sturdy

WHITEPAPER · STURDY × CLAUDE

The success plan, rebuilt from real conversations

A renewal-readiness plan built from what your customers and your own team actually said, how they said it, and whether it moves you toward renewal. Nothing in it is self-reported.

BEFORE YOU START

What we believe we have built

We believe we have built a success plan that cannot go stale and cannot be gamed. Human input per account is about ten seconds. It runs as a project inside Claude, on top of Sturdy's canonically organized language data pipeline delivered over MCP, alongside your CRM. You type an account name. It returns a renewal-readiness assessment against your own criteria, a register of every promise made in both directions, the open loops you are sitting on, a map of who is carrying the relationship, and the specific exchanges behind each finding.

It is built entirely from real interactions with your customers: what was said, who said it, the tone they said it in, and how all of that maps to the conditions your organization has decided must be true before a renewal is safe. You write those conditions once. The plan measures every account against them.

There is no status field in it, no percent complete, nothing anyone can set to green.

This paper is for current and prospective Sturdy users, and for anyone who has watched a customer success platform's success plan module go unused since the platform was implemented.

> The first half explains why success plans fail, what we removed, and what replaced it. The second half is the setup guide. The project instructions and the render template are prebuilt, and you can have this running in your own Claude environment in under an hour.

01

How the plan actually gets updated

A CSM opens an account record the week before a QBR. The success plan is there, built during onboarding, four objectives with dates. Two are marked in progress. One is at sixty percent. The last status change was in March.

It is August.

She updates it, because the QBR deck exports from this object and it will look bad otherwise. She marks the first objective complete, because it basically is. She moves the sixty percent to eighty, because the project has clearly advanced. She pushes two dates. The plan now shows an account tracking well, and the export renders cleanly, and the meeting goes fine.

Nothing she entered was false, and she was not being careless. She reconstructed five months of an account from memory, in twenty minutes, under deadline, for a form whose only reader was a slide.

This is why they fail. **The success plan asks a human to observe everything that happened, translate it into a status field, and do so on a cadence nobody has time for.** That translation step is the whole product, and it runs under the precise conditions that guarantee it runs badly.

02

What a success plan is actually made of

Every customer success platform ships one, and they are structurally identical.

A plan record attaches to an account. It carries a type, an owner, dates, and a status. Below it sit objectives: the customer-side outcomes, each with a description, an owner, a due date, and a status. Below those sit tasks or CTAs, the internal work items. Task completion rolls up into objective percent-complete, which rolls up into plan percent-complete, which feeds a dashboard and often a health score.

Look closely at what flows through that structure.

Every status is typed by the person accountable for it being green. Not observed, not computed, not derived. A CSM under renewal pressure sets a dropdown, and the system has no mechanism to disagree. That is not a character flaw in CSMs; it is what happens when anyone grades their own work in a field their manager reads.

Percent-complete is the tell. Ask what the progress bar measures: the fraction of tasks someone marked done. That is data entry dressed up as progress, and the two are unrelated. A CSM can close every task on an objective the customer quietly abandoned, and the bar reads one hundred percent.

The objectives are usually the vendor's, not the customer's. "Mutually agreed success plan" is the sales language. What usually happens: a CSM writes plausible outcomes after a kickoff call, the customer

nods once in a meeting, and nobody on their side ever sees the record again. The plan describes what the vendor hoped would matter.

Nothing in it is the customer's own words. The pricing conversation that ran for three months, the sponsor who stopped replying in April, the security review that has blocked your integration since February, the sentence where their ops lead wrote "we are pushing this to next quarter": none of that is in the plan. It is in the email threads and the tickets and the call transcripts, and the plan never read any of them.

> **A success plan built from self-reported status is a record of what your team believed and had time to type. It is not a record of the account. Nothing in the artifact tells you how far apart those two things have drifted.**

THEN THERE IS THE PRICE

The success plan module is one of the anchor features that justifies per-seat customer success platform pricing. It is also, consistently, the least-adopted module in the deployment. You are paying for an object model whose central field is a dropdown, and paying again in CSM hours to keep the dropdown current.

03

Staleness is structural, not a discipline problem

The standard diagnosis of the unused success plan is a discipline problem: the team did not keep it up, so buy training, add a compliance report, tie it to a QBR gate.

That diagnosis is wrong, and it is worth understanding why before you replace anything.

Any artifact that needs manual maintenance to stay true decays at a rate proportional to how busy its maintainer is. That is arithmetic, not a behavioral observation. And a CSM's workload is not spread evenly across the book; it concentrates on the accounts in trouble.

So the plans that go stale fastest are the plans on the accounts that need them most. The quiet, healthy account has a beautifully maintained plan nobody needs to read. The account in trouble, sponsor gone quiet, integration blocked, renewal ninety days out, has a plan last touched in March. Its CSM has spent every hour since then working the account instead of describing the work in a form.

We should be clear about what we are and are not saying. We fundamentally believe that customer success professionals work hard, are engaged with their customers, and are actively working to deliver value, keep their portfolio happy, and drive retention. The problem is not effort or intent. In today's world, manual inputs and administrative tasks get glossed over in the rush to the next customer call, and that hurried record is unfortunately the one the business is run from.

A compliance report makes this worse. It converts the plan from a thinking tool into an obligation, and obligations under time pressure get satisfied at the lowest cost that passes inspection. Hence the week-before-QBR refresh: quarterly, retroactive, performative.

The second-order effect does real damage. A stale plan is worse than no plan, because it still feeds the roll-up. Leadership opens a dashboard showing eighty-one percent of objectives on track, built from records nobody has touched in a quarter. The number looks like management information. It is a summary of the team's last data-entry sprint.

None of this is fixable by trying harder. The only fix is removing the maintenance requirement.

04

What we refused to build

We did not build a status field.

Nowhere in this artifact does a human declare how something is going. No status picklist, no percent-complete, no health field, no last-reviewed date, no composite score. None of those fields is missing by oversight. Each was considered and rejected for the same reason: a field a human sets is a field a human sets under pressure.

We refused a single plan score for a related reason: a number gets optimized. Give a team a composite and within two quarters the composite is what gets managed, not the account. Per-criterion verdicts with evidence attached carry more information and resist gaming, because there is nothing to move except the underlying reality.

And we refused to author the objectives. This is the one judgment in the whole system that is genuinely not derivable, and pretending otherwise would be the most dangerous thing the artifact could do. A human decides what a safe renewal looks like. The system decides nothing about that, and says so on its own face.

Remove every field a person can set and what remains is, surprisingly, almost everything you wanted. The plan still tells you where the account stands, what is moving, what is stuck, what you owe them, who is in the room, and what to worry about. It just never asks anyone what they think.

05

What changed: the record became computable

None of this was anyone's failure of imagination. Success plans were built around self-reported status because status was the only thing available in structured form. The actual record, the communication itself, sat unstructured across five systems. You cannot compute on prose in mailboxes.

There is a human version of the same constraint. The exchanges carrying the real signal sit in the support queue, the sales inbox, other CSMs' books, and calls nobody indexed. Even a diligent CSM works from a fraction of what the account said, which is why the translation step in Section 01 loses so much. She was not summarizing the account. She was summarizing the part of it she happened to see.

Sturdy removes both constraints. The pipeline ingests customer communication across every channel, normalizes the language into a canonical structure, resolves every message to the CRM account that owns it, classifies it against your own signal taxonomy, and serves the result as structured data over MCP. Direction is carried at the sentence level, and resolution state is tracked and queryable.

The entity resolution is what makes a success plan possible instead of merely a search tool. A plan is an assertion about an account, and every message has to be attributable to that account with confidence before any assertion built on it means anything.

Once the record is computable, the questions a success plan always wanted to answer become countable:

- Whether anyone with budget authority has spoken in the last six weeks
- Whether every promise your team made in writing was kept, and how late the ones that were not are running
- Which threads are sitting with the customer waiting on you, and for how long
- How many distinct people on their side are still participating, and which of them have quietly stopped
- Whether the customer has described an outcome in their own words, unprompted, in the last quarter
- Whether anyone is discussing a next phase, or whether the conversation has flattened to maintenance
- Which of your contacts convert a request into action, and which absorb requests without reply

None of those need a form. All of them are already in the record.

> **Sturdy determines what the record says. Claude determines how it is read and presented, and it does not alter the values in the underlying communications data.**

06

The criteria

What varies between accounts is not what good looks like. It is whether this account has it. Safe renewals resemble each other. The sponsor is present. The vendor keeps its promises. The customer can articulate value in their own words. More than one person carries the relationship. Somebody is talking about the next thing. Those conditions hold across most of enterprise software, and writing them fresh for every account produces variation without producing insight.

So the criteria are standing. They ship as five defaults, they are yours to change, and they apply to every account in the book. Account-specific reality still surfaces in the evidence and the watch list, where it belongs, as a finding instead of a premise.

THE FIVE DEFAULT CRITERIA

Each criterion had to pass one test to earn a place: **can communications alone answer it?** A criterion that needs product telemetry, ROI figures, or survey scores fails no matter how good it looks in a playbook. The pipeline does not carry those things, and a criterion the system cannot evidence renders permanently gray.

1 · Economic sponsor is present. A contact with budget authority authored a message within forty-five days. This catches the account whose entire relationship runs through a practitioner who will not be in the room when the line item is defended.

2 · We are keeping our promises. No vendor commitment open past fourteen days, and no thread where the customer spoke last left unanswered past ten. This is the one no other system can run.

3 · Value stated in the customer's own words. A customer-authored message in the last ninety days describing an outcome for their own users or business, unprompted. This is the softest of the five and the most interpretive. We kept it anyway, because it is the closest derivable proxy for the question that decides renewals: a customer who has never articulated what you do for them cannot defend you internally when someone asks.

4 · More than one person carries the relationship. At least three distinct customer contacts authored a message in sixty days. This is not a weaker version of the first criterion. An account can have six engaged practitioners and no economic sponsor, or one attentive executive and nobody else, and those are different problems with different plays.

5 · A next thing is live. A named next use case, team, data source, or phase discussed in sixty days. Without it, the best available outcome is a flat renewal, and the account has no defense against a cheaper alternative arriving with a story about the future.

07

The commitment register

This is the component with no equivalent in any customer success platform.

A CSP records tasks, which are things your team assigned itself inside the tool. What it does not record, and structurally cannot, is the promise your solutions architect made in an email on the seventeenth: *we will have the field mapping over to you by Thursday.* That promise was made where work is actually agreed, and it never entered the system.

The register extracts those. Both directions, from full message text, with the message that made each one, the stated or inferred deadline, and whether later traffic shows commitments were kept or missed.

It also reaches past the account owner. Commitments get made wherever the customer is being talked to, which means support engineers, solutions architects, product managers, and executives on both sides, not only the CSM whose name is on the account. Those promises are the ones most likely to go unrecorded, because the person who made them does not live in the CRM and may never speak to the account team about it. The register reads every channel in the pipeline, so a promise made in a support thread carries the same weight as one made on a QBR follow-up.

Vendor-side commitments are listed first, overdue at the top. This is uncomfortable by design. Nobody self-reports their own misses, and no field anywhere in your stack records the four things your team said it would send and did not. Yet those four things are the substance of what a customer means by "hard to work with." Not one dramatic

failure. An accumulation of small unreturned promises, none worth escalating on its own, that together produce a renewal conversation with an edge nobody on the vendor side can account for.

It counts as a commitment when there is a statement of future action with an identifiable actor. Direct ones are easy. Vague ones still count, because "I will be back in touch shortly" is a promise the customer will remember whether or not the person who said it does. Conditional offers count once the condition is met. And when someone assigns a decision to the other party and no owner emerges, that becomes an **unowned decision**, tracked separately, because those items sit for months while each side believes the other has it.

It does not count as a commitment when the language is a pleasantry, a standing offer of availability, a hypothetical nobody took up, or scheduling with no deliverable attached. "Happy to help" is not a promise and treating it as one destroys the register's credibility on first read.

One rule carries more weight than the rest:

> **A commitment is never marked fulfilled on the promiser's own later assertion. You need the artifact, or the other party's acknowledgment.**

A vendor writing "as discussed, we sent that over" is not evidence that anything was sent. Without this rule the register quietly reverts to self-reported status, the thing this entire system exists to remove.

WHAT IT FOUND ON OUR OWN ACCOUNT

We ran this internally before we wrote about it. The first target was one of our own enterprise accounts, inside a renewal window, that we thought we knew well. The register's first run surfaced two open items that were on nobody's list, and both were ours.

The first was a document we had promised to fix. Our team had offered, in writing, to correct an inconsistency in a data scope document. The customer had explicitly asked for that document to be internally consistent before it went to their security review. Twenty days later the correction had never been sent, and the customer had already forwarded the document onward. Nobody did anything wrong at any single step. The offer was made in good faith on a Tuesday, it never closed, and the thread moved on to other things.

The second was a decision we had handed over without an owner. We had recommended that the customer decide whether to retain or purge a set of records. The recommendation was sound. No owner was ever named on either side, and the question sat open inside an active security review for three weeks while both parties assumed the other had it.

Neither item appeared in our CRM, and neither would have appeared in a status field. Both were sitting in threads our own people had read at the time and would have had to recall unprompted, weeks later, under deadline. That is the argument for the register in a sentence: we built it, we ran it against ourselves first, and it told us two things we did not know.

08

How the verdicts are computed

Each criterion is assessed against the matched communications for that account, and the assessment is deterministic. Judgment enters the prose and the evidence selection, never the verdict.

Evidence density is how much matched communication exists at all, graded thick, adequate, thin, or absent. Absent is a real and useful output: the criterion is unevidenced, which the reader needs to know and which no percent-complete could ever say. Alongside it runs an inbound share check, and below thirty percent the plan flags that the vendor is talking to itself. High activity with low inbound is not engagement; it is a CSM being busy in the direction of a customer who has left.

Momentum compares the trailing thirty days against the prior thirty, weighs in how long since anything moved, and returns accelerating, steady, decaying, or stalled. It replaces percent-complete, and it is honest where percent-complete is not: it reports whether the thing is moving, not how close it is to a finish line nobody can locate.

The friction ledger isolates threads where the customer spoke last and nothing came back, aged from the last inbound message, and separates them from blockers the customer owns. Those two situations look similar in a queue and call for opposite responses.

The coalition map counts who authored messages, not who was copied. It grades each contact active, decaying, gone dark, silent, or pending, and it flags three patterns specifically: a senior contact gone quiet while working-level traffic continues normally, which is the most reliable

renewal warning available; a relationship narrowing to a single person; and anyone named as the owner of an open item who has never written a message, which is where deliverables disappear.

It also reports roster size against active count. Sixty-eight contacts on record and six active in ninety days says more than either number alone, and it is a comparison most CRMs will happily hide from you.

Routing efficacy is the one nobody asks for and everybody uses. For each contact who has received a direct request, it computes the share answered within five business days. When one contact absorbs requests without reply while another consistently converts them, it names both and shows the arithmetic.

Language drift, which runs only in the comprehensive render, compares how the customer talks about the relationship now against earlier in the window. It is presented as a hypothesis with dated excerpts, never as a score, and when the excerpts do not show a clear arc the section is deleted rather than filled. Three cherry-picked quotes always look like a trend, which is why this one needs a stated discipline around it.

09

Why it cannot go stale

This is the structural answer to Section 03, and it is the property we are most confident about.

Nothing is stored. The plan is not a record that gets updated. It is computed at read time from the current state of the communications record, and then it is a file. Regenerate it next month and you get next month's account, not last month's plus whoever remembered to revise it.

There is no stale state because there is no state. The only persistent thing in the system is five sentences of criteria in a project instruction field, and those describe your renewal standards, not any account's condition. They are supposed to be stable.

It also means the plan cannot be gamed, in the specific sense that there is no surface to game. A CSM who wants the plan to look better has exactly one move available: improve the account.

And because the render is deterministic given the same record and window, this month's plan is directly comparable to last month's. Any difference between them is a change in the relationship, not a change in who had time to type.

> **A plan that is regenerated cannot rot. A plan that is maintained always will, fastest on the accounts where it matters most.**

10

Setting it up

Three things are required: a Claude account with projects enabled, Sturdy connected as an MCP server with your communication sources ingesting, and your CRM connected.

NO CUSTOM OBJECTS REQUIRED

The CRM connection uses standard objects only. Nothing is written back. The project is read-only by design; it reads account, opportunity, and contact records and renders a file. It will not create, update, or delete anything in your CRM, and the instruction set forbids it explicitly in two places.

STEP 1

Create the project

In Claude, create a new project. Name it something your team will recognize; "Account Success Plans" works.

Open the project's instructions and paste in the complete instruction set from Appendix A. It ships pre-configured with the five default criteria, so there is no setup interview and nothing to fill in before first use.

STEP 2

Add the render template

The render template is hosted at Success Plan Template. Save the page and add it to the project's files, or point the project at the URL directly.

The template does two jobs, and the second is easy to overlook. It is the render contract, carrying the token names, the repeatable blocks, and the design system, which is what makes every plan structurally identical and comparable month over month. It is also a worked example: the model reads it and sees the intended shape of the output, which is far more reliable than describing that shape in prose and hoping.

The template also carries a distinction worth knowing about before you open your first plan. The criteria you wrote are set in a serif typeface, the kind with small strokes on the ends of the letters, like the type used in a printed book. Everything the system computed is set in a sans-serif typeface, the plainer kind without those strokes, like the type on most websites. That is not decoration. It means you can tell at a glance which parts of the page a human asserted and which parts the system derived from the record, and it is why nobody can mistake an opinion for a finding.

Keep the style block unmodified. Everything else in the file is scaffolding the model fills in.

That is the entire build. There is nothing else to configure.

STEP 3

Run it on an account you know

Start a conversation in the project and type an account name. Nothing else.

The project asks which of two depths you want. Brief is the routine read and renders fastest. Full adds the evidence trail behind every claim plus the language-drift section, and is what you want before a

QBR or anything a customer will see. The prompt arrives before any data is pulled, so nothing is ever recomputed. If your team settles on one mode, changing `MODE_PROMPT` to `off` in the configuration block at the top of the project instructions turns the question off permanently.

Pick an account whose situation you can verify from memory. Read the output against what you already know. The criteria verdicts should match your instinct or disagree with it for a reason you can find in the evidence. The people in the coalition map should be the people actually in the relationship.

STEP 4

Do the exercise that actually validates it

Open the commitment register and check the vendor-side rows against your own memory.

This is a better test than checking the criteria, because the criteria will mostly confirm what you suspected. The register will tell you things you did not know. Look for a promise your team made in writing and did not keep. If it is real, you have just been told something no other system in your stack could have told you, and it took ten seconds of input.

A wrong row is equally useful, and it points somewhere specific. Over-extraction usually means a courtesy was read as a promise; the exclusion rules in the instruction set are where you tighten that. Under-extraction usually means a thread was not read in full, which the thread budget controls.

Ten minutes of that on one familiar account is what earns the artifact a seat in a real renewal conversation.

STEP 5

Tune the criteria

The criteria live in the project instructions, in the configuration block at the very top. To change them, open the project's instructions, edit the text between the two EDIT markers, and save. That is the whole mechanism. There is no admin console, no custom object, and no request to file with anyone. Any criterion can be reworded, retimed, cut, or replaced by the person who owns the project, and the change takes effect on the next run.

Run it on three or four accounts before changing anything. The defaults are conservative starting values.

When a criterion consistently reports something that disagrees with reality, the disagreement points at a specific line in that block: either the threshold in `good_looks_like` or the retrieval terms in `evidence_cues`. Both are plain text your team owns.

Evidence cues are the thing people get wrong, so it is worth being explicit. They are retrieval input, not description: concrete nouns, system names, role words, the phrases people type in real email. Abstract words like *value*, *adoption*, or *engagement* match everything and therefore nothing, and a criterion with vague cues reports thin evidence forever while looking like a data problem. If a criterion comes back consistently thin, look at its cues before you look at your pipeline.

Hold the line at six criteria. Each one is a card in the render, and thin evidence spread across many criteria is worse than solid evidence on a few. If someone wants a seventh, ask which one it replaces.

STEP 6

Put it on a schedule

Open Scheduled in Claude and create a task with the same invocation and a cadence.

The highest-value pattern is renewal-triggered: generate the plan ninety days before a renewal date, while a broken commitment is still fixable and a quiet sponsor can still be re-engaged. Monthly on strategic accounts is the other pattern worth running, because a deterministic render is built to be compared against itself, and each scheduled run lands as a dated conversation you can read as a sequence.

11

When to run it

TRIGGER	TIMING	WHAT YOU GET
Renewal preparation	90 and 30 days out	Whether the sponsor is still present, whether anyone has articulated value, what you owe them, and whether a next phase exists
Weekly book scan	Monday, brief mode	Which accounts have an open promise aging past the threshold, and where a sponsor has gone quiet since last week
QBR preparation	Week before, full mode	The account's actual state with the evidence attached, instead of a plan reconstructed the night before
Escalation review	When an account is raised	Whether the problem is theirs or yours. The commitment register and friction ledger answer that in one look
Book handoff	At assignment	A new owner learns what is open, who is engaged, and what their predecessor promised, without a knowledge-transfer meeting
Champion or sponsor change	When a key contact departs	What the relationship actually rests on now, and whether the account is effectively single-threaded
Executive account review	Monthly	A defensible basis for every claim, with an explicit statement of what the record could not see
Post-onboarding check	60 to 90 days after go-live	Whether promises made during the sale have been kept, which is where trust is won or lost early

12

The honest limits

It cannot see what was never written down. An agreement reached on a call and never confirmed in writing is invisible. This is the real ceiling on the system, and it is why the provenance footer names which channels were live and which were not. A thin record still yields a thin plan, and the artifact says so instead of filling the gap.

Commitment extraction is inference. It reads promises out of ordinary prose, and it will occasionally miss an implied one or over-read a courtesy. Ambiguous readings are marked ambiguous. Treat the register as a strong first pass that a human confirms, not a legal record.

Criteria are judgment-set. The five defaults are conservative starting values chosen for derivability, not constants fitted to churn outcomes in your business. They are legible and tunable, which is the difference from a platform's weighting, and they get sharper as you calibrate.

The value criterion is the softest. Whether a customer has articulated value in their own words takes more interpretation than counting who wrote a message. Read the evidence behind this one before you rely on it.

It does not decide what matters. A human writes the criteria. The system will not tell you whether your renewal standards are the right ones, and it says as much on its own face.

It is one account at a time. Portfolio ranking requires a wide extraction pass this project does not run.

13

Get the implementation

Everything you need is in Appendix A. It is the complete project instruction set, ready to paste into the project instructions field in Step 1, pre-configured with the five default criteria. Nothing in it needs editing before first use.

The HTML template is hosted at Success Plan Template, because it belongs in project files rather than in the instructions field. Open it 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, including the commitment register exercise in Step 4. That exercise tends to be the one that decides whether a team adopts it.

Want to learn more?

hello@sturdy.ai

Appendix A · Project instruction set

PASTE TARGET: CLAUDE PROJECT INSTRUCTIONS

Paste everything inside the block below into the project instructions field. It ships pre-configured with the five default criteria in the configuration block at the top; there is no setup step before first use. That block is also where you edit the criteria later. The render template is not reproduced here — it is hosted at Success Plan Template and belongs in project files.

PROJECT INSTRUCTIONS · PASTE IN FULL

```
# Derived Success Plan: Project Instructions

**Requires:** Sturdy connector (tenant instance) and Salesforce connector.
Read-only: nothing is created, updated, or deleted in any system.
**Knowledge file:** the render template, hosted at
https://successplantemp.netlify.app/

---

## CONFIGURATION BLOCK

<!— ========== EDIT BELOW THIS LINE ========== —>

```
STATUS: CONFIGURED
DEFAULT_MODE: brief
MODE_PROMPT: on

RENEWAL_CRITERIA:
- id: C1
```
```

```
  name: Economic sponsor is present
  good_looks_like: A contact with budget authority (VP or above, or someone
named as an approver) authored a message within the last 45 days.
  evidence_cues: VP, SVP, Chief, Head of, Director, sponsor, budget, approve,
sign off, procurement, leadership, steering, QBR
- id: C2
  name: We are keeping our promises
  good_looks_like: No vendor commitment open past 14 days, and no thread where
the customer spoke last has gone unanswered past 10.
  evidence_cues: I'll send, we'll get, we'll have, following up, any update,
still waiting, circling back, as promised, sorry for the delay, apologies for
- id: C3
  name: Value stated in the customer's own words
  good_looks_like: A customer-authored message in the last 90 days describes an
outcome for their own users or business, unprompted.
  evidence_cues: our team, our users, our reps, our customers, saved, reduced,
cut, faster, we're seeing, rolled out, went live, time savings, feedback from
- id: C4
  name: More than one person carries the relationship
  good_looks_like: At least three distinct customer contacts authored a message
in the last 60 days.
  evidence_cues: looping in, introduce, adding, my colleague, taking over, new
owner, handing off, cc'ing, wanted you to meet
- id: C5
  name: A next thing is live
  good_looks_like: A named next use case, team, data source, or phase has been
discussed in the last 60 days.
  evidence_cues: next phase, expand, additional, other teams, new use case,
roadmap, pilot, also interested, what would it take, scope

# OPTIONAL. Uncomment to add. C6 is inverted: finding nothing is the pass.
# - id: C6
#   name: No commercial or competitive threat language
#   polarity: absence_is_good
#   good_looks_like: No mention of alternatives, cost pressure, consolidation,
or contract renegotiation in the last 90 days.
#   evidence_cues: alternative, other vendors, evaluating, consolidate, budget
cut, pricing, discount, renegotiate, RFP, legal review, not renewing
# - id: C7
#   name: Support burden is falling
#   good_looks_like: Issue-related threads in the trailing 90 days are flat or
down against the prior 90.
#   evidence_cues: ticket, case, issue, bug, error, not working, broken,
degraded, outage, failing
```

These defaults ship working. The project runs on them immediately. No setup, no paste, just an account name. They are renewal-defense criteria, chosen because each one is answerable from communications alone. To change them, say *"customize criteria"* and follow Section 2.

This block holds only what cannot be derived. Vendor name, tenant, org identity, and every account fact are resolved from the connectors at runtime.

<!-- ========== EDIT ABOVE THIS LINE ========== -->

---

1. Startup check, before anything else

The project ships `CONFIGURED`. In the normal case there is nothing to set up.

If the message is an account name → go straight to Section 4 (mode), then Section 5 (execution). Beyond the mode prompt, ask nothing and confirm nothing. The only output is the finished HTML file.

If the user asks to customize criteria ("customize criteria", "change the criteria", "use our own"), run Section 2.

If `STATUS: NOT_CONFIGURED` (someone cleared the block) → run Section 2 before pulling any account data.

This gate is the entire performance design. A configured project has zero conversational turns between the account name and the artifact.

2. Customizing the criteria

Only runs when asked, or if the config block has been cleared. About two minutes.

Show the current defaults first so the user edits rather than starts from scratch. Most people want to change two of five, not write five.

Ask what they would change: *"Here are the five defaults. What would you cut, reword, or add?"*

Do not ask for their company name, tenant, or any account detail. All of it is derivable, and asking for a fact you can look up is the kind of friction this design exists to remove.

```
Hold the line at **six criteria**. Each one is a card in the render, and thin
evidence spread across many criteria is worse than solid evidence on a few. If
they want a seventh, ask which one it replaces.

Reject any criterion that communications cannot answer. Product usage, ROI
figures, and NPS scores are not in the pipeline. Say so plainly and offer the
nearest derivable proxy. "Value stated in the customer's own words" is the
derivable proxy for ROI; there is no proxy for usage.

For each criterion, **you draft** the `good_looks_like` line and the
`evidence_cues`. Do not ask the user for evidence cues. They will write the
criterion and skip the cues, and matching then degrades silently with nothing
in the output to announce it. Cues must be concrete retrieval terms: role
words, action words, system nouns. Generic words like *value*, *adoption*, or
*engagement* match everything and therefore nothing.

Output the completed block in a copyable code fence and say:

> Paste this over the configuration block in the project instructions,
replacing everything between the EDIT markers. That's the last setup step. From
here it's account names only.

Output the revised block in the same format as the shipped defaults, in a
copyable code fence.

**One honest note to give the user:** you cannot write this back yourself.
There is no tool that edits project instructions or project knowledge; both are
user-controlled. The paste is a one-time fifteen-second action, after which the
project never asks for input again.

---

## 3. Why criteria are standing, not per-account

Per-account objectives would require a confirmation turn on every new account,
which is the round-trip this design removes. Standing criteria are written once
and applied to every account, so an account name is sufficient input forever.

This also keeps the declared/derived line honest. A human wrote the criteria;
that judgment is not derivable and it renders in serif. Everything computed
against those criteria renders in sans. You never invent a criterion and never
present an unconfirmed judgment as declared.

Account-specific workstreams still surface, inside the criteria assessments and
the watch list, as derived findings, which is what they are.
```

```
---

## 4. Mode selection

On every run, ask which mode before making any tool call. Use
`ask_user_input_v0`: one question, two options, tappable.

**Ask first, run once.** The prompt costs one tap and precedes all data work,
so nothing is ever recomputed. Asking after the pull would waste the pull; not
asking at all risks rendering the wrong depth and forcing a full re-run, which
is far more expensive than the tap.

Question text, with the account name substituted:

> Which read for **{{ACCOUNT_NAME}}**?
> **Brief**: the routine scan. Criteria assessment, commitment register,
friction, coalition, watch list. Roughly 16 KB, fastest to render.
> **Full**: everything in brief, plus the evidence trail behind every claim and
a language-drift read. Roughly 26 KB. For QBRs, escalation reviews, and
anything shown to a customer.

Options: `Brief` and `Full`. List `DEFAULT_MODE` first.

**Skip the prompt in three cases:**

1. The user named a mode inline ("Northwind full", "brief read on Acme"). Never
ask for something they already told you.
2. They gave a standing instruction earlier in the conversation ("always full",
"stop asking"). Honor it for the rest of the session.
3. `MODE_PROMPT: off` in the configuration block. Then use `DEFAULT_MODE`
silently.

After calling `ask_user_input_v0`, the turn ends. Their selection arrives as
the next message. Do not keep writing, and do not begin the pull in the same
turn.

---

## 5. Execution on configured runs

Input is an account name. Output is an HTML file. Nothing in between.

### Call sequence
```

Run these without narrating them. No "let me pull the data", no interim findings, no summary before the artifact. Chat prose ahead of the file is pure latency.

0. Identity. `soqlQuery("SELECT Name FROM Organization LIMIT 1")` → this is `VENDOR_NAME`, authoritative, one cheap call. Never ask the user for it and never infer it from an email domain when this query is available. If it fails, fall back to the domain on `getUserInfo().identity.email`.

`TENANT_LABEL` is the name of the Sturdy connector you are calling. Read it from your own tool namespace (e.g. `Sturdy on Sturdy` → "Sturdy on Sturdy"). Do not ask.

Multiple Sturdy connectors. If more than one Sturdy tenant is connected, they hold different datasets and you must not merge them. Run step 1 against each. Use the tenant that resolves the account. If more than one resolves it, that is the second legitimate reason to ask a question: name the tenants and let the user pick. If none resolves it, say so and list the tenants you tried.

1. `search_accounts(search="<name>")` → Sturdy `id`, `sf.Account.Id`, owner, CSM. An ambiguous match within a tenant is the one other case where you may ask.

2. `soqlQuery`:
```sql
SELECT Id, Name, Amount, CloseDate, StageName, Probability, NextStep,
Owner.Name
FROM Opportunity WHERE AccountId='<sf_id>' AND IsClosed=false ORDER BY
CloseDate ASC
```
Take the nearest close date. No open opportunity → use the most recent closed-won for ARR and say so in the header. Standard fields only; do not call `getObjectSchema` on configured runs.

3. `get_available_signals()` → name-to-ID map. **Never hardcode signal IDs; they are tenant-specific.**

4. Counts, using `limit=1` and reading `total_comms`. Never paginate to count.
lifetime · trailing 30 (`beg_when`) · prior 30 (`beg_when`+`end_when`) · inbound-last trailing 30 (`last_item_direction="incoming"`)

5. `search_comms(account_ids=[id], beg_when=<today-90>, limit=20)` for the working set. One page unless `has_more` and the account is large; cap at two.

6. `search_comms(account_ids=[id], last_item_direction="incoming", resolution="unresolved", limit=15)` for friction.

7. `get_comm_messages` on **4 threads** brief, **6** full. Choose by `num_items`, recency, and critical or warning signals. Not optional: `key_sentences` cannot support commitment extraction, because promises live in ordinary prose that no signal fires on.

8. `search_people(account_ids=[id], person_type="contact", limit=50)` for the roster.

Target: 13–15 calls brief, 15–17 full. Step 0 adds one cheap SOQL call and removes a configuration field, a trade worth making, because a config field can be wrong, go stale, or be filled in differently by each person who copies the project.

Then map, compute, render

Map threads to criteria using `evidence_cues` plus `semantic_search_text` from `good_looks_like`. One criterion per thread, its dominant one. Threads matching nothing go to an unassigned pool; if that pool is large, say so in the render.

Compute the dimensions in Section 6, then emit the file. The artifact is the response.

---

6. Dimensions

Inverted criteria

A criterion carrying `polarity: absence_is_good` reverses the reading. C6 in the optional set is one: finding nothing is the pass.

- **Evidence `Absent` is a PASS**, not "unevidenced." Render it as clean, and say what was searched for so the reader knows the check ran.
- **Any match is the headline.** One matched thread on an inverted criterion outweighs a hundred on a normal one. Surface it in the card and consider it for the watch list.
- **Suppress momentum entirely.** Report match count and recency instead. A `Stalled` badge on an inverted criterion reads as failure when it means the opposite, and no reader will parse that correctly.
- **Never fire `inbound_flag`** on an inverted criterion. Low inbound share on a threat search is meaningless.

```
If you add a criterion where absence is the good outcome and forget the
polarity flag, the render will report a healthy account as failing every check.
Read the flag before scoring.

### Evidence density
Matched items per criterion, trailing 90. `Thick` ≥15 across ≥2 threads ·
`Adequate` 5–14 · `Thin` 1–4 · `Absent` 0.

On **Absent**, say plainly that no communications matched. An empty panel is a
valid finding: the criterion is unevidenced and the reader needs to know.

Inbound share of matched items, trailing 30. **Below 30%, fire the
`inbound_flag` block.** The vendor is talking to itself and volume is masking
disengagement.

### Momentum
Recency: `recent` ≤7d · `cooling` 8–20d · `cold` ≥21d
Volume (T30 vs P30): `up` ≥+25% · `flat` ±25% · `down` ≤-25%

| State | Rule | Class |
|---|---|---|
| Accelerating | recent + up | `mom-accel` |
| Steady | recent + flat | `mom-steady` |
| Decaying | cooling, or recent + down | `mom-decay` |
| Stalled | cold, or zero in T30 | `mom-stall` |

Apply mechanically. Where the state understates something (a milestone cleared,
the ball legitimately with the customer), put that in the prose, not the state.
A classifier a human can argue out of a bad state always returns green.

Below ~10 matched items per criterion per quarter, omit the state and report
density and recency instead. At that volume a 25% delta is one message.

Never render a percent-complete or a composite score. Both get optimized.

### Commitment register
Explicit promises, both directions, from full message text.

**Include:** direct ("I'll send it Thursday") · vague but real ("I'm pulling
that together", "back in touch shortly") · conditional offers whose condition
has since been met · assigned decisions, which become **unowned decisions** if
nobody claims them.

**Exclude:** pleasantries and standing availability ("happy to help", "let us
know") · hypotheticals never taken up · pure scheduling with no deliverable
```

```
attached.

**Soft due dates:** explicit date → that date · "tonight"/"by morning" → next
business day · "shortly"/"soon"/"this week" → +5 business days · nothing stated
→ Open at 14 days, escalate at 30.

**Never mark Met on the promiser's own later assertion.** A vendor saying "as
discussed, we sent that over" is not evidence. You need the artifact or the
other party's acknowledgment. A self-report-tolerant register is just a task
list.

**Vendor-side first, overdue at top.** Cap at 8 rows brief, 12 full. Overdue
and open always survive the cap; fulfilled rows are what you drop. If a vendor
commitment has been open weeks while the customer already took a downstream
action assuming it was done, that is a watch-list headline, not a table row.

### Friction ledger
Unresolved, customer last to speak, aged from last inbound. `Fresh` ≤3d ·
`Aging` 4–10d · `Overdue` 11–20d · `Abandoned` ≥21d. Track customer-owned
blockers separately and label the distinction, because the two call for
opposite responses.

### Coalition map
Days since last **authored** message. CC is not participation.

`Active` ≤14d `st-active` · `Decaying` 15–45d `st-decay` · `Gone dark` ≥46d
`st-dark` · `Silent` (CC only, never authored) `st-dark` · `Pending` (named,
zero comms) `st-dark`

Flag: **sponsor decay** (senior contact dark while working-level traffic
continues, the most reliable renewal warning there is) · **coalition
narrowing** · **assigned but silent** (named owner of an open item who has
never authored a message, which is where deliverables disappear).

**Champion departure** is a discrete event, not a standing criterion, so it is
not in the criteria set. Detect it here and send it straight to the watch list:
an `Executive Change` signal, a farewell message, a new name introduced as
taking over, or a bounced address on a previously active contact. On an account
inside 120 days of renewal, treat it as the top watch item regardless of what
the criteria say.

Cap at 6 rows; group the tail. Report roster size against active count.

### Routing efficacy
Per contact receiving direct asks, share answered within 5 business days.
```

```
Report at ≥2 asks. Emit a **reroute recommendation** when one contact has ≥2
asks and 0 responses while another has ≥2 and ≥2. Name both, cite the asks,
state it numerically. No CRM field records this.

### Language drift, full mode only
Two or three dated excerpts showing an arc in how the customer talks. Toward
outcomes = adoption. Toward cost, alternatives, or internal justification = the
earliest renewal signal available. Toward gatekeeping = security or procurement
posture, not necessarily unhappiness.

**Hypothesis with quotes, never a score. If the excerpts don't clearly show an
arc, delete the whole block.** Three cherry-picked quotes always look like a
trend.

---

## 7. Render

Use the render template at https://successplantemp.netlify.app/. Substitute
tokens, expand repeats, delete inapplicable optional blocks, preserve the CSS
byte-for-byte.

**Typography encodes the architecture.** Serif = declared (the criteria a human
wrote). Sans and mono = derived. Never put derived content in serif or declared
content in sans; it breaks the one visual argument the artifact makes.

Set `DECLARED_SOURCE` to `Standing criteria · project config`.

### Modes

| | Brief (default) | Full |
|---|---|---|
| Evidence disclosures | omitted | 4 per criterion |
| Language drift | omitted | included if clear |
| Narrative | 3 paragraphs | 4–5 |
| Commitments | 8 rows | 12 |
| Watch items | 3 | 5 |
| Coalition rows | 6 | 8 |
| Typical size | ~16 KB | ~26 KB |

Mode comes from the Section 4 prompt. `DEFAULT_MODE` is listed first in that
prompt, and is used silently when `MODE_PROMPT: off` or when the user names a
mode inline.

### Volume discipline
```

```
The render is the slowest part of a run and length is the cost. The caps above
are ceilings, not targets. Prose paragraphs run three to four sentences.
Evidence paraphrases are one sentence. Never pad a section to look complete. A
short section with four cited items beats a long one with filler, and it
arrives sooner.

---

## 8. Honesty

State what wasn't covered: transcripts not ingested, Slack not enabled,
attachments unparsed, telemetry outside the pipeline. The provenance footer is
not boilerplate. A reader who doesn't know the corpus boundary can't calibrate
the plan.

Report tool calls, corpus size, date range.

Where evidence is thin, say so in place. "Four items matched this criterion in
ninety days" is a finding.

Mark ambiguous commitment readings as ambiguous. An over-confident register
loses trust on the first false positive a reader spots.

Never claim the plan replaces the customer conversation or that it decides what
matters.

---

## 9. Anti-patterns

**Never write to any system.** No `createSobjectRecord`, `updateSobjectRecord`,
or `deleteSobjectRecord`, even if asked. Offer the HTML instead.

**Ask exactly three things, and only these:** the mode prompt in Section 4, a
genuinely ambiguous account match, and an account that resolves in more than
one connected Sturdy tenant. Everything else is derived. No confirmations, no
"shall I proceed", no summarizing your plan back before executing it.

**Never ask for a fact you can derive.** Vendor name, organization, tenant,
CSM, owner, ARR, renewal date, contact roles, and every account attribute come
from the connectors. If you are about to ask the user for something a query
would answer, run the query instead.

**Never narrate the pipeline.** The file is the answer.
```

```
Do not hardcode signal IDs. Do not paginate to count. Do not extract
commitments from `key_sentences`. Do not call `getObjectSchema` on configured
runs. Do not treat CC as participation. Do not render a health score, composite
score, or percent-complete.

Do not reproduce credentials, passwords, tokens, or passcodes found in message
bodies, even when germane. Reference the artifact, omit the secret.

Do not smooth over vendor-side failures. The register's value is proportional
to its willingness to be unflattering.

Do not paste message text. Paraphrase. Quote only where exact wording carries
weight, under fifteen words, one quote per source thread.

---

## Appendix: Template contract

74 tokens, 13 repeat blocks, 3 optional blocks.

**Repeats:** `criterion` · `narrative_para` · `criterion_card` · `evidence` ·
`vendor_commitment` · `customer_commitment` · `unowned_decision` · `friction` ·
`person` · `drift_excerpt` · `watch_item` · `tool_call` · `gap`

**Optional:** `inbound_flag` (inbound share <30%) · `unowned_decisions` (a
decision nobody claimed) · `drift` (full mode, clear arc only)

**Class values you set, not invent:**

| Token | Values |
|---|---|
| `MOMENTUM_CLASS` | `mom-accel` `mom-steady` `mom-decay` `mom-stall` |
| `STATE_CLASS` commitments/friction | `s-open` `s-prog` `s-met` |
| `STATE_CLASS` coalition | `st-active` `st-decay` `st-dark` |
| `ROW_CLASS` | `flag` for overdue/abandoned/unowned, else empty |
| `BALL_COLOR` | `var(--owed)` vendor owes · `var(--met)` customer owes ·
`var(--derived)` shared |

Citations in prose: `<span class="cite">THREAD·DATE</span>`.

Sections marked `[FULL ONLY]` are deleted in brief mode.
```

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

Appendix B · What the render contains

The template — hosted at Success Plan Template — produces one self-contained HTML file per run. Section order, top to bottom:

01 **Account header.** ARR, renewal date, days remaining, stage, last contact, generation timestamp

02 **Renewal criteria.** The conditions you wrote, shown as typed

03 **What's actually true.** Derived narrative, every clause cited

04 **Criteria assessment.** One card per criterion with momentum, evidence density, last movement, and which side holds the ball

05 **Commitment register.** Vendor-side first, overdue at top, unowned decisions in their own band

06 **Friction ledger.** Open loops aged from last inbound

07 **Coalition map.** Participation states, roster size against active count

08 **Language drift.** Comprehensive mode only, omitted when the arc is not clear

09 **Watch list.** Derived risks, each with the evidence that triggered it

10 **Provenance.** Corpus size, date range, entity resolution IDs, tool calls made, and an explicit statement of what the record could not see

Two render depths. Brief omits the evidence disclosures and the drift section and is the routine read. The comprehensive render includes both and is what you want in front of a customer.