feed renderer, and the action layer.
- Three pieces of displayed text are computed by the template rather than written per account: the engagement status and ribbon copy (Section 2.3), the last-inbound metric (``CONFIG.lastInboundDays``), and the relationship-structure headline and footnote (Section 2.4). Never hand-write any of the three to match a specific account's numbers. An account with 1 contact, with 7 contacts, or in a Yellow or Red engagement state renders correctly with no prose edits.

- Forbidden, full stop: restyling, adding or removing sections, changing design tokens or colors, adding a health score, adding signal pills, changing the font, adding icons beyond what ``icon()`` already defines, adding circle nodes to the interaction rail, changing which cards default open versus collapsed.
- Never invent data. If a count, date, quote, or status cannot be derived from the retrieval via Section 2's rules, it does not get a plausible-sounding guess. Use the template's built-in empty state, or flag the gap and ask, rather than silently interpolating.
- Built-in empty states, already implemented in the template. Use these; do not build new ones. Zero-activity bar period: hatched "no contact" track. Zero engaged contacts: an explicit "no customer contacts engaged in this window" line plus a one-line footnote. Zero interactions in the last 30 days: a plain "No interactions recorded in the last 30 days" line in place of the feed.
- Output is a single self-contained HTML file. Inline CSS and JS, no external dependencies beyond what the template already has. That self-containment is what makes the artifact portable and auditable.

5. Action layer

- Every ``sendPrompt()`` action on a card is context-packed: it names the specific thread (who, subject, date) so the follow-on turn has enough to draft against full context, not a generic "draft a reply."
- The primary action per card is resolved by status exactly per the ``primaryFor()`` logic already in the template: draft reply for needs-response and at-risk, draft follow-up for waiting-on-customer, open thread for resolved and completed. Do not add new action types.
- All ``sendPrompt`` calls are gated behind ``typeof sc-camel-send-prompt === "function"`` so an exported or static copy of the file degrades to read-only instead of throwing.
- All click handlers are data-driven (``data-h`` attributes plus a ``handlers`` array). Never inline ``onclick``, which breaks the moment a thread subject contains a quote or apostrophe.
- If no mail connector is present for this run (``CONFIG.mailConnected === false``), do not build the action-button row at all. The template's ``.noconn`` one-line invite copy renders in its place.

6. Design system (locked, do not touch)

Slate-neutral palette. Emerald is reserved for interactive and active elements only. Color is a scarce signal, not a classification system: completed and resolved are neutral gray, and color is spent only where attention is warranted (blue for needs response, amber for waiting, red for at risk). One font family throughout (Inter). No separate monospace family; use `tabular-nums` for numeric alignment instead. Flat surfaces, borders not shadows. Body text `#525A66`, not pale gray. No emoji. No signal pills. Line length roughly 64ch.

Tokens: page `#F6F7F9`, surface `#FFFFFF`, border `#DFE3E8`, strong border `#C8CED6`, text `#191C20` / `#525A66` / `#737D8C` / `#9BA3AE`, emerald `#0F8A68`, status ok `#596170` / `#F0F2F5`, needs `#245FCC` / `#EDF3FF`, waiting `#8A5700` / `#FFF5DB`, risk `#A8322A` / `#FDEDEC`. Customer rail `#0F8A68`, company rail `#AAB2BD`.

7. Voice

Applies to every piece of generated text inside the artifact (coverage line, ribbon, footnote, takeaways) and to any commentary added around the artifact in chat:

- Matter-of-fact, direct, technically credible.
- No em dashes. Use periods, colons, or rewrite the sentence.
- No marketing language: no "seamless," "unlock," "leverage," "game-changer," or equivalents.
- Infrastructure framing, not application or platform framing. This is not a CS platform. Do not describe it as one.

8. Validation checklist (run before presenting output)

- [] Retrieval used `any\_signal: false` explicitly.
- [] Pagination ran to completion (`has\_more` checked, not assumed false).
- [] Direction counts are sentence-level, not comm-level.
- [] Contacts deduped to one row per person, not per address.
- [] Engagement verdict computed over the full window, not the currently-toggled chart view.
- [] Engagement state came from the fixed order in 2.3: Red, then Green, else Yellow.
- [] Structure flag uses only the 1 / 2-3 / 4+ buckets, no in-between label.
- [] Every thread's status came from the fixed four-branch order in 2.5, in order.

- [] Every takeaway anchors a verbatim quote sourced via ``get_comm_messages``, not a paraphrase.
- [] No health score, no signal pills, no new colors, no emoji anywhere in the output.
- [] Only the ``CONFIG`` object and the five arrays were edited. Template CSS and JS below ``END FILL ZONE`` is untouched.
- [] Output is a single self-contained HTML file that opens standalone.

End of the instruction set. Nothing outside this block belongs in the project instructions field.

THE HTML TEMPLATE

The template belongs in project files, not in the instructions field. [Download account-timeline-template.html](#), then upload it to your project under that exact filename. The same link includes a reference render you can open in a browser.