Don't Start From Scratch

How HR can use its own history to put AI to work. The prompts and checklists from the From Day One webinar.

Clayton Holz · Chief Product Officer, Wisq · clayton@wisq.com

Your HR inbox already records how the work really gets done: what people ask, who answers, where things stall, and which rules nobody wrote down. These prompts turn a year of that history into answers to five questions:

1. **Where do we start?** Rank the work by effort, not volume, then pick what to build first. (P1a–P1d)
2. **How does the work actually happen?** One label can hide several processes. (P2)
3. **Where are we missing policy?** The rules that live only in HR's replies, and which to fix first. (P3, P3m, P3p)
4. **When should a person take over?** Situations, not keywords. (P4)
5. **What do we need to connect?** Specific reads and writes to bring to IT, and what you can do without them. (P5, P5b)

Before any of that, **Step 0** makes your export legible (P0, P0b). The last section covers what to do with what you find.

Practice on Halverline first. This folder includes a fictional company's inbox and handbook. Run the prompts on them until you trust the method. Then decide with privacy, legal, and IT whether to run them on your own data.

What's in this folder

File	What it is
This guide	Setup, every prompt, a checklist for checking each result, and what to do next
halverline_hr_inbox_export.csv	12 months of three fictional HR shared inboxes: 880 conversations, 4,524 emails
Halverline_Employee_Handbook.pdf	The fictional company's handbook, for Questions 1 and 3

About Halverline. Halverline Distribution is a fictional company with 1,250 employees, six distribution centers and a headquarters; about 70% are hourly. The export covers three shared inboxes: People Ops, Benefits & Leave, and Employee Relations. It's messy on purpose, like a real export: forwards and quoted replies, vendor notices, emails sent from phones at 5am,

personal email addresses, and a few messages in Spanish. Every person, email address, and phone number is invented.

Set up Claude (10 minutes)

1. **Use your company's approved account,** never a personal one. On Claude's Team and Enterprise plans, your data isn't used to train models by default.
2. **Turn on code execution.** In Settings › Capabilities, turn on *Code execution and file creation* (it may be called *Create and edit files*). If you don't see it, your admin controls it for your organization. Claude needs it to count across the whole file instead of estimating. Without it, the export is too large to upload.
3. **Use a regular chat, not an incognito one.** Incognito chats can't run code.
4. **Give it time.** Each prompt takes a few minutes while Claude works through the file in batches. Let it finish.

How to run the prompts

- **Make it legible first.** Run P0b once on the raw export. Claude saves a clean file called threads.csv. Download it and use it in every later chat instead of the raw export.
- **Start a new chat for each question, and carry your files forward.** Long analyses fill up a chat. In each new chat, upload your files, paste the ground rules (P0), then paste the prompt.
- **Download the files Claude saves,** especially threads.csv, the classification CSV from P1b, and the findings from each P3 run. If Claude saves findings as a document instead of a file, download it or ask for a CSV.
- **Replace anything in [brackets]** with your own category, inbox, or workflow.
- **Check every result before you believe it.** Each prompt has a short checklist. Every finding cites conversation IDs, so open a few and read them yourself.
- **If Claude stops early,** say: *"Continue until every conversation is covered, then give me the full result."*
- **Your numbers will differ slightly from ours,** even on the same data. The patterns shouldn't.

Which chat, which files

Chat	Prompts	Upload
1	P0, then P0b	The raw export
2	P0, then P1a, P1b, P1c, P1d	threads.csv and the handbook
3	P0, then P2, P5, P5b	threads.csv and the classification CSV from P1b
4	P0, then P3	threads.csv and the handbook. For a big export, use one chat per mailbox
5	P0, then P3m and P3p	The findings file from each P3 chat. With only one P3 chat, run P3p there and skip P3m
6	P0, then P4	threads.csv. For a big export, split it the same way as P3 and merge with P3m

P3 and P4 read every thread in full, which is more text than one chat holds for a large export. In our run, P3 took three chats, one per mailbox, plus a merge.

Before you use your own data

- [ ] **Privacy, legal, and IT have said yes.** You're moving employee records into a new tool.
- [ ] **You're using your company's approved AI account,** never a personal one.
- [ ] **You've practiced on the Halverline data first.**
- [ ] **You're starting with lower-risk mail:** People Ops before Benefits & Leave or Employee Relations, which carry medical and investigation details.
- [ ] **The export covers 12 months or more, and every message.** No samples.
- [ ] **P0b is the first thing you run.** It replaces names and email addresses with IDs and removes SSNs, bank and card numbers, and medical details before any analysis. If your privacy team wants that done before anything leaves the mail system, ask IT to do it in the export instead.
- [ ] **You know when you'll delete the raw export.** Keep the cleaned file and the findings, not the raw mail.

Getting your export

Once you have a yes, try to get the export yourself first, and ask IT only if you need to.

- **Gmail:** Google Takeout exports a mailbox as an MBOX file, which works as is.
- **Outlook:** the built-in CSV export leaves out the date each email was sent, which breaks the timing analysis. Ask IT for an eDiscovery export of the individual messages instead. Ask IT for shared or delegated mailboxes too.
- **Ticketing tool:** export the description and every reply or comment, not just the summary field.
- **Any format works.** A CSV, mail files, a spreadsheet, or a ticket dump: P0b turns it into one clean table. Don't clean it by hand.
- **Files can be up to 30MB each.** If your export is larger, split it by month or by mailbox.

Step 0 · Make it legible

P0 · Ground rules (paste first, in every new chat)

> I'm going to analyze an export of our HR shared inboxes to figure out where AI could take on real work. Ground rules for this whole conversation:
> 1. Use the complete file. Never sample unless I ask you to. If something is too big to do in one pass, tell me and work in batches until every row is covered.
> 2. Every finding needs evidence: cite the conversation IDs so I can check it myself.
> 3. Tell me when you're inferring rather than reading something directly.
> 4. Don't repeat personal details (names, health information, ID numbers, pay rates) in your answers. Refer to conversations by ID.
> 5. Ignore quoted earlier messages (text below a "From: ... Sent: ..." block) so you don't double-count.

Why it matters: rules 1 and 2 make the output checkable. Rule 4 keeps sensitive details out of your chat history and anything you share.

P0b · Make any export legible

You'll get: one clean table of conversations, whatever you started with: a mailbox export, a ticket dump, a spreadsheet, or chat logs. Run it once; every later prompt uses its output.

> I've uploaded an export from our [HR shared inbox / ticketing tool / other source]. Before any analysis, use code to turn it into one clean, legible table. Don't summarize anything yet.
> 1. Work out the format (CSV, spreadsheet, MBOX or EML mail files, ticket export, chat log) and tell me what each column or field holds.
> 2. Make one row per message with these columns: conversation_id, message_id, sent_at, mailbox_or_queue, sender_id, sender_role (employee, HR, manager, outside party, or automated system), recipient_ids (everyone it was sent or copied to), subject, body. If there's no conversation ID, group messages into threads by normalized subject line and participants, and tell me how you did it.
> 3. Clean each body: remove quoted earlier messages, signatures, confidentiality footers, and "external sender" banners, so each row holds only what that person newly wrote.
> 4. Protect privacy: replace every person's name and email address with a stable ID (the same person always gets the same ID), and remove anything that looks like an SSN, a bank or card number, or a specific medical detail (a diagnosis, condition, or medication; general words like "surgery" can stay). Then search the cleaned text again for names or email addresses you missed, fix them, and tell me how many you found. Keep the lookup table out of your answers.
> 5. Check the data: find and remove duplicate messages, report the date range, the number of months covered, and how many messages have no date, and fix text that broke across rows.
> 6. Flag automated notices (vendor alerts, system notifications, out-of-office replies) so later steps can set them aside, and mark the ones that still ask HR to do something.
> Save the result as threads.csv. Show me the row and conversation counts before and after, what you removed and why, the date range, and five sample conversations so I can check them.

Check:

- [ ] The conversation count looks right for your inbox.
- [ ] The date range covers at least 12 months, and few messages are missing a date.
- [ ] The five samples read like real conversations, with nothing quoted twice.
- [ ] No names or email addresses are left in the bodies. In our run, the re-check caught 35 names the first pass missed.
- [ ] SSNs, bank and card numbers, and medical details are gone.
- [ ] You downloaded threads.csv. On your own data, delete the raw export once you have it.

Why 12 months or more: HR months are seasonal and noisy. In Halverline's year, six different categories were the busiest of the month at some point, a third of benefits enrollment questions came in November, and accommodation requests didn't appear at all in 5 of the 12 months. One month, or a sample, shows you the wrong work.

Question 1 · Where do we start?

Run P1a to P1d in one chat. Upload threads.csv and the handbook.

P1a · Profile the export

You'll get: the basic shape of your inbox in a few minutes.

> I've uploaded threads.csv from P0b (one row per message). Use code to profile it:
> - the date range
> - emails and conversations per mailbox
> - who starts conversations (employees, managers, outside parties, automated systems)
> - the share of employee emails sent outside 8am–6pm Monday–Friday
> - the median time to the first reply from HR, and how many conversations started by a person never got one
> - the distribution of emails per conversation
>
> Use the sender_role column to tell HR from employees. If you didn't run P0b: treat the shared mailboxes and the HR, payroll, and IT staff who answer them (identify them from their signatures) as HR, and everyone else at our company, plus anyone writing from a personal email address, as employees. Either way, list who counts as HR before you calculate so I can check it.
> Show one short table and the three findings you'd act on first.

Check:

- [ ] The email and conversation totals match the file.
- [ ] The list of who counts as HR is right. If HR staff are counted as employees, the after-hours share is wrong.
- [ ] Outside parties and automated systems are separate from employees.
- [ ] Keep these numbers (after-hours share, time to first reply, conversations never answered). They're your baseline.

P1b · Sort every conversation

You'll get: every conversation sorted by request type, and by what it actually took to resolve, judged from the whole thread.

> Now sort the conversations into request types.
>
> Step 1: Propose 15–30 categories an HR leader would recognize (name + one-line definition). Read every subject line, and scan the message bodies too: some processes, like leave, show up under many different subjects. If our ticketing tool already has categories, start from those. Show me the list and wait for my edits.
>
> Step 2: Classify every conversation into one category. Then read the whole thread, not just the first message, and tag what it actually took to resolve by the end, using these tests:
> - General answer: the same answer for everyone; it could come from a document
> - Personal answer: the answer changes by person or by day; it needs this person's record
> - Action: one change in one system, finished in one conversation without anyone else acting
> - Workflow: someone other than the person asking has to act before it's done
> - Notification: an automated or vendor notice
> - Noise: nothing to do
>
> Judge by the end of the thread: a question that looks simple often ends with a vendor, an approval, or a payroll fix. Work in batches of 100 until all are done. Then save a CSV of conversation_id, category, level, and give me counts by category and level.

Tip: edit the categories in Step 1 before Claude classifies. Use your team's language.

Check:

- [ ] The classified count equals the conversation count.
- [ ] No "Other" bucket is bigger than a real category.
- [ ] You spot-checked 10 conversations by ID, reading each to the end.
- [ ] You downloaded the classification CSV. Every later question uses it.

Expect more Workflows than the first messages suggest. In our run, about half of all conversations were Workflows once judged from the whole thread.

P1c · Rank by effort, not volume

You'll get: the chart that shows where the work actually is.

> For each category, use code to compute: conversations, total emails, median emails per conversation, median days from first to last email, median number of people involved, and the share of conversations handed off to another team or vendor.
>
> Rank the categories by volume, then by an effort score (conversations × median emails × median people). Plot volume against effort and label the top categories. Which categories are high-effort but not high-volume?

Check:

- [ ] The ranking changes between volume and effort. If it doesn't, ask whether "people involved" counted everyone on the thread.
- [ ] Look hardest at the categories that are high-effort but not high-volume. That's usually where the surprise is.

P1d · Pick your first workflow

You'll get: a shortlist of what to build first. The category with the most effort isn't always the best first build: some need a person to finish the job no matter what. Make sure the handbook is uploaded to this chat, so Claude can score how well each category is documented.

> Now help me pick the first workflow to build. Take the top 8 categories by effort from P1c and score each one:
> - Team priority (0–3): has our team said this matters? Our priorities: [list them, or "none yet"]
> - Documented (1–3): 3 = our handbook or another document describes how it works; 2 = partly; 1 = only what we can infer from the emails
> - Volume (1–3): 3 = top quarter of categories by conversations; 2 = middle half; 1 = bottom quarter (don't count automated notices)
> - What AI could handle (1–3): 3 = AI could take it end to end; 2 = AI could take the intake and a person finishes; 1 = mostly expert judgment, outside parties, or fixing records by hand
> - Complexity (0–2, subtracted): 0 = 1–2 steps in one system; 1 = 3–5 steps or 2 systems; 2 = 6+ steps, 3+ systems, or legal risk
> Total = team priority × 3 + documented × 2 + volume × 2 + what AI could handle × 2 − complexity. Read at least 5 conversations per category before you score, and give one line of evidence with conversation IDs for each score. Then sort into: build first (15+), quick wins (11–14), needs scoping (7–10), later (under 7). Show the scores with and without team priority.

Check:

- [ ] Every score has evidence.
- [ ] The categories you know are hard landed in "needs scoping."
- [ ] You agree with the team priorities you gave it.
- [ ] You can say in one sentence which workflow you'll look at first, and why.

Why the weights: team priority counts most because a workflow nobody owns won't get finished. Complexity is subtracted because each extra system or legal rule adds weeks.

Question 2 · How does the work actually happen?

Start a new chat. Upload threads.csv and the classification CSV. P5 and P5b run later in this same chat.

P2 · Reconstruct one workflow

You'll get: how one request type really gets done, including the variants nobody wrote down. Start with a high-effort category from P1c.

> Using the classification CSV I uploaded, focus on [CATEGORY, e.g. leave requests]. Pull every conversation in that category and read each thread in full. Reconstruct how the work actually gets done:
> 1. Trigger: how requests arrive and how people describe them, in their own words.
> 2. Steps: in order, who does each step (employee, HR, manager, payroll, a vendor), and which system they touch.
> 3. Decisions: who decides what, and based on what.
> 4. Variants: are there different processes hiding under this one label? Group the conversations into variants and say what distinguishes them.
> 5. Friction: where it stalls, loops back, or gets reworked, and how often.
> Output a step table (step, owner, system, typical elapsed time) for each variant, and cite 2–3 conversation IDs per variant.

Check:

- [ ] There's more than one variant. There usually is.
- [ ] Every step has an owner.
- [ ] You read three of the cited threads yourself.
- [ ] You traced three closed cases by hand and wrote down who actually decided. Does that match Claude's answer?

Question 3 · Where are we missing policy?

Start a new chat for P3. Upload threads.csv and the handbook.

P3 · Compare practice with the handbook

You'll get: the rules that live only in HR's replies. For most teams this is the single most useful exercise, even if they never deploy AI.

> I've also uploaded our employee handbook. Compare what HR actually told people in the inbox with what the handbook says. Find:
> A. Unwritten rules: HR replies that apply a rule, exception, or threshold the handbook doesn't mention.
> B. Contradictions: HR replies that conflict with the handbook.
> C. Gaps: questions people ask repeatedly that the handbook doesn't answer.
> D. Inconsistencies: places where HR answered the same question differently in different conversations.
> For each finding: quote the HR reply, quote the handbook (or note that it's silent), list every conversation ID where it shows up, and count them. Sort by frequency. Work through the export in batches; don't stop until every conversation has been checked.

Check:

- [ ] The quotes are real. Check three against the conversation and the handbook.
- [ ] The counts add up across batches.
- [ ] The same rule isn't listed twice under different names.
- [ ] You downloaded the findings as a file, so you can merge runs later.

Then decide each top finding: codify it, change it, or retire it.

P3m · Merge results from several runs

Use it when you ran P3 (or P4) separately for each mailbox or quarter. Start a new chat and upload each run's findings file.

> I've uploaded the findings files from [3] separate runs of the same analysis, each covering different conversations. Merge them into one answer:
> - Treat the same rule found in more than one run as one finding, and combine its conversation IDs.
> - Count distinct conversations for each finding.
> - Keep the quotes, and sort the findings by count within each type.
> - List anything that appears in only one run with a single conversation in an appendix, so nothing is dropped silently.

Check:

- [ ] Every finding from every run is either merged, kept, or in the appendix.
- [ ] Merged counts add up to the counts in the original runs.

P3p · Turn the findings into a short action plan

Use it when the findings list is too long to act on. Run it in the same chat, right after P3 or P3m.

You'll get: a top 10, the quick wins you can finish this week, the calls only leadership can make, and anything that needs legal review, with the full ranked list saved as a CSV.

> Turn these findings into a short action plan I can actually work through. Use code where it helps.
> 1. Group findings that one fix would resolve, such as the same handbook section or the same policy decision.
> 2. For each group, write the action as one sentence that starts with a verb, and tag it:
> - Decision: correct the handbook, codify what HR already does, change the practice, or retire it
> - Reach: distinct conversations it covers
> - Risk: high if it touches pay, leave rights, safety, discrimination, or a legal deadline; otherwise medium or low, with one line on why
> - Effort: small (edit the text), medium (needs a policy decision), or large (needs legal review or a system change)
> - Owner: the role that should make the call
> - Blocks [WORKFLOW]: yes if an AI handling [WORKFLOW] would give a wrong answer until this is fixed
> 3. Rank by risk, then reach, then effort, smallest first.
> Show me the top 10 actions, the quick wins I can finish this week, the decisions to take to leadership (with the options for each), and anything that needs legal review. Save the full ranked list as a CSV.

Check:

- [ ] Every action names the findings it closes.
- [ ] If you plan to automate a workflow, the items that block it come first.
- [ ] A person, not Claude, makes each leadership call.
- [ ] Every action has an owner and a date.

Question 4 · When should a person take over?

Start a new chat. Upload threads.csv.

P4 · Map every handoff

You'll get: handoff rules written from what your team actually did, and a list of the escalations that were missed.

> First, use code to screen every conversation for signs of a handoff (a specialist, manager, or vendor added to the thread or forwarded in, or a reply from outside the mailbox it arrived in) or a serious-sounding word. Tell me how many candidates you'll read, then read each one in full.
>
> Find every conversation where the person who first handled it handed it to a specialist (Employee Relations, Legal, Safety, Payroll, a benefits manager, IT, or a vendor) or escalated it. For each one: what in the message signaled it, who took over, how quickly, and the conversation ID.
>
> Then find false alarms: conversations containing words that sound serious (court, attorney, lawyer, harassment, injury, terminate, complaint) where HR handled it routinely without escalating.
>
> Finally, write the handoff rules as a table: When [trigger] → route to [owner] → how [email, ticket, live handoff by phone or meeting, or employee self-service] → with [what context] → within [how long]. If the emails don't show how it happened, write "email" and flag it. If the rule's timing needs a faster channel than the one your team used (a phone call for a same-hour safety issue, say), give both. Flag any rule supported by fewer than 3 conversations, and any escalation that was missed or late.

Check:

- [ ] The screen covered every conversation.
- [ ] The triggers describe situations, not words.
- [ ] Every rule says how the handoff happens, not just who gets it.
- [ ] Missed or late escalations are listed with IDs, and thin rules are flagged.
- [ ] A person has reviewed every missed or late escalation.
- [ ] The team that owns each rule has confirmed it.

Counts of handoffs and misses only cover the threads Claude read, so treat them as a floor.

Question 5 · What do we need to connect?

Go back to the chat where you ran P2.

P5 · Write the brief for IT

You'll get: a one-page integration brief with specific fields, instead of "can you integrate our HR system?"

> The export, our classification CSV, and the P2 results for this workflow are all in this chat. For the [WORKFLOW] workflow, list every piece of information someone had to look up and every change someone had to make to finish the work. Group them by system (HR system, timekeeping, payroll, benefits platform, leave administrator, email, and so on). For each: read or write, the specific field or action, which step needs it, and how many conversations it appeared in.
>
> Then write a one-page brief for IT: systems in priority order, read access, write access, and the questions to ask each system owner.

Check:

- [ ] Fields are specific.
- [ ] Every write is tied to a step.
- [ ] Nothing sensitive is requested that the workflow doesn't need. For leave, no medical data is needed.

P5b · Document, build, or integrate

You'll get: the honest answer to "can we build this ourselves?", item by item. Run it in the same chat, right after P5.

> For [WORKFLOW], sort every step and every common request we found into one of three buckets:
> - Document: practice exists but isn't written down, or the written policy is wrong. The fix is to write or correct the policy.
> - Build: AI could do it today with no access to our systems, by asking questions, applying written rules, drafting a summary, and handing off to a person.
> - Integrate: it needs a live look-up or change in one of our systems (balances, hours, claim status, starting or ending a leave). Not possible without a connection to that system.
> For each item, give one line of evidence, the conversation IDs, and how many conversations it covers. Then give each conversation one primary bucket, set by the most demanding thing it needed (Integrate first, then Document, then Build), and use code to show the cumulative share it could handle: Build alone (end to end or a clean hand-off), then with the Document fixes, then as you add each Integrate connection, most valuable first.

Check:

- [ ] Nothing that needs a system look-up is in Build.
- [ ] Every Document item names the policy to fix.
- [ ] The shares are counted, not estimated.
- [ ] You know the first connection to ask IT for, and how much it adds.

What to do next

You don't need to build anything to get value from this. Most teams start here:

- **Fix the source.** Work the P3p list: quick wins this week, decisions to leadership, legal review for the rest.
- **Publish the answers.** Questions with one answer for everyone (about 1 in 5 in our run) belong in an FAQ or help page. That covers a lot of the after-hours mail.
- **Adopt the handoff rules.** Make the P4 rules your team's playbook, starting with safety, harassment, and terminations. Fix any rule where the channel is too slow: a live hazard needs a phone call, not an email.
- **Take the brief to IT.** The P5 brief names the systems, the fields, read or write access, and the questions to ask each system owner.
- **Measure, then re-run.** Keep your P1a baseline and re-run the prompts each quarter on a fresh export.