TECHNOLOGY WHITE PAPER

3PM Ecosystem Technology White Paper

Independent applications, one shared entity layer, scoped access instead of replication, and a governed interface for agentic AI in construction.

VERSION	ISSUED	SECTIONS	READING TIME
1.0	28 August 2026	15 plus appendices	About 35 minutes

> Written for architects, engineering leaders and technically minded evaluators. Assumes familiarity with service boundaries, identity and access models, and tool-using language models. Assumes no knowledge of construction.

Contents

How to read this document

Sections 2 to 4 set out the problem and the shape of the platform, and are the minimum for understanding why the rest exists. Sections 5 to 7 are the substance for anyone assessing the data model and the access model. Section 8 covers the agent programs, section 10 walks one request end to end, and section 13 states where this architecture is weaker. Connector and device figures throughout are generated from the same source the website renders, so this document cannot quietly drift from the product.

Traces are illustrative in their values and accurate in their sequence and tool names. Where a claim is a design intent rather than a measurement, it is written as one.

SECTION 1

Executive summary

Construction software does not fail at the model. It fails at the data. This paper sets out how the 3PM ecosystem is built: independent applications, each with its own database, sharing one authoritative set of entities through scoped access rather than replication, with a governed interface through which AI agents read and act.

The argument runs as follows. A construction business runs many functions, and each function wants software shaped around it. Integrating those functions by copying records between them produces drift, reconciliation work and an audit trail nobody trusts. Consolidating them into one schema produces a system that fits no function well and cannot be changed without agreement from every team. 3PM takes a third route: applications stay independent, the entities they share are referenced from a common layer, and the platform, not each application, enforces who may see which field and for what purpose.

That architecture was chosen with agents in mind. An agent is only as useful as the context it can be trusted with. When context and permission already exist as platform concerns, adding an agent becomes a configuration task rather than an integration project. 3PM also runs four agent programs of its own beneath the applications, whose users are other agents rather than people: IoT Hub, Extract, Forms and Flow. They turn telematics feeds, documents, field capture and multi-step work into governed records on the shared layer.

What this paper claims

1. Records are referenced, not copied, which removes a class of reconciliation and drift failures rather than managing it.
2. Applications keep their own databases and schemas, and can be released, scaled and fail independently.
3. Two applications may hold different fields on the same entity, so no team is constrained by another team's schema.
4. Field-level access and purpose control are enforced by the platform, once, rather than reimplemented per application.
5. A new application integrates against the shared layer once, instead of building point-to-point connections to every existing application.
6. The context and permission model an agent needs is already present, so agent capability is configured rather than engineered.

> **Who this is for**
>
> Architects, engineering leaders and technically minded evaluators. It assumes familiarity with service boundaries, identity and access models, and the general shape of tool-using language models. It does not assume any knowledge of construction.

SECTION 2

The problem: the last mile of construction data

The gap between what a construction business knows and what its software can answer is not a modelling problem. It is a plumbing problem, and it has three distinct causes.

2.1 Proprietary feeds, private schemas

Telematics providers, machine manufacturers, positioning systems, tool platforms and sensor networks each publish their own schema, units, identifiers and update frequency. A single excavator can carry a manufacturer telemetry feed, an aftermarket tracker, a machine control system and a fuel sensor, each naming the machine differently. None of them is wrong. None of them agrees. The reconciliation is left to whoever is asked the question.

2.2 Half the record is a document

Drawings, consents, specifications, contracts, delivery dockets, timesheets, certificates and site photographs hold facts that exist nowhere else in structured form. A supplier docket is the only evidence of what actually arrived. A consent condition is the only statement of what is legally required. These are not edge cases in construction, they are the primary record.

2.3 The rest exists only in someone’s head

What was agreed on site, why a sequence changed, which subcontractor accepted a variation verbally: this information is real, decision-relevant, and not written down anywhere until somebody is asked. Any system that assumes complete structured input will be wrong about a construction project.

2.4 Why the two usual answers do not work

The first usual answer is integration by copying. Each application keeps its own copy of the client, the project and the site, synchronised by jobs and webhooks. It works until two copies disagree, at which point the business has a reconciliation process, a defensive habit of trusting nothing, and no way to say which value was authoritative at the time a decision was made.

The second usual answer is one system for everything. A single schema, a single database, one release train. It ends with a schema that satisfies no function precisely, change control that requires agreement across teams with different priorities, and a blast radius that makes small improvements expensive.

Point a capable model at either arrangement and it will produce confident answers from partial evidence, which is worse than no answer. The architecture in the rest of this paper exists to make the evidence complete, attributable and scoped before an agent ever sees it.

SECTION 3

Architecture overview

3PM is a set of independent construction applications over a shared entity layer, reached through one governed access layer. Applications own their own data. The entities they have in common are referenced from the shared layer rather than duplicated into each application.

3.1 Independent applications

Each application in the ecosystem is a separate service with its own database and schema, shaped around the function it serves. Estimate is built for measure, rates and tender assembly. Drive is built for inspections, defects and maintenance

intervals. Move is built for dispatch and proof of delivery. They do not share tables, they are not released together, and one falling over does not take the others with it.

3.2 Reference, not replication

When two applications need the same client, project, site, asset or worker, they hold a reference to one record on the shared layer rather than a copy of it. An address corrected in one application is correct everywhere, immediately, because there was only ever one address. There is no synchronisation job to fail, no drift to detect and no reconciliation to schedule.

3.3 Contextual flexibility

Shared identity does not mean shared shape. Two applications may hold entirely different fields against the same entity: dispatch needs an access note and a gate code against a Site, cost control needs a budget code and a retention percentage. Both attach to the same Site without either team having to negotiate the other's schema. The shared layer holds identity and the facts that must agree. Everything else stays local to the application that cares about it.

3.4 Consequences of the choice

No data replication	Records are referenced rather than copied between applications, which removes an entire class of reconciliation and drift problems.
Independent applications	Each application has its own database and schema, and can be released, scaled or fail independently of the others.
Contextual flexibility	Applications hold different fields on the same entity, so no single team is constrained by another team's schema.
Enforcement at the platform	Field-level access and purpose controls are applied by the platform rather than reimplemented in each application.
Incremental expansion	A new application integrates once, against the shared layer, instead of building point-to-point connections to every existing application.
Agent readiness	The context and permission model agents need is already in place, so adding an agent is a configuration task rather than an integration project.

> **The trade being made**
>
> This architecture accepts a harder access layer in exchange for a simpler data story. Every cross-application read is a scoped, audited request rather than a local query, which costs latency and engineering discipline. What it buys is that no two applications can hold different answers to the same question.

SECTION 4

The four layers

The platform is described in four layers: the applications teams use, the shared source of truth, the models and agent programs, and one governed way in. Each layer has a single responsibility and talks to its neighbours through defined interfaces.

LAYER	RESPONSIBILITY	HOLDS
1. Application layer	Serve a construction function well, in its own shape, at its own release cadence.	Its own database, schema and MCP connector.
2. Shared data layer	Hold one authoritative view of the entities applications have in common.	The 3PM Lattice: identity, relationships, provenance.
3. AI agent layer	Reason over ambiguous input and turn it into governed records and drafted work.	Frontier and specialised models, plus the four 3PM agent programs.
4. Access layer	Authenticate, scope, route and audit every request from any client.	3PM Auth, the MCP interface, policy and the audit log.

4.1 Direction of travel

It is worth being precise about which way the layers point. Applications sit above the shared data layer and read from it. The agent programs sit in the AI layer but work downwards, for the data layer: their output is records, not answers. A user-facing agent, by contrast, works upwards, consuming the shared layer through the access layer to answer a question or draft an action. The same layer therefore serves two directions, and confusing them is the most common misreading of this architecture.

4.2 The applications today

The application layer is the part that grows. Each new application integrates once against the shared layer and inherits identity, scoping and audit rather than implementing them. The current set is listed in Appendix A.

SECTION 5

The 3PM Lattice

The Lattice is the shared data layer: a small, deliberately conservative set of entities that every application refers to, with the relationships between them and the provenance of every value.

5.1 The entity set

The Lattice holds the entities that must mean the same thing in every application. It is intentionally short. Anything an application can own without the rest of the business needing to agree stays in that application.

ENTITY	WHAT IT SETTLES
Organisation	One client, supplier or subcontractor record shared by every application. An address changed in one application is reflected in all of them.
Person	Contacts, subcontractors and suppliers held as a single identity across the platform.
Project	The quote, the schedule, the deliveries and the payments all point at the same project.
Site	One site record covering location, access and details, shared by the applications that need it.
Asset	Plant, vehicles, equipment and materials, recorded once and visible where relevant.

Worker	Roles, competencies and certifications follow the worker between applications.
Contract	Contracts and agreements linked to the right people, projects and payments.
Invoice	Invoicing and payments connect back to the exact project, contract and client.

5.2 Identity resolution

A serial number, a registration plate, a device identifier and an internal asset code can all refer to one excavator. Resolving those to a single Asset is the Lattice's central job, and it is done by the agent programs rather than by asking users to keep a mapping table. Resolution is recorded, not assumed: when an identifier is attached to an entity, the evidence and the confidence of that match are stored with it, and a low-confidence match can be surfaced for a human rather than silently accepted.

5.3 Provenance

Every value on the Lattice carries where it came from: which application, which agent program, which source document and page, or which person and when. This is what makes an AI-assisted record defensible. An agent that reports plant utilisation against a project can be asked to show the position pings, the geofence definition and the provider that supplied them. A figure extracted from a supplier docket can be traced to the page it was read from.

5.4 Write rules

- Writes are scoped: an application or agent may write only the fields its purpose requires, and only against entities it has been granted.
- Writes are attributed: every change records the acting identity, whether a person, an application or an agent, and the request that carried it.
- Conflicting writes are resolved by authority, not by recency. The application that owns a fact wins, and the losing write is retained as a claim rather than being discarded.
- Raw telemetry and source documents stay in their source systems. What lands on the Lattice is the resolved meaning, with a pointer back to the evidence.

5.5 What the Lattice does not hold

It is not a data warehouse and it is not a document store. It does not accumulate every position ping, every draft, or every application is working state. Keeping it small is what keeps it authoritative: a shared layer that grows to hold everything becomes the monolithic schema this architecture exists to avoid.

SECTION 6

3PM Auth: one identity, scoped access

Users and agents authenticate once. Each application and each agent is then granted only the fields its function requires. Access is a property of the platform, expressed as scope and purpose, and enforced at the point of the request.

6.1 Field-level scope, worked

The same client and project resolve differently depending on who is asking and why. The table below is the canonical example: three applications, one client record, three different views of it.

FIELD	DRIVE	ES-CROW	COM-MAND	WHY
Client name and address	yes	yes	yes	Delivery routing, payee verification and client management each require it.
Verified bank account	no	yes	no	Only payment release requires bank details, so only Escrow is granted them.
Project budget and costs	no	no	yes	Delivery does not depend on cost data and escrow is scoped to the payment.
Trust account balance	no	yes	no	Held for the payment function alone.
Site access notes	yes	no	yes	Needed to reach the site and to plan work on it.

The point of the example is not the specific grants, which are configuration. It is that the grant is expressed once, at the platform, and cannot be circumvented by an application choosing to ask a different way.

6.2 Workspaces and multiple parties

Construction work is done by parties who are not employees of the same business. Workspaces model that: a subcontractor, a client or a joint venture partner is granted a scoped view of the packages, projects or payments that concern them, without a licence for everything else and without a copy of the data leaving the platform. A joint venture in which a partner needs programme visibility but not cost visibility is a scope configuration, not a bespoke integration.

6.3 Agent identity

An agent is an identity in its own right, not a user impersonation. It holds its own scopes, and where it acts on behalf of a person it carries a delegated scope that cannot exceed that person’s own. Two consequences follow: an agent can be granted narrower access than any human user, which is usually the right default, and every action an agent takes is attributable to the agent and to the delegation behind it.

> **Approval is a scope, not a setting**
>
> Whether an agent may act, or only propose, is expressed in its scope rather than in application logic. An agent with drafting scope and no execution scope cannot send, pay or commit, whatever it is asked to do and whatever an application would otherwise permit.

SECTION 7

The access layer and the MCP interface

Every request into 3PM, from an application, a person or an agent, arrives through one governed access layer. For agents that layer speaks MCP, the open protocol tool-using models use to reach external systems.

7.1 Why a protocol rather than an API key

A conventional API assumes the caller knows what it wants and has been granted it in advance. An agent arrives with an intention expressed in language and needs to discover what it may do. MCP gives it a described tool surface it can enumerate, and gives the platform a place to apply scope before any tool executes. The agent does not need to know which application holds a fact, or which telematics provider a value came from.

7.2 Request lifecycle

1. Authenticate. The caller presents its identity, whether a person, an application or an agent with a delegated scope.
2. Scope. The request is intersected with the caller's granted fields and purposes. Anything outside the grant is not filtered from the response, it is never read.
3. Route. The access layer resolves which application or Lattice entity holds the answer and calls it. The caller is not told where the data lives.
4. Execute. Reads return scoped data. Writes are checked against write rules and attributed to the caller.
5. Audit. The request, the scope applied, the decision and the result are recorded, so an agent chain is as auditable as a single request.

7.3 Trace of a scoped agent request

```
auth     identity=agent:site-assistant  on-behalf-of=user:foreman.42
scope    granted=[project.read, asset.read, site.read]  denied=[invoice.*]
tool     asset.utilisation(project="hillcrest-stage-2", window="2026-07-21..2026-07-27")
route    Lattice/Asset + Drive/Utilisation  (2 sources, 1 request)
result   4 assets, 61.4 h working, provenance retained per value
audit    logged id=req_8f21c0  duration=380ms  scope-violations=0
```

Illustrative trace. Tool names are real, values are examples.

7.4 Bringing your own agent

Because the interface is MCP rather than a private SDK, a customer's own agent, or a general assistant such as Claude or ChatGPT, connects through the same layer as 3PM's own agents and is subject to the same scoping and audit. There is no privileged internal path. That is a deliberate constraint: it means the governance story does not depend on which agent is asking.

SECTION 8

Agent programs and agent to agent

3PM runs four agent programs of its own. They are headless: their users are other agents, not people. They sit in the AI layer and work for the data layer, turning proprietary feeds, documents and field capture into governed records, and they call each other over agent to agent.

8.1 Why agents rather than scripts

The last mile of construction data is ambiguous by nature: a docket that does not match the order, a drawing revision that changed one dimension, a machine identifier that appears in three formats. A script fails on ambiguity and stops. A

program that can reason about it proposes a resolution, attaches its confidence and the evidence, and escalates what it cannot settle. That is the difference these four are built on.

8.2 IoT Hub

IoT Hub stands between the equipment yard and the graph. It connects to telematics providers, plant and machine controllers, environmental sensors, fuel and load monitors, trackers and site gateways, each with its own schema, units, identifiers and update frequency, and reconciles them into one model. A serial number becomes an Asset, a geofence crossing becomes a Site arrival, a shift pattern attaches to a Worker, and a run of position pings becomes a utilisation figure against a Project. Raw telemetry stays in the source system.

Takes in	Telematics providers, machine OEM feeds, GNSS and machine control, tool tags, sensors and site gateways.
Publishes	Asset, Site, Worker, Project, utilisation events.
MCP tools	asset.whereWas, asset.utilisation, site.arrivals, plant.exceptions
Escalates	Identifier matches below confidence threshold, and telemetry that contradicts a prior resolved fact.

8.3 Extract

Extract turns documents into data. It reads drawings, consents, specifications, contracts, invoices, delivery dockets, timesheets, certificates and site photographs, then resolves what it finds against records that already exist rather than creating duplicates: this invoice belongs to that project, this certificate belongs to that worker, this docket matches that delivery. Every extracted value carries its source document, page reference and a confidence score, so a downstream agent can tell the difference between a figure it may act on and one that needs a human.

Takes in	Drawings, consents, specifications, contracts, invoices, delivery dockets, timesheets, certificates and site photographs.
Publishes	Invoice, Contract, Project, Worker, document provenance.
MCP tools	document.extract, document.classify, record.reconcile, evidence.trace
Escalates	Low-confidence reads, and any extraction that would create a second record for something that may already exist.

8.4 Forms

Forms is capture designed for the graph rather than for paper. A site diary, inspection, permit or quality sign-off is presented as the few questions that are genuinely new, prefilled from what the platform already knows and from the previous entry, and what comes back lands as structured records rather than a PDF in a folder. The design goal is completion rate: a form that takes ninety seconds gets filled in honestly, and one that takes five minutes gets filled in on Friday for the whole week.

Takes in	Field capture from phones and tablets, often offline, plus the context the platform already holds.
Publishes	Project, Site, Worker, Asset, inspection and compliance events.
MCP tools	form.request, form.status, capture.attach, evidence.trace
Escalates	Answers that contradict a known state, such as an inspection passed on a machine flagged out of service.

8.5 Flow

Flow is what makes agent to agent a system rather than a set of one-off calls. It holds the sequence, the state and the rules for work that spans several agents, several applications and several people: what runs next, what happens on failure, what may proceed automatically and what stops for a human. A single instruction becomes a run in which other programs are delegated to, and the result is written to the Lattice once every step has passed. Every run is recorded, replayable and attributable, and no outbound action executes without the approval its policy requires.

Takes in	Instructions from people, events from the Lattice, schedules, and requests from other agents.
Publishes	Project, Contract, Invoice, run history.
MCP tools	flow.start, flow.status, flow.approve, flow.audit
Escalates	Any step whose policy requires human approval, and any failure that leaves a run partially applied.

8.6 Agent to agent

The programs delegate between themselves rather than being orchestrated by an application. Flow asks IoT Hub to confirm a plant window; Extract asks Forms to collect the signature a docket is missing. Each returns a structured result rather than prose. Every hop is authenticated, scoped and logged, which is what makes a chain of agents as auditable as a single request. MCP is how an agent reaches a tool; agent to agent is how agents reach each other.

SECTION 9

Connectors: what plugs into IoT Hub

Connectors are written per provider. Meaning is modelled per device class. It is the meaning that lands on the Lattice, which is why adding a provider is a connector rather than a platform change. 18 providers across 5 categories are covered today.

9.1 Categories

CATEGORY	WHAT COMES FROM IT	PROVIDERS
Fleet & vehicle telematics	Utes, trucks, trailers and light vehicles. Position, distance, driver, road user charges and compliance.	EROAD, Teletrac Navman, black-hawk.io, Geotab, Samsara, Argus Tracking
Plant & machine OEM	Manufacturer telematics off the machine itself. Engine hours, fuel burn, fault codes and service intervals.	Caterpillar, Komatsu, John Deere, Volvo CE, Hitachi
Positioning & machine control	Survey, GNSS and machine guidance. As-built levels, volumes moved and progress against the design model.	Trimble, Topcon
Tools & equipment tracking	Tags, readers and tool platforms. Which tools are on which site, who has them and what is due for test.	Hilti, Milwaukee

Sensors & site gateways	Environmental, structural and access sensors on LoRaWAN, cellular and site gateways.	Bosch, Siemens, LoRaWAN & cellular

9.2 Device classes

Whatever the provider, a device belongs to a class, and the class determines what it can tell the platform. This is the layer that keeps the rest of the architecture ignorant of vendors.

CLASS	TYPICAL HARDWARE	SIGNAL
Vehicle trackers	Utes, trucks, trailers	Position, trips, distance, driver
Plant controllers	Excavators, cranes, hoists	Engine hours, load, fault codes
GNSS & machine control	Rovers, blades, buckets	Levels, volumes, as-built surfaces
Fuel & load cells	Tanks, weighbridges, bins	Consumption, weights, dockets
Environmental	Concrete, dust, noise, water	Maturity, exceedance, thresholds
Tags & readers	Tools, plant, materials	Last seen, custodian, movement
Cameras & vision	Progress, safety, security	Progress frames, detections
Access & gates	Turnstiles, hoardings, keys	Who is on site, and when
Site gateways	LoRaWAN, cellular, edge	Backhaul for everything above

9.3 Adding a provider

1. Connect the way the provider expects: its API, its webhook, its message format, its update frequency.
2. Map its identifiers to the device class and to the Lattice entity they resolve to, with the evidence for each match.
3. Translate its units and its event vocabulary into the shared events, such as arrived, worked, idled, left, out of service.
4. Nothing above the AI layer changes. No application, agent or report needs to know a provider was added.

Devices on open networks need no provider connector at all: anything reachable by API, webhook, MQTT topic or a nightly file can be ingested against a device class directly.

SECTION 10

Worked example: closing out a slab pour

An end to end example, because the architecture only makes sense in motion. The instruction is one sentence from a project manager: close out the Hillcrest slab pour. Four programs, three applications and one human approval are involved.

```
instruct  user:pm.07  "close out the Hillcrest slab pour"
flow      run=run_4471 policy=commercial-signoff  steps=4

a2a       Flow -> IoT Hub    confirm plant on site for the pour window
a2a       IoT Hub -> Flow    confirmed 07:58 to 16:04, pump truck and boom on site
           evidence: 214 position pings, geofence Site/hillcrest-stage-2
```

```
a2a       Flow -> Extract    read the concrete supplier docket
a2a       Extract -> Flow    34.5 m3 delivered, docket 88213, confidence 0.97
           evidence: docket page 1, matched to Project/hillcrest-stage-2

a2a       Flow -> Forms      collect the QA sign-off from the foreman
a2a       Forms -> Flow      signed by user:foreman.42 at 16:41, 3 photos attached

gate      commercial approval required before write  ->  user:qs.03 approved 17:02
write     Lattice: Project progress, Asset utilisation, Invoice expectation 34.5 m3
audit     run_4471 replayable, 11 attributed steps, 0 scope violations
```

Run trace. Illustrative values, real tools and real sequence.

10.1 What the architecture contributed

- Nobody typed a quantity. The 34.5 cubic metres came from the supplier docket, with the page it was read from retained.
- Nobody reconciled the plant hours. The window came from position data resolved against a geofence, not from a timesheet.
- The record is one set of facts, not four copies. Cost control, dispatch and payments read the same Project and Asset.
- A human approved before anything was written, because the policy on this run required it.
- The whole run can be replayed and attributed months later, when a claim depends on what happened that afternoon.

10.2 What it did not do

It did not decide whether the pour was acceptable. It did not price the variation. It did not release a payment. Those are judgements with consequences, and the platform is designed to bring the evidence to a person rather than to remove the person.

SECTION 11

Model strategy

The AI layer is deliberately not a single model. Frontier models are used where broad reasoning matters, specialised and smaller models where a narrow task is better served by one, and the choice is a platform decision rather than an application one.

11.1 Routing and fallback

Requests are routed by task rather than by preference: document classification, extraction, scope reasoning, drafting and conversation have different requirements. Routing, fallback between providers, guardrails and cost control sit in the AI layer, so an application never hard-codes a model and a change of provider is not a change to the applications.

11.2 Context is scoped before it is prompted

A model only ever receives context the calling identity was entitled to. Scoping happens at the access layer, before assembly of a prompt, which means the confidentiality question is answered by the permission model rather than by prompt hygiene. An agent with no invoice scope cannot leak an invoice into a prompt, because it was never able to read one.

11.3 Where determinism is preferred

Not everything should be inferred. Arithmetic, unit conversion, rate application, permission decisions and audit are handled deterministically. Models are used for judgement over ambiguous input: what this document is, which record it refers to, what a run of telemetry means, how to phrase a draft. Keeping that boundary sharp is what makes the output reviewable.

SECTION 12

Governance, audit and assurance

An AI-assisted record is only useful if it can be defended. Three properties make that possible: attribution, provenance and replay.

Attribution	Every read and write records the acting identity, the delegation behind it and the scope applied. Agents are identities, not impersonations.
Provenance	Every value on the Lattice can be traced to its source: an application, a document and page, a device and provider, or a person and a timestamp.
Replay	Multi-step runs are recorded as sequences of attributed steps and can be replayed, which is what makes an agent chain examinable rather than merely logged.
Policy	What may proceed automatically and what stops for a human is expressed as policy on the run, not as logic inside an application.

12.1 Failure containment

Because applications hold their own databases, the failure of one does not corrupt the others. Because the shared layer is referenced rather than copied, a bad write is a single correction rather than a fan-out of stale copies. Because writes are scoped, the worst an over-eager agent can do is bounded by its grant, which is the only bound worth relying on.

12.2 Questions worth asking any vendor

- Show me a value and trace it to the document, device or person it came from, without leaving the product.
- Show me what an agent was permitted to read at the moment it answered, not what it is permitted to read now.
- Show me a multi-step automated action replayed after the fact, with each step attributed.
- Tell me which parts of the answer were inferred and which were computed.
- Tell me what happens to the other copies of this record when I correct it here.

SECTION 13

Limits and honest failure modes

A white paper that only lists strengths is a brochure. These are the places where this architecture and these programs are known to be weaker, and what we do about them.

Hand-drawn and poor-quality documents	Typed documents extract cleanly. Hand-annotated markups, photographed drawing sets and microfiche scans do not, and are flagged for manual review rather than guessed at.
Unusual geometry in measurement	Automated takeoff is strongest on orthogonal work. Curved and radiused elements warrant a manual check, and the product is built to make that check easy rather than to claim it is unnecessary.
What no drawing shows	Heritage substrates, unknown services and undocumented as-builts are not in the inputs, so they cannot be in the output. Site knowledge remains a human contribution.
Local reality outside the feeds	Road closures, weight restrictions and local arrangements are not knowable from telematics. They are maintained as constraints, and they need an owner.
Cross-application reads cost latency	Referencing rather than copying means some reads cross a scoped boundary. That is a deliberate trade against reconciliation, and it makes caching and read patterns an engineering concern.
Data quality is still upstream	A duplicated asset register or an unmaintained cost library degrades every layer above it. The platform surfaces that faster than a spreadsheet does; it does not fix it for you.

SECTION 14

Adoption patterns

The architecture supports incremental adoption, and that is how it is best taken up. Nothing here requires replacing a finance system or migrating a live project mid-stage.

14.1 Run alongside, integrate rather than replace

Project delivery data belongs on the platform; the general ledger usually does not. The common pattern is delivery here, finance where it already is, with the two integrated at defined points such as committed cost and invoice expectation. Applications integrate against the shared layer once, so a finance connector is not repeated per application.

14.2 Start where the evidence is worst

The fastest demonstrable value tends to come from whichever part of the business currently retypes the most: supplier invoices, pre-start inspections, a document archive nobody has read. That is also where provenance and attribution are most obviously better than what they replace.

14.3 Migrate at boundaries

Bring new work onto the platform and let in-flight work finish where it is, or migrate at a stage boundary rather than mid-stage. Attempting to move a live project in the middle of a stage costs more than it returns.

14.4 Give data quality an owner

One named person per project or per yard who owns the asset register, the cost codes and the entity mapping. Every implementation that drifts, drifts here first.

SECTION 15

Glossary

3PM Lattice	The shared data layer: the entity set every application refers to, with relationships and provenance.
3PM Auth	The identity and access model. One authentication, field-level scope and purpose per application and agent.
Access layer	The single governed entry point. Authenticates, scopes, routes and audits every request.
MCP	Model Context Protocol, the open standard tool-using models use to reach external systems. 3PM exposes its scoped tool surface through it.
Agent to agent (A2A)	Delegation between agents. Authenticated, scoped and logged per hop, returning structured results rather than prose.
Agent program	A headless 3PM agent whose users are other agents: IoT Hub, Extract, Forms and Flow.
Provenance	The recorded origin of a value: application, document and page, device and provider, or person and timestamp.
Scope	The set of fields and purposes an identity may read or write. Applied by the platform at request time.
Workspace	A scoped view granted to a party outside the account, such as a subcontractor, client or joint venture partner.
Device class	A category of hardware defined by what it can report, independent of who manufactured it.

APPENDIX A

The applications

Each application is an independent service with its own database, integrating once against the shared layer. This list is generated from the live product catalogue at the time this document was issued.

Contractor	Available
Base	Available
Command	Available
Drive	Available
Move	Available
Estimate	In development
Escrow	In development
Deconstruct	In development
Scope	In development
Source	In development
Buddy	In development

Connector status vocabulary

Provider connectors are published with one of three states, and the website shows the same three: live connector, in build, on the roadmap. A provider listed in section 9 is not an assertion that every device it sells is supported, only that the connector and the device classes named are.

APPENDIX B

MCP tool surface of the agent programs

The tools the four agent programs expose through the MCP layer. Every call is authenticated, intersected with the caller's scope, routed, executed and audited. Tools are listed for orientation; the described surface an agent enumerates at runtime is authoritative.

PROGRAM	ROLE	TOOLS
IoT Hub	Machines, vehicles and sensors	asset.whereWas, asset.utilisation, site.arrivals, plant.exceptions
Extract	Documents into structured data	document.extract, document.classify, record.reconcile, evidence.trace
Forms	Capture designed for the graph	form.request, form.status, capture.attach, evidence.trace
Flow	Orchestration and approval	flow.start, flow.status, flow.approve, flow.audit

Questions, or a technical walkthrough

The interactive version of this material, including the scope explorer, the agent request traces and the full connector list, is at www.3pm.app/resources/technology. For an architecture session with the engineering team, or for the security and data processing detail that sits behind section 12, get in touch.

hello@3pm.app · 3pm.app