

WHITE PAPER · DELIVERD CONTEXT

Own your *context*

One supplier, five agents on five different AI platforms, and one shared record of what they know, what they are working on and how the job gets done. A worked example of Knowledge, Objectives and Playbooks.

SUMMARY

Your agents learn things. Where does that knowledge live?

Most organisations now run more than one AI platform. Procurement uses one assistant, the risk team another, finance builds agents in Microsoft, engineers code with a third. Each keeps its own memory, and that memory belongs to the platform.

When an agent learns that a supplier portal logs people out after fifteen minutes, or that the risk owner must approve any supplier that uses AI on your data, that lesson usually stays inside one conversation on one platform. Nobody has checked it. The next agent does not get it. When you change model or tool, it is gone.

Deliverd Context keeps it in one place that you own, and gives it to every agent that does the work, whichever platform it runs on. It has three parts.

KNOWLEDGE

What your agents know

Facts, conventions, decisions, procedures, terms, contacts and known issues. Agents propose them. Only a person confirms one, and only confirmed entries reach an agent.

`search_knowledge` ·
`get_knowledge` ·
`propose_knowledge`

OBJECTIVES

What they are working on

The outcome a piece of work is for, a short task state any agent can resume from, side tasks for the detours, and findings and decisions a person confirms.

`get_task_context` ·
`update_task` ·
`create_side_task`

PLAYBOOKS

How the job gets done

A job written down once: its purpose, the inputs a run needs, and each step with the check that applies before the agent acts. Your governing roles put it in force.

`list_playbooks` ·
`get_playbook` ·
`start_playbook`

Underneath all three is a record. Every time knowledge or a playbook step is handed to a piece of work, Deliverd writes down when, to which agent, which entries, and who confirmed each one. So when someone asks what an agent knew when it acted, there is an answer.

This paper follows one job from start to finish: onboarding a new supplier at an invented company, with five agents on five platforms sharing one context. Each section of the feature appears where it does real work.

THE PROBLEM

Four ways context goes wrong with more than one agent

It is locked in

Memory kept by a model provider or an agent framework moves with that vendor. Switching platforms, or running two, means starting over or copying notes by hand.

The goal gets buried

Long work drifts. Forty messages into debugging, neither the model nor the person can say what the work was for, and a second agent picking it up gets a transcript with no clear state.

Nobody checked it

An agent's memory is whatever it decided to remember. A wrong lesson spreads as easily as a right one, and there is no one whose job it is to say "yes, that is how we do it".

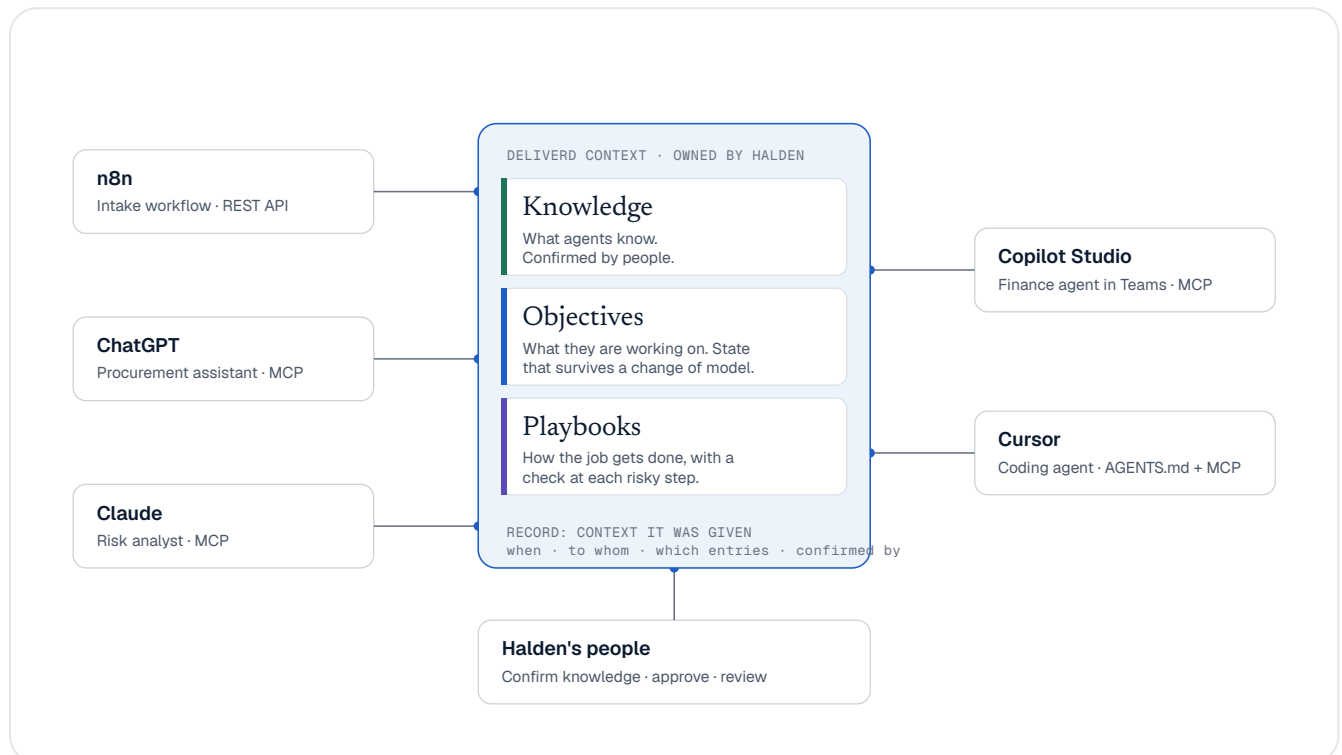
Each agent does the job its own way

Without the job written down, the risky step is handled differently by every agent: one asks for approval, another checks a policy, a third just acts. An auditor sees no single process.

THE SCENARIO

Halden onboards a new freight supplier

Halden is an invented 600-person distributor. Its logistics team wants to start using Corran Freight, an invented regional carrier whose route planning uses AI. Halden has five agents involved in the job, on five platforms.



Every agent reads from and proposes to the same context. People decide what becomes knowledge and approve the risky steps.

[n8n](#) · [REST API](#)

Intake workflow

Watches the supplier request form and starts the onboarding playbook when a request arrives.

[ChatGPT](#) · [MCP connector](#)

Procurement assistant

Gathers the supplier's onboarding pack and chases what is missing.

[Claude](#) · [MCP connector](#)

Risk analyst

Assesses the supplier's AI use and drafts the risk summary.

[Copilot Studio](#) · [MCP tool](#)

Finance agent in Teams

Answers staff questions and creates the supplier record once policy allows.

[Cursor](#) · [AGENTS.md](#) and [MCP](#)

Coding agent

Helps an engineer build the sync that sends approved suppliers to the finance system.

What Halden's agents already know

Before Corran Freight came along, Halden's knowledge base already held entries its people had written, imported from an existing notes file, or confirmed from agents' proposals. Each entry is one thing an agent should rely on, of one of seven kinds. Each confirmed entry has a review date, six months out unless someone picks another.

These are the entries this job will touch. **K8** does not exist yet; an agent proposes it on day four.

K1	Decision	A supplier that uses AI on our data needs the risk owner's approval as well as the business owner's. Why: agreed at the risk committee after a review of supplier AI use.	Confirmed by the risk owner
K2	Fact	Logistics suppliers must hold at least €5m public liability cover.	Confirmed by the finance controller
K3	Procedure	To chase missing documents, send one message listing everything outstanding, allow five working days, then tell the business owner.	Confirmed by the procurement lead
K4	Known issue	The supplier portal ends a session after fifteen minutes without warning. Workaround: prepare every file first and upload them in one go.	Proposed by an agent, confirmed by the procurement lead
K5	Convention	Record a supplier's legal name exactly as on the company register, with its registration number.	Confirmed by the finance controller
K6	Glossary	"Business owner" means the person accountable for the relationship and its spend, who is often not the person who asked for the supplier.	Confirmed by the procurement lead

K7

Contact

Questions about a supplier's bank details go to the treasury team's shared mailbox, never to the supplier's own contact.

Confirmed by the finance controller

K8

Proposed on day four

The finance system rejects supplier names over 60 characters.

Proposed by the coding agent

Shorten the display name only; keep the full legal name in the reference field.

Agents propose. People confirm.

An agent records what it learns with `propose_knowledge`. The entry waits until a person confirms or rejects it. Until then, no agent is given it. No agent can confirm knowledge, whatever its key allows.

Confirmed words never change

To change a confirmed entry, someone revises it. The revision is a proposal; once confirmed it replaces the original, which stays on the record as superseded. You can always see what an agent was told at the time.

The job, written down once

Halden did not start from a blank page. Deliverd lists a **Supplier onboarding check** starter, worded for any business. Halden's procurement lead opened it as a draft, linked the knowledge particular steps rely on (K2 and K3 to the gap check, K1 to the approval, K5 to creating the record), and added one step of their own: creating the supplier record behind a policy gate on `supplier.create`. Halden's compliance admin then put it in force. From that moment its words are fixed, and a change becomes a new version.

PLAYBOOK · IN FORCE

Supplier onboarding check

Inputs a run needs: **supplier**, **service**, **business_owner**, and optionally **uses_ai**

- | | | |
|---|--|-----------------|
| 1 | Collect the onboarding pack
List each document with its date. Do not contact the supplier yet. | No check |
| 2 | Check it is complete and flag gaps
Draft one message asking for everything outstanding. | Person reviews |
| 3 | Assess AI use
Open a supplier AI assessment if the service uses AI or this is not known. | No check |
| 4 | Draft the risk summary
One page, citing the documents relied on, with a recommendation. | Person reviews |
| 5 | Business owner and risk owner approve
The person who asked for the supplier may not approve it. | Person approves |
| 6 | Hand over for set-up
Tell the team that keeps the supplier list, with the approval's link. | No check |
| 7 | Create the supplier record (added by Halden)
Check the policy for <code>supplier.create</code> and do what it answers. | Gate check |

Each check is a call the agent already knows

A review step means `request_review`, an approval step means `request_approval`, a gate step means `check_gate` for the named action. The playbook names the call in the step's goal, so no agent has to guess it, and the risky moment is handled the same way whichever agent runs it.

Who can put it in force

Owners, administrators and compliance admins put a playbook in force directly. Anyone else can draft one and request approval; it goes in force when that approval is approved. Only the version in force runs.

Four days, five platforms, one objective

Running a playbook starts an **objective** with the playbook's purpose and one side task per step. Each side task holds its step's instruction, the run's inputs and the confirmed knowledge that step relies on. Here is the run, beat by beat. Days and times are illustrative.



Monday 09:12

n8n

Starts the run

A request form starts the playbook

Halden's logistics manager fills in the supplier request form. The n8n workflow behind it calls Deliverd's REST API to start the onboarding playbook, passing the inputs and the form's own submission ID as an external ID, so a retried webhook returns the same run instead of starting a second one.

Deliverd creates the objective **Onboard Corran Freight** with seven side tasks, and records that version 1 of the playbook, and the entries its steps rely on, were given to it.

CALL

POST /api/v1/playbooks/{id}/start

INPUTS

supplier: Corran Freight Ltd · service: regional
pallet delivery with AI route planning ·
business_owner: head of logistics · uses_ai:
yes



Monday 09:40

ChatGPT

Steps 1 and 2

The procurement assistant collects the pack

The procurement assistant in ChatGPT reads the first step's context. It does not need a long prompt about Halden: the context hands it the instruction, the inputs, and the knowledge that applies, including the portal timeout (K4) and the legal-name convention (K5). It lists eleven documents with their dates.

On step 2 it compares the pack with the checklist. Corran's insurance certificate shows €2m public liability, below the €5m in K2. Following K3, it drafts one message listing everything outstanding and calls `request_review`. The procurement lead reviews the draft, approves it, and the message goes to Corran.

CALLS

`get_task_context` `update_task`
`add_finding` `request_review`

CONTEXT GIVEN INCLUDES

Playbook step 1 and 2 · K2, K3, K4, K5

A secret is refused. The assistant tries to save the supplier portal's login as knowledge. Deliverd refuses an entry that looks like it holds a password, key or token, because knowledge goes to every agent that asks. The assistant proposes "the portal login is kept in the procurement vault" instead, and the procurement lead confirms it.

Tuesday 14:05

Claude

Steps 3 and 4

A different platform picks up exactly where the last one stopped

The risk analyst runs in Claude, with no shared memory with ChatGPT. It does not need any. Its context for step 3 gives it the objective, the task state (insurance chase sent, waiting on Corran, next action: assess AI use), the decisions made so far, and K1 and K6.

It opens a supplier AI assessment for Corran. The answers show route planning runs on a model from a third-party provider, and nobody knows where delivery addresses are processed. Rather than bury that in the main thread, it opens a **side task**: "Find out where route data is processed and how long it is kept". The side task starts with a snapshot of the objective, not a copy of the conversation.

The side task comes back with a result: the provider's region, its retention period, the evidence it relied on, and a suggested decision. Once that result is applied, its decisions are added to the objective and its suggested next action becomes the objective's. A decision recorded by an agent stays a suggestion until a person confirms it. The analyst then drafts the one-page risk summary and calls `request_review`.

CALLS

`get_task_context` `create_side_task`
`add_finding` `request_review`

CONTEXT GIVEN INCLUDES

Task state and decisions so far · K1, K6

Wednesday 10:30

People

Step 5

Two people approve, with the summary attached

Corran sends a new certificate with €5m cover. Step 5 asks the head of logistics, as business owner, and the risk owner, because K1 says a supplier using AI on Halden's data needs both. Each sees the risk summary and the evidence. Both approve, and the decision is recorded with who made it.

CALL

`request_approval`

WHY TWO APPROVERS

Decision K1, confirmed by the risk owner, linked to the playbook

Wednesday 16:40

Copilot Studio

Steps 6 and 7

The finance agent in Teams hands over and checks the gate

A colleague asks the finance agent in Teams: "Who do I talk to about Corran's bank details?" It searches knowledge and answers from K7: the treasury team's shared mailbox, not the supplier's contact. That answer came from an entry a person confirmed, not from the model's guess.

On step 6 it tells the supplier-list team that onboarding is approved, with the approval's link. On step 7 it calls `check_gate` for `supplier.create` and does what Halden's policy answers before it creates the record, using the legal name and registration number as K5 requires.

CALLS

`search_knowledge` `get_task_context`
`check_gate`

CONTEXT GIVEN INCLUDES

Steps 6 and 7 · K5, K7

Thursday 11:15

Cursor

Learns something new

A coding agent reads the same context from a file

An engineer is building the sync that sends approved suppliers to the finance system. They download Halden's context as an **AGENTS.md**, holding the confirmed knowledge and the playbooks in force, and put it in the repository. Cursor reads it like any other project file, so it already knows the legal-name convention.

Testing against the finance system, the agent finds that names over 60 characters are rejected. It proposes K8 through Deliverd's MCP server. The integration lead confirms it, and the next agent on any platform that touches supplier records is given it.

CALLS

`deliverd knowledge agents-md`
`propose_knowledge`

CONTEXT GIVEN INCLUDES

AGENTS.md: the confirmed knowledge and the playbook in force

THE RECORD

What did the agent know when it acted?

A month later, Halden raises the insurance minimum for logistics suppliers to €10m. The finance controller revises K2. Once the revision is confirmed it replaces the original, and the €5m entry is kept on the record as superseded. From then on, every agent is given the new figure.

At the next audit, the question comes up: Corran had €5m cover, so why was it approved? The objective's page answers it under **Context it was given**.

WHEN	GIVEN TO	WHAT	CONFIRMED BY
Mon 09:12	Objective, at start	Supplier onboarding check, version 1, and the entries its steps rely on	Put in force by the compliance admin
Mon 09:41	Procurement assistant (ChatGPT)	K2 Insurance minimum €5m (<i>now superseded</i>), K3, K4, K5	Finance controller; procurement lead
Tue 14:06	Risk analyst (Claude)	K1 Two approvers for AI suppliers, K6	Risk owner; procurement lead
Wed 16:41	Finance agent (Copilot Studio)	K5, K7	Finance controller

The rule at the time was €5m, it was confirmed by a named person, and the agent was given that version. Nobody has to reconstruct this from chat logs on three platforms.

OWN YOUR CONTEXT

Change platforms without starting over

The next quarter, Halden's procurement team decides to move its assistant to a different platform. There is nothing to migrate. The knowledge, the open objectives with their task state, and the playbooks all live in Deliverd. The new assistant connects to the same MCP server and gets the same context on its first call.

And if Halden ever leaves Deliverd, its context leaves with it. From the Context overview, or from a terminal, knowledge downloads as Markdown that imports back into any organisation, as JSON, or as an AGENTS.md for tools that read files. Import works the other way too: a notes file Halden already kept,

such as an AGENTS.md or CLAUDE.md, becomes one proposed entry per heading, ready for a person to confirm.

From a terminal

```
deliverd knowledge agents-md --out  
deliverd knowledge export --out kno  
deliverd knowledge import CLAUDE.md
```

After the run, the work improves itself

Having done the job once, the risk analyst can draft an "Annual supplier review" playbook with `draft_playbook`, using what it learned. It stays a draft until Halden's compliance admin puts it in force, or approves the request to.

REFERENCE

What each part does, and why it matters

PART	WHAT IT HOLDS	WHO CHANGES IT	WHY IT MATTERS
Knowledge	Facts, conventions, decisions, procedures, glossary terms, contacts and known issues, each with a review date	Agents and people propose; people confirm, revise and retire	Every agent works from the same checked facts, and what one agent learns reaches the rest
Objectives	The outcome, a short task state, side tasks, findings and decisions	Agents update the state and suggest decisions; only a person changes what the work is for	Work survives a restart, a new session or a different model, and the goal stays in view
Playbooks	Purpose, inputs and steps, each with no check, a policy gate, an approval or a review	Anyone drafts; owners, administrators and compliance admins put one in force	The job is done the same way by every agent, with the risky step governed every time
Context it was given	When, to whom, which entries and playbook version, and who confirmed each	Written by Deliverd every time context is handed out	The answer to "what did the agent know when it acted?"
Export and import	Markdown, JSON and AGENTS.md out; Deliverd exports and notes files in	Downloaded from the Context overview, the API or the CLI	Your context is yours, on any platform and if you leave

How agents connect

- Any MCP client: Claude, ChatGPT, Cursor, Codex, VS Code, Windsurf, Gemini CLI, and Copilot Studio agents. Seven Context tools sit beside Deliverd's approval, gate and review tools.
- REST API and SDKs for TypeScript and Python, for workflows such as n8n.
- AGENTS.md for coding agents and other tools that read files.

Limits by design

- An agent is given at most eight relevant confirmed entries with a piece of work, so its context stays short and on point. It can search for more.
- Entries that look like secrets are refused. Personal data is allowed, and you are told when an entry contains some.
- Entries and playbooks can be kept inside a workspace, readable by its members and administrators.
- Context is included in the Team, Business Starter, Business and Enterprise plans.

What your agents know, what they are working on, and how the job gets done.

Shared by every agent, confirmed by your people, owned by you. Read the guide at deliverd.dev/docs/guide/context, or connect an agent to Deliverd's MCP server and ask it to search your knowledge.

Deliverd Engineering Team · October 2026. Halden and Corran Freight are invented for this paper, as are the times, people's roles and knowledge entries. Platform names identify the tools the example agents run on; they do not imply endorsement or a customer relationship.