

Cameo — Automation Engineer

20 minutes with Adam Ostrum, Chief of Staff / VP People

What this is. Twenty minutes is short, and your interviewer is not an engineer — he is the Chief of Staff and VP People. That combination tells you almost everything about what this call is for.

The thesis: a Chief of Staff runs cross-functional operations; a VP People owns the internal employee experience. This role exists to **automate the internal company**. He is not just a gatekeeper — he is very likely one of your **primary customers**. Talk to him the way you would talk to a stakeholder whose workflow you are about to rebuild, not the way you would talk to a technical screener.

What he is screening for

Signal	Why he cares
Can you talk to non-technical people?	The JD says it outright: work effectively with technical and non-technical stakeholders. This call <i>is</i> the test — he is the non-technical stakeholder.
Will you find the right problems?	The role starts by mapping workflows and finding bottlenecks. Curiosity about other people's work is the core competence.
Are you self-directed?	"Comfortable with ambiguity, able to scope and prioritise work independently." There is no backlog waiting for you.
Do you actually use AI?	A hard requirement in the JD: 6+ months hands-on with Claude Code, Copilot, Cursor or equivalent <i>as part of a regular workflow</i> .
Do you want this job?	This is internal tooling, not product engineering. Some engineers consider that a step down. He will be listening for whether you find it genuinely interesting.

Budget the 20 minutes

Minute	What should be happening
0–2	Intros. He explains the role and why they opened it.
2–7	Your background — short, and angled at automation.
7–14	Why this role, how you approach it, AI. The core.
14–18	His questions on fit, logistics, Chicago.
18–20	Your questions. Have two.

1. The role, from the actual posting

Worth reading closely, because the posting is unusually specific about what they want and you can mirror its language back.

What you would do

- **Process analysis:** partner with Product, Marketing, Sales, Operations, Customer Success and **Talent Management** (the team that works directly with Cameo's celebrities and creators) to map workflows, find bottlenecks and spot automation opportunities.
- **Tool evaluation:** research and deploy automation platforms — the posting names **Retool, Zapier, Make**, workflow automation software and RPA tools.
- **Custom development:** build scripts, integrations and automations in **Python or JavaScript/Node** to connect systems.
- **Documentation and training** for the teams who will use what you build.
- **A roadmap:** maintain a prioritised automation plan aligned to company goals.

What they require

- 2–4 years writing production code in Python, JavaScript/Node or similar.
- **API fluency** — integrations, connecting systems together.
- Communication with technical *and* non-technical stakeholders.
- Self-direction and comfort with ambiguity.
- **AI-assisted development: 6+ months hands-on** with Claude Code, GitHub Copilot, Cursor or equivalent, as part of a regular development workflow.
- *Nice to have:* internal tooling experience, Zapier/Make/Power Automate/Workato, RPA tools, SQL and NoSQL, a background in operations or process improvement, startup pace.

The tell is one line in the posting: “prefer building practical solutions over perfect code.” This is a pragmatism screen. If you present as someone who wants to build the elegant general framework, you are describing the wrong person. The right person ships the ugly Zapier flow this week and replaces it with real code only when it earns that.

Context on Cameo

- Personalised video messages from celebrities to fans. Consumer marketplace — two sides, talent and fans, with an operations team in the middle.

- **Downtown Chicago HQ**, and the posting is HQ-based. Be ready to talk about location honestly.
- Their engineering stack, from the companion Staff Backend posting: **Node.js**, **MongoDB**, **AWS**, event-driven architecture with Kafka/queues, REST APIs.
- That same posting says they want people who “help define how we build with AI — from using agentic coding tools to move faster, to designing AI-powered systems.” AI fluency is a company-wide position, not a line item.

2. What to say

“Why this role?” — the question that decides it

An automation engineer role gets a lot of applicants who want a product engineering job and will leave when they find one. Your answer needs to make the case that this is the work you want, not a stepping stone.

The strongest honest framings:

- **Feedback loop.** You ship something on Tuesday and watch someone's Wednesday get better. Most engineering is much further from the person it helps.
- **Breadth.** Every department is a different problem. You learn how a company actually works, which is unusual for an engineer.
- **Leverage.** Automating twenty hours a week of someone's manual work is a larger, more measurable return than most feature work.
- **It is the AI-native job.** The gap between “this process is annoying” and “this process is automated” has collapsed. This role is what an engineer with agentic tooling can do that a team of five could not two years ago. That framing will land hard with a Chief of Staff.

How you would approach the first 90 days

He may not ask directly, but having a crisp answer makes you sound like you have done this before. A defensible shape:

1. **Go and watch.** Sit with each team, especially Talent Management, and observe the actual work. People describe their process wrong; you have to see it.
2. **Count.** Frequency $\times$ time per instance $\times$ number of people. That gives you hours per month per process, which is how you prioritise and how you later prove value.
3. **Ship one visible thing fast.** Pick something small, painful and popular. Credibility first; roadmap second.
4. **Then the boring question:** which of these should be automated, which should be *eliminated*, and which is a symptom of a broken upstream system? Automating a process that should not exist is the classic failure of this job — and saying so out loud is a strong signal.
5. **Instrument and document**, so a workflow that silently breaks does not become a trust problem.

Have one concrete story ready: a manual process you automated, how you found it, what you built, and hours saved. If you do not have a work example, a personal

one is fine — this role values the instinct more than the pedigree. What matters is that you noticed, quantified and shipped.

The AI answer

This is an explicit requirement, so answer it concretely and without hedging:

- **Name the tools and the workflow**, not just the tools. Which agent, for what class of task, and what you review before it lands.
- **Say what you keep**. Data model, anything touching money or customer data, anything where a plausible-looking mistake is expensive.
- **Give a number**. Something that used to take a day and now takes an hour.
- **Connect it to this role specifically**. Automation work is unusually well suited to agents: lots of small integrations, well-specified transformations, glue code, one-off scripts. You can now serve more internal requests than a person could before.

3. The technical rounds after this

Cameo is small enough that there is no public bank of tagged interview questions. Expect the later rounds to be run by engineers and to look like the job: **practical scripting, API integration, and pragmatic design** rather than hard algorithms. Prepare in that order.

Most likely: “here is a manual process, automate it”

The canonical shape: data lives in system A (a CRM, a spreadsheet, an internal admin), needs to end up in system B, on a schedule or on an event, with humans notified and errors visible. The competence they are testing is **integration engineering**, and it has a small number of ideas that come up every single time.

Concept	What to say
Idempotency	Retries and duplicate webhooks are guaranteed. Use an idempotency key so processing the same event twice is harmless. This is the single most important idea in the whole domain.
Retries + backoff	Exponential backoff with jitter. Jitter matters because synchronised retries become a thundering herd.
Dead-letter queue	After N failures, park the item somewhere a human can see it. Silent failure is what destroys trust in an automation.
At-least-once vs exactly-once	Exactly-once delivery does not really exist; you get at-least-once delivery plus idempotent processing, which is equivalent in effect.
Rate limits	Third-party APIs will throttle you. Respect Retry-After, batch where the API allows it, and queue rather than hammering.
Pagination	Cursor-based beats offset-based: offsets skip or duplicate rows when the underlying data changes mid-scan.
Reconciliation	A periodic job that compares both systems and reports drift. Every real integration needs one eventually.
Webhook security	Verify the signature, check timestamps to prevent replay, respond fast and process asynchronously.

A sync loop with the right bones

```
import time, random, requests
```

```
def with_retries(fn, attempts=5):
    """Exponential backoff with jitter; re-raises after the last try."""
    for i in range(attempts):
        try:
            return fn()
```

```

except requests.HTTPError as e:
    status = e.response.status_code
    if status < 500 and status != 429:
        raise # client error: retrying won't help
    if i == attempts - 1:
        raise
    wait = min(60, 2 ** i) + random.random() # jitter
    retry_after = e.response.headers.get("Retry-After")
    time.sleep(float(retry_after) if retry_after else wait)

def sync(source, sink, store):
    cursor = store.get_cursor() # resume where we left off
    for page in source.pages(after=cursor):
        for record in page.items:
            key = f"{record.id}:{record.updated_at}"
            if store.seen(key): # idempotency: skip duplicates
                continue
            try:
                with_retries(lambda: sink.upsert(transform(record)))
                store.mark_seen(key)
            except Exception as exc:
                store.dead_letter(record, str(exc)) # visible, not lost
        store.set_cursor(page.next_cursor) # only after the page is done

```

Talk through *why* each line is there: resume cursor so a crash does not restart from zero, idempotency key so a replay is free, 4xx not retried because it will never succeed, jitter to avoid herds, dead-letter so failures are visible, and the cursor committed only after the page is done so nothing is skipped.

Design prompts that fit this role

- **“Talent Management wants to know when a creator hasn't responded to a booking in 24 hours.”** — event vs polling, where the timer lives, what happens on redelivery, who gets notified and how you avoid alert fatigue.
- **“Ops needs to bulk-update a thousand talent records safely.”** — a Retool-style internal tool, preview and diff before commit, an audit log, permissions, and a way to undo.
- **“Sync our CRM with the production database.”** — which is the source of truth, how conflicts resolve, and reconciliation.
- **“Build an approval workflow.”** — a state machine, human in the loop, timeouts and escalation, and an audit trail. Model the states explicitly rather than with booleans.

If there is a conventional coding round

Assume medium difficulty and practical framing. The patterns most likely to appear in a role like this:

- **Hash-map aggregation and grouping** — counting, bucketing, joining two lists of records by key.

- **String and data parsing** — CSV/JSON wrangling, normalising messy input, date handling.
- **Intervals** — merging schedules, finding availability, detecting overlaps. Very common in anything involving bookings.
- **Rate limiting / sliding window** — directly relevant to calling third-party APIs politely.
- **SQL** — joins, GROUP BY, and a window function. The JD lists SQL and NoSQL as nice-to-have; expect at least a query.

```
from collections import defaultdict
```

```
def group_by(records, key):
    out = defaultdict(list)
    for r in records:
        out[key(r)].append(r)
    return out
```

```
def merge_intervals(intervals):
    intervals.sort()
    out = []
    for s, e in intervals:
        if out and s <= out[-1][1]:
            out[-1][1] = max(out[-1][1], e)
        else:
            out.append([s, e])
    return out
```

```
def chunked(seq, n):
    """Batch API calls: most endpoints take up to N ids at a time."""
    for i in range(0, len(seq), n):
        yield seq[i:i + n]
```

Closing the call

Two questions to ask him

- “Which team's workflow is the most painful right now — where would you want the first win?” This is the Chief of Staff's favourite kind of question, it gets you a real answer, and it signals that you think in terms of impact rather than tools.
- “How will you know in six months that this hire worked? Is it hours saved, processes automated, or something less countable?” It surfaces how the role is actually measured, which is worth knowing before you accept.
- *(If there is time:)* “Is this role expected to build things itself, or to enable other teams to build their own?” These are quite different jobs and the posting could be read either way.

Do

- **Be warm.** A VP People is evaluating whether other departments will enjoy working with you. In this role that is a technical requirement, not a soft one.
- **Ask about their pain.** If you can get him talking about a process that annoys him, the call is going well — and you will learn what to say next.
- **Speak in outcomes.** Hours saved, errors avoided, people unblocked. Not frameworks.
- **Be honest about Chicago** and about what you want. Twenty minutes is too short to fake alignment, and this is the person who would find out.

Don't

- Don't go deep technical. He is not the audience, and it reads as not knowing your audience — which is the exact skill being tested.
- Don't describe this as a stepping stone to product engineering.
- Don't dismiss no-code tools. “I'd rather write it properly” is the wrong instinct here; the posting explicitly prefers practical over perfect.
- Don't be vague about AI. It is a stated requirement, and hedging on it is the easiest way to fail this screen.

Twenty minutes, one job: come across as an engineer who genuinely likes making other people's work easier, is comfortable being handed an ambiguous mess, and already works with agents daily. The algorithms can wait for the next round.