centralise

DECISION GUIDE

FOUR WAYS IN · WHICH ONE DOES WHAT

Connector, CLI or API?

HubSpot now gives you four ways to put AI to work in your portal.
Each one suits a different kind of job.
This is a guide to what each is good at, where it runs out, and how to pick.

Built by Centralise: HubSpot Elite Partner and member of the Anthropic Partner Network.

The four ways in

Most people meet this the same way. You connect HubSpot to Claude, ask it something about the pipeline, get a better answer than you expected, and then hit a wall the first time you want it to change more than a handful of records. At that point it looks like AI on the CRM is a demo rather than a tool. It isn't. You have outgrown the easiest of four connections, and the other three exist for different work.

01 CHAT

HubSpot connector for Claude

.
Add HubSpot from the connector directory in Claude and sign in. Ask questions and make small changes from a chat. It reads and writes standard records and engagement history. No terminal, no code, nothing to maintain

02 AGENT

HubSpot Agent CLI

BETA

A command-line tool HubSpot wrote for AI agents rather than for people. Claude Code, Claude Cowork and Codex install it and work the portal
by chaining commands together. This is the one that handles volume.

03 CODE

The REST API

The full platform, and what every integration you already own runs on. Your engineers, your infrastructure. The only route of the four that can start work on its own when something changes in HubSpot.

04 BUILD

Developer MCP server

A local install for teams building on HubSpot: apps, CMS themes, UI extensions, serverless functions. It gives coding tools the context of HubSpot's developer docs. It does not read your CRM records.

HOW TO THINK ABOUT THE FOUR

Two questions sort them.

Who or what starts the work: a person in a chat, an agent on a schedule, or an event in HubSpot. And how much can change in one go if the instruction was wrong: ten records, a whole object, or the entire portal. Answer both and the choice usually makes itself.

The failure cases are symmetrical. One team spends six weeks building an integration for a question they will ask twice a year. Another gives an agent write access across the contact database with nothing between it and every report that that depends on lifecycle stage.

Nothing below assumes you have a developer. Two of the four routes work without one.

Side by side

The Developer MCP server is not in this table because it does not touch CRM records. For most mid-market teams the rows that settle it are whether it runs with nobody watching, how many records one action can touch, and what a wrong instruction reaches.

	CONNECTOR FOR CLAUDE	AGENT CLI (BETA)	REST API
Who starts the work	You, in a chat	You or a schedule, through an agent	Your systems
Runs with nobody watching	No	Yes	Yes
Starts on a CRM event	No	No	Yes, through webhooks
Records per write action	Ten at a time (beta limit)	Bulk, by piping results between commands	100 per batch call
Custom objects	Not on the published list	Yes, documented	Yes
Preview before it writes	Ask, then read what it proposes	--dry-run on every write command	Whatever you build
Audit trail	Per user, through OAuth	Per user with OAuth. None with a service key	Per app
Who can run it	Anyone on a paid Claude plan	Anyone comfortable in a terminal or Cowork	Developers
Time to first useful result	Minutes	An afternoon	Weeks
What one bad instruction reaches	Ten records	A whole object, unless you gate it	The whole portal

If you read nothing else

Four jobs, four routes. Match the job to the route and most of the risk goes away before you start.

Ask a question	The connector	One-off, exploratory, a handful of records. You get an answer in minutes, and the ten-record limit stops you doing much harm while you learn.
Repeat a process	The Agent CLI	The same job every week across hundreds or thousands of records. Put a dry run in front of it and prove it in a sandbox.
React to an event	The API and workflows	Something has to happen whether or not anyone is watching. No AI route does this today.
Build on HubSpot	Developer MCP server	Apps, CMS assets, UI extensions. It is developer tooling, not a way into your records.

- Two questions settle it: who starts the work, and how much one wrong instruction can reach.

01
The connector: questions, & small changes

You add HubSpot from the connector directory in Claude and sign in. From then on Claude sees what your own HubSpot user sees and nothing else. Permissions are inherited rather than redefined, which is the part to understand before a team gets hold of it: if a rep can edit a deal in the interface, they can edit it through Claude, and the same goes for everything else their role reaches.

WHAT IT IS GOOD AT

Questions you had not planned to ask. This beats building a report because the follow-up question costs nothing.

Small changes inside a conversation. Log the call you have just come off, create the follow-up task, correct one lifecycle stage.

Reading engagement history at a volume nobody would work through by hand. Months of emails, calls and notes on one account, summarised before a renewal.

WHAT YOU TYPE

Show me every deal closing this quarter with no logged activity in 14 days. Group by owner. Flag anything over £25k, and tell me what the last recorded touch actually was.

What comes back is a grouped table you can work from, plus the reasoning behind the ordering. Ask it to draft check-in emails for the top three and you have the work as well as the analysis. That is a five-minute exercise that currently sits in somebody's Friday afternoon.

WHERE IT RUNS OUT

Bulk create and update stop at ten records at a time. That is HubSpot's documented beta limit, and it is where nearly every team meets the ceiling.

It only works while you are in the chat. No schedule, no triggers. Close the window and it stops.

No access to custom Sensitive Data properties. If your portal has Sensitive Data switched on, check what that does to engagement access before you build a process on top of it.

Custom objects are not on HubSpot's published list of what the connector can reach. If your commercial model lives in custom objects, this route will not see it.

02
The Agent CLI: the job you keep repeating

HubSpot wrote a command-line tool aimed at AI agents rather than at people. The agent installs it, signs in, and works out what it can do by reading the tool's own help output. The difference from the connector is composition: one command's output feeds into the next, so an agent can find a set of records, write a plan covering all of them, then apply it. No ten-record ceiling.

WORTH USING FOR

The job you do every week across hundreds or thousands of records: the Monday cleanup list, the property audit, the pipeline hygiene pass. Anything that currently means an export and a re-import.

Structural work the connector cannot reach: properties, pipelines, saved views, workflows, custom object schemas, association labels.

Call and meeting transcripts, with recording links and AI summaries, for coaching or win/loss themes.

Saved reports written in CRM SQL, so the answer lands in HubSpot rather than in a chat window.

WHAT TO WATCH

It is in beta. HubSpot says commands, flags and behaviour may change without notice, and writes can permanently delete live data. Test in a sandbox and dry-run before applying.

Something still has to start it. A scheduled agent is not the same as a webhook, and there are no CRM event triggers here either.

Two sign-in modes with very different consequences. OAuth mirrors the user's permissions and leaves a per-user trail. A service key gets account-level access and, in HubSpot's words, no per-user audit trail. Default to OAuth.

On Claude Team or Enterprise an admin has to allowlist api.hubapi.com before Cowork can install it.

02
The Agent CLI: the job you keep repeating

FIND, PLAN, REVIEW, THEN APPLY

This pattern is worth understanding even if you never open a terminal.

```
hubspot objects search --type contacts --filter "email~example.com" \
      | hubspot objects update --type contacts --property tier=gold \
        --dry-run > plan.jsonl
cat plan.jsonl | hubspot objects update --type contacts
```

The first command writes a plan rather than a change: a file listing every record it would touch and what it would do to each, which you can read, edit or throw away.

Only the second writes anything, and destructive commands refuse to run until a dry run has produced a safety digest.

03
The API: anything that has to happen on its own

None of this changes what the API is for. If the requirement starts "when a deal moves to Closed Won" or "every night without fail", it is API and workflow work.

An agent waits to be asked. A webhook fires whether or not anyone is at a keyboard.

THE RIGHT TOOL WHEN

Something has to happen the moment a record changes, with nobody present.

You are syncing two ways with finance, product or a warehouse, and reconciliation matters more than flexibility.

Volume. Batch endpoints move up to 100 records per request, against a limit that counts requests rather than records.

There is an SLA attached, or an auditor who will ask about it later.

THE COST

Weeks rather than minutes, and engineers who will still be here in a year.

Every failure mode is yours: retries, backoff, 429 handling, schema drift. HubSpot expects error responses to stay under 5% of daily requests.

Too much machinery for a question you will ask twice.

RATE LIMITS

PRIVATELY DISTRIBUTED APPS	PER 10 SECONDS	PER DAY
Free and Starter	100 per app	250,000 per account
Professional	190 per app	625,000 per account
Enterprise	190 per app	1,000,000 per account

Marketplace OAuth apps are capped at 110 requests every ten seconds per installing account, and the API limit increase add-on does not lift that. Search endpoints are stricter again. Webhook calls fired from workflows do not count towards the limit at all, which is one reason event-driven design costs less traffic than polling.

> The daily allowance is shared across every app in the account, and every route in this guide draws on it. That is how a long agent session ends up slowing a nightly sync. It shows as data arriving late rather than as an error, which is why it takes a while to notice.

Where this usually goes wrong

01

Bulk cleanup through the chat connector

A re-segmentation job across a few thousand contacts, ten records at a time. The workaround people find is to tell Claude to loop it in batches of ten. It works, and it gets round a limit HubSpot put there on purpose. If the job is volume, use the route built for volume, with a dry run in front of it.

02

An integration built for a one-off question

Developer time, plus a maintenance liability that outlives whoever asked for it, to answer something the connector handles in a chat. Build the integration when the answer has to arrive without anyone asking.

03

Expecting any of it to watch HubSpot for you

None of the four connections fire on a deal stage change, a form submission or a contact going cold. They wait to be asked. If the requirement contains the word "when", it is workflow and webhook work.

Before you connect anything

Six decisions worth making deliberately rather than discovering later.
None takes long. All are harder to unpick once a team has settled into a way of working.

- Decide which sign-in mode you are using, and why. OAuth by default, service keys narrowly and for admins only.
- Insist on a dry run for every write until a pattern has worked twice.
- Settle your Sensitive Data position first if you are in financial services, healthcare, or anything else regulated.
- Switch on App Governance so a Super Admin controls who can connect the Agent CLI.
- Prove it in a sandbox before it goes near the live portal.
- Look at what your HubSpot roles actually permit today. An agent inherits them exactly as they are.

Which route you need depends on the job.

IF YOU WANT A SECOND OPINION

Book a call with our team

Thirty minutes. Bring the thing you were about to automate and we will tell you which route it needs, or whether it needs one at all.

→ centralise.com

The teams that get value from any of this tend to fix the data first. Properties nobody owns, lifecycle stages that contradict the deal pipeline, attribution that falls apart when you look twice. An agent pointed at a portal like that will still answer, and the answer will sound right.

centralise

centralise

Centralise is a revenue systems integrator for mid-market companies.

HubSpot Elite Partner and member of the Anthropic Partner Network.

We build the connection layer between the two and the data foundations underneath it.