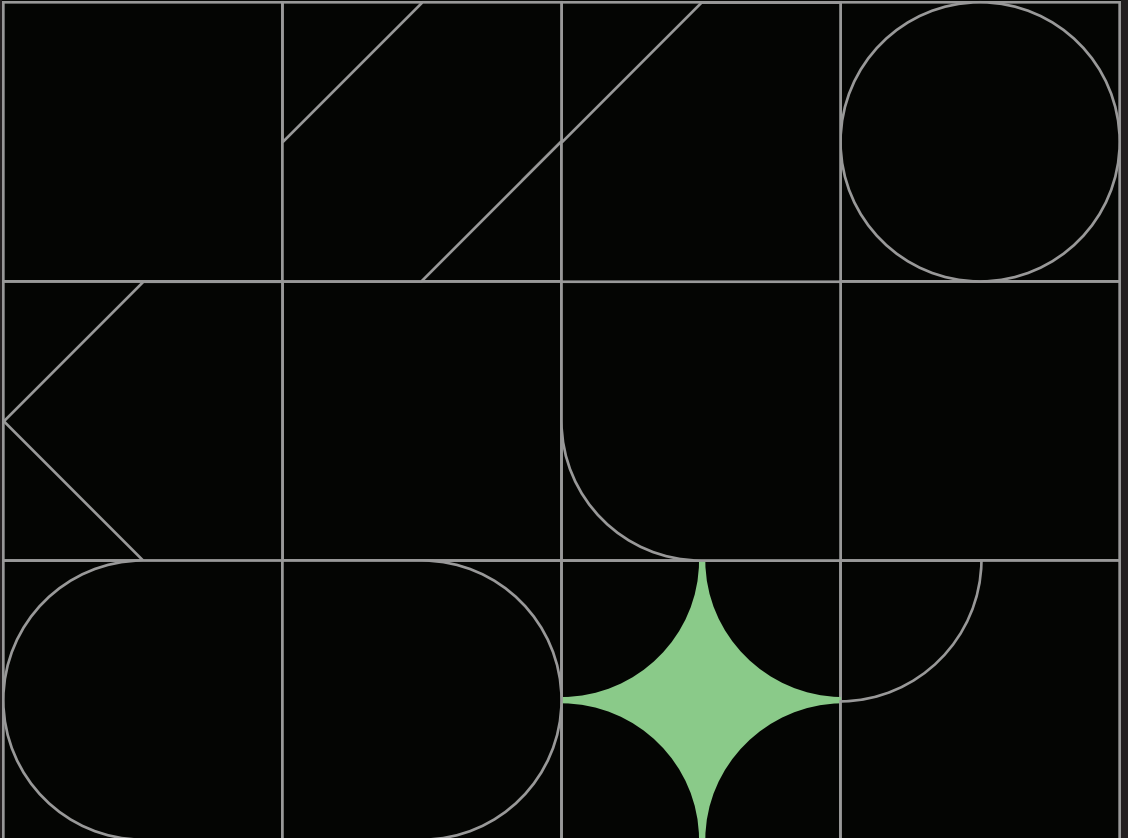


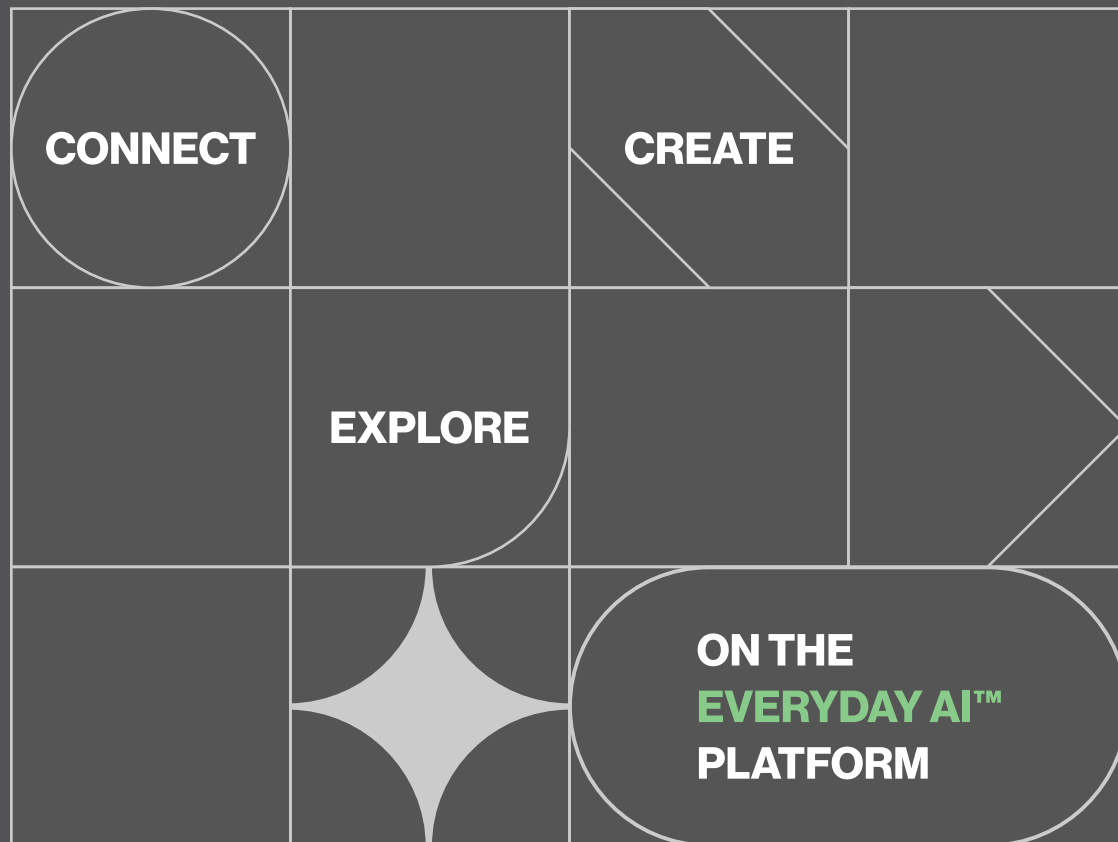


The **ROI Playbook** for AI in Multifamily

A guide for housing enterprises taking AI past the pilot:
How to measure the return, and how to stop losing it between the first
use case and the fifth.



Build with Travtus.



Contents

Introduction: Past the pilot.	2
What AI-native actually means	4
How much independence should we give?	5
Cost, Data, Trust	6
The Everyday AI Platform: How you build	8
Measure it with a unit you already know	11
Step 1: Connect	13
Buy context, not another application	14
What “connected” means in a portfolio	16
Step 2: Explore.	18
Find the return before you build	19
Step 3: Create	21
The four things you build	22
Build once, reuse everywhere	24
Match autonomy to the cost of being wrong	25
Set the minimum bar before you ship	27
Holding the return as usage grows	28
Guardrails keep you moving	30
Your first 90 days	32
Next steps	34

Introduction: Past the pilot

Most large housing enterprises have stopped asking whether AI works. They have seen it work. They have run the leasing assistant, the call summarizer, the maintenance triage pilot, and watched each one do more or less what it promised inside its own narrow lane.

The harder question is why so little of it reaches the P&L.

The pattern repeats. A pilot proves out in one region. Widening it means a new integration, a new set of permissions, a new vendor conversation. By the time the second use case is live, the cost of getting there has consumed the gain from the first. Twelve months on there are six AI tools, six invoices, six ways of working, and nothing compounds.

That is not a model problem. It is an architecture problem, and it shows up in how you run the business rather than in the technology budget.

Returns in housing are no longer won on cheap deals or cheap financing. They are won on how well the whole enterprise runs: the decisions it makes and the work it executes. Add high labor costs and the tribal knowledge locked in people's heads, and the pressure lands squarely on the operating model. A departmental AI tool cannot reach that.

The enterprises pulling ahead have stopped buying AI features and started building their own. They set the hard parts up once: the context, the permissioning, the governance, the human checkpoints. Everything after that reuses them. The first use case takes a quarter. The fifth takes a fortnight. That gap is where the return actually lives.

In this playbook

A practical way to size, capture and hold the return on AI across a housing portfolio, following the three steps of how you build: **Connect, Explore, Create**. It is written for the people who have to defend the number: the COO, the CFO, the CTO and the asset management team.



95%

**automation rate on live resident
communication workflows¹**

~15%

**productivity gain across teams
building on the platform¹**

1. Measured across Trivantis enterprise deployments spanning 500,000+ units under management, including Cortland, MAA, BH, Continental and Preiss.

What AI-native actually means

In a traditional enterprise, people perform the work, software helps execute it, and AI is a feature bolted on: a chatbot, a forecast. The shape of the organization is unchanged, and so is the cost base.

An AI-native enterprise is a consequence of reasoning models and the architecture they allow. It works backwards from four goals, and each one is a lever on the return:

01 **Context as the foundation,** not an afterthought. This is the part that appreciates.

02 **Workflows become agentic,** flexible and reasoning-driven rather than hard-coded.

03 **Intelligence and decisions are centralized,** so the enterprise functions as one.

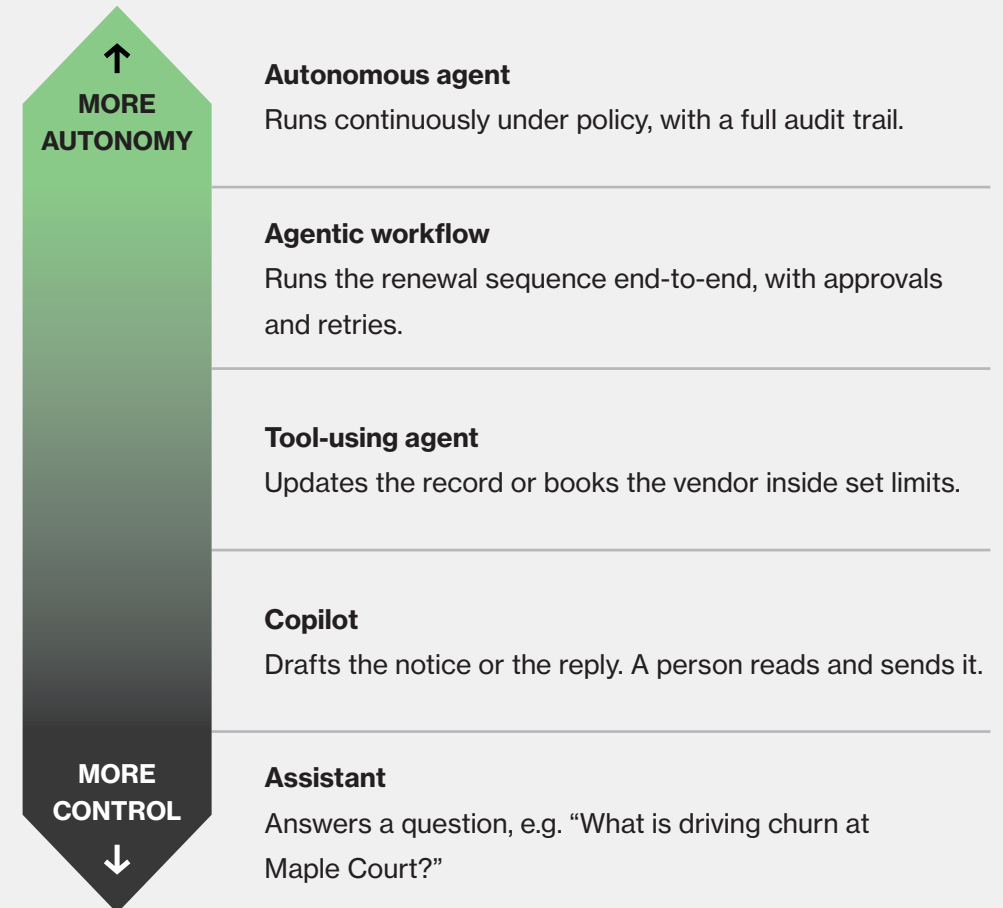
04 **Proprietary capability at scale.** Because you build your own, your knowledge becomes the advantage.

Two behaviors follow, and both are worth money.

Transparency: When intelligence is centralized and always on, leaders can see what is happening everywhere without being on the ground.

Flexibility: Because your teams build the reports, scores and workflows themselves, you decide what matters and change it as the business changes, instead of waiting on a vendor.

How much independence should we give?



The further up you move, the less the return depends on the model and the more it depends on governance. That is good news. Governance is something you build once and reuse.

Cost, Data, Trust

Today's approach to AI has created three problems. They show up after the pilot, not during it, and between them they account for most of the value that goes missing on the way to the portfolio.



Cost

Every point solution bundles its own infrastructure. Three vendors, each with their own email, telephony and payment calls. Making them interoperable means an endless demand for APIs, webhooks and vendor coordination, and the fees rise with every one you add.

Each new use case becomes its own integration project, so time goes into wiring rather than outcomes and spend scatters across so many line items that nobody can tie cost back to a specific workflow.



Data

Every solution sits in its own orbit with no context sharing. The leasing assistant cannot see the maintenance history, and the maintenance triage cannot see the payment record.

Whatever cross-business reasoning you were promised lands on the data team as another pipeline request, and the promise of AI stays siloed.



Trust

Product and roadmap dependency, sequential releases, and you do not own your IP. The logic that encodes how your business actually runs, your exception handling, your escalation rules, your definition of an at-risk resident, ends up owned by a vendor. That quietly caps what the investment can ever be worth.

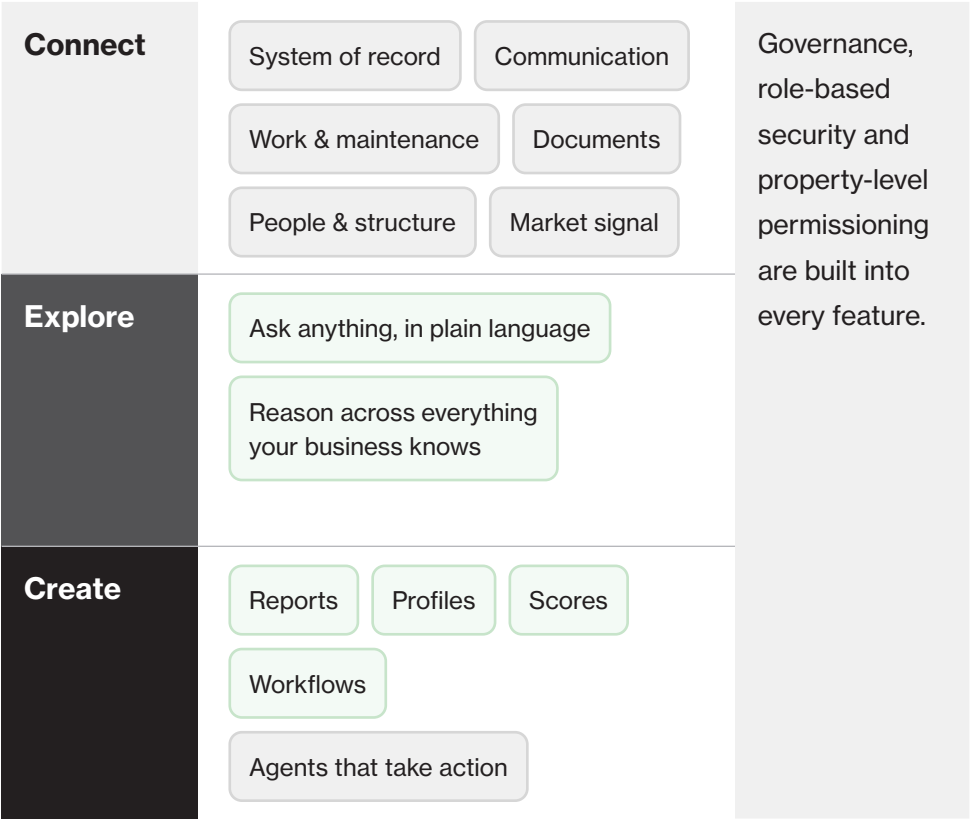
The multiplier on all three: Whether anyone uses it

A capability that is not used returns zero, however well it performs in a demo. In housing this is the quiet killer, because site teams will not open one more system and learn one more interface on a day that is already full. The difference between a workflow used by a fifth of a region and one used by four fifths is four times the return, and none of it is a technology question.

All three trace back to the same decision: buying applications instead of building your own. Point solutions embed infrastructure inside the application, which creates duplication, an API and webhook tax, and a data team asked to build ever more. The fix is in infrastructure, not another application. Build on your own knowledge and the IP comes back in-house.

The Everyday AI Platform: How you build

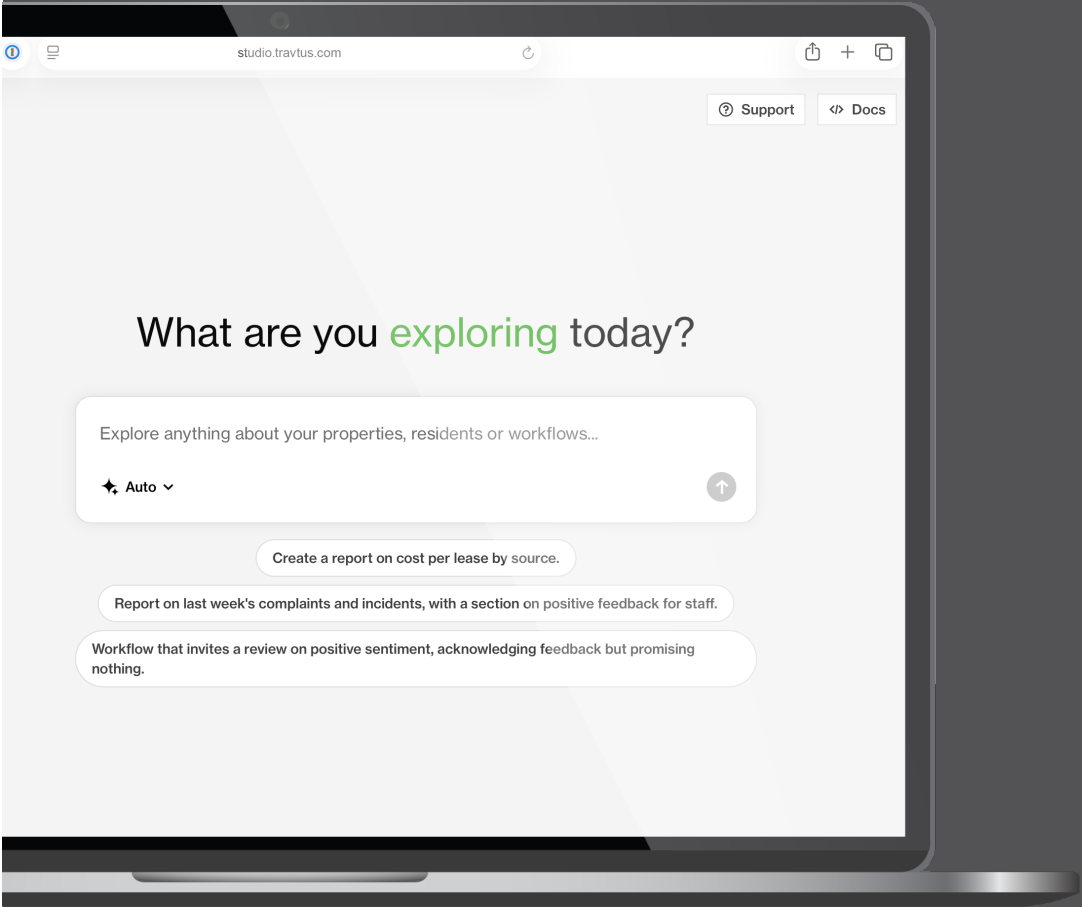
One enterprise platform rather than a pile of point tools. Three steps, two surfaces, and a single governance layer that every feature inherits by default.



Note what is not in the diagram: A separate integration project per use case, permissioning model per tool, or a separate invoice per capability. Removing those three is most of the economic argument.

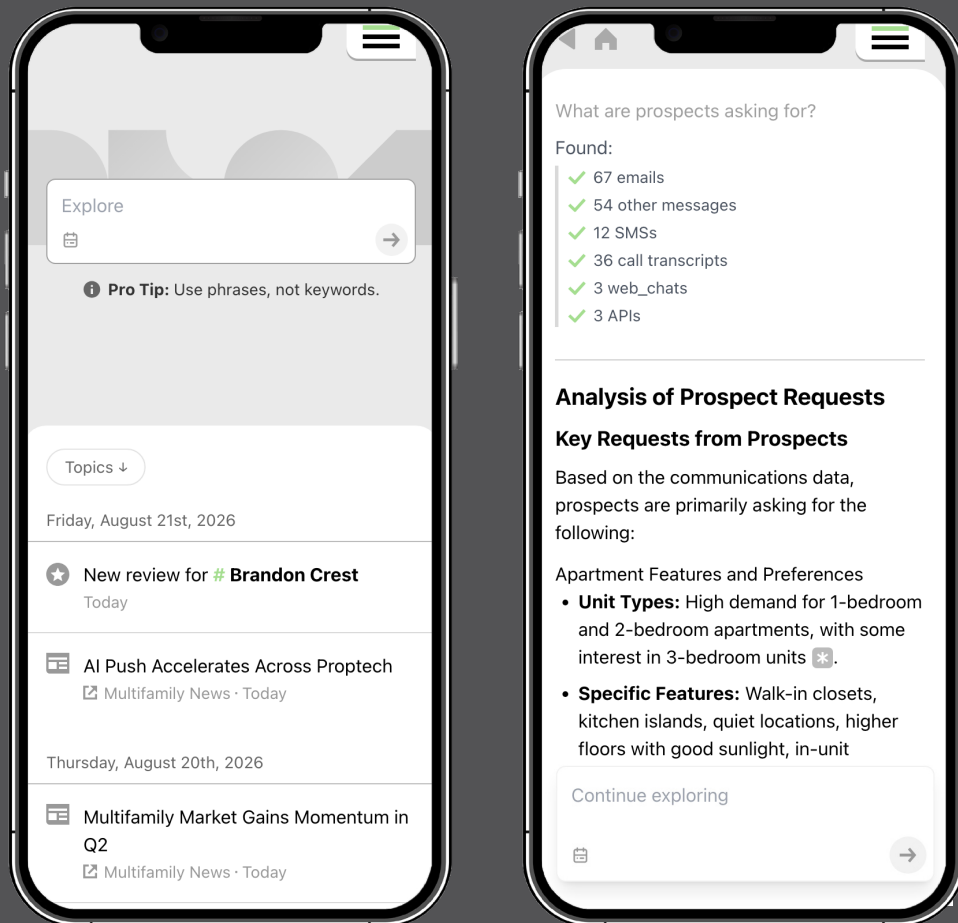
Introducing the Travtus Studio

Studio is where you build your AI and publish it to your world. Your teams bring in context, build their own features, and publish them to employees, clients and investors. No engineering required to make something new.



Introducing the Travtus Gateway

Gateway is where your world meets your AI, everyday. The everyday place people ask their own questions and use what has been published to them. This is where adoption is won or lost.



Measure it with a unit you already know

The return becomes tractable the moment you express it as cost to serve and time to value for one workflow, against a denominator your teams already report on. Pick one and hold it steady.



Cost to serve.

Fully loaded cost per unit per month for a named process.



Cycle time.

Minutes per work order from report to close.
Days per lease from inquiry to signature.



Touches.

Human interventions per renewal, per escalation, per turn.

Three numbers worth tracking from week one

01

Unit economics

Baseline cost per unit, minus new cost per unit, across the units in scope, less what the platform costs you. Run it on one workflow before you run it on the portfolio.

02

Time to value

Weeks from kickoff to the first workflow running in production. Then measure it again for the second. The delta between the two is the real signal, and it is the number most enterprises never capture.

03

Reuse rate

The share of each new use case that runs on what you have already built. Healthy looks like more than 70% by the third use case. Flat reuse means you are running projects, not building a platform.

Why a unit metric and not a percentage: A single headline return figure is easy to quote and impossible to defend, because nobody can trace it back to a decision. Cost per unit on a named workflow survives a CFO's questions, moves week to week, and tells you where to look next.

Step 1: Connect

Bring together everything your business already knows once, so nothing has to be rebuilt for the next use case.

“

How do we serve our residents better if we don't know what's bothering them? This is a good way for us to get some deep-seated insights.

MIKE GOMES
CHIEF EXPERIENCE OFFICER, CORTLAND

Buy context, not another application

25%

**lift in resident retention where
scores and profiles are in production¹**

500k+

**units under management across
enterprises building on the platform¹**

The reason the second use case costs almost as much as the first is that nothing carried over. Context is the thing that carries over, and it is the only part of an AI investment that appreciates.

An application depreciates from the day you buy it. A context layer does the opposite. Every conversation, ledger entry, work order and lease that flows into it makes every future use case cheaper to build and better at its job.

This is the practical difference between adding an AI feature and becoming AI-native. A feature is bolted onto one team's tool and leaves the shape of the organization unchanged. Connecting your context changes what is possible everywhere at once, because reasoning models can think across the business, and that only works when the business is readable to them.

It is also the commercial argument. Because you build your own, you open up use case after use case, and what your business knows becomes the advantage rather than the vendor's.

¹. Measured across Trvtus enterprise deployments.

What connecting buys you



Transparency

Leaders can see what is happening across regions without being on the ground and act on it.



Flexibility

Your teams build the reports, profiles, scores and workflows, so you can decide what matters and change it as the business changes.



Compounding

Use case five draws on everything use cases one through four previously taught the platform.

You already generate everything this needs. Connecting it is the first step of getting started, not a prerequisite you have to build first.

What “connected” means in a portfolio

Vendors use the word loosely. In practice, a housing enterprise holds six categories of context, and the value of any AI capability is capped by how many it can reason across at once.

01 The system of record

Ledgers, charges, delinquency, lease terms, move-ins and move-outs. The numbers everyone already trusts.

02 Communication

Calls, emails, texts, chat, and notes. Where intent, frustration and risk surface, and the richest source most enterprises leave unread.

03 Work & maintenance

Work orders, vendor activity, turn history, recurring faults, and unit condition over time.

04 Documents

Leases, addenda, policies, notices, inspection reports, and standard procedures. The written version of how you operate.

05 People and structure

Who manages what, which region owns which asset, who approves what. Without this, permissioning is guesswork.

06 Market signal

Comps, submarket movement, and seasonality. Public information everyone has but useful only when blended with the five above.

The test: Ask any AI vendor which of these six their product reasons across. Most say one, maybe two. That answer is your ceiling on return, because a product that sees one category can only solve problems inside it. Few housing problems stay in one.

Connecting is not a migration. The systems of record stay where they are and keep doing their job. What changes is that what they hold becomes readable by everything you build next, under the access rules you already enforce.

Step 2: Explore

Explore is where your AI learns your business. Ask it anything and teach it everything you know, so every AI you build already understands how you run.



We wanted to utilize AI to understand our data, our asset performance, our customer sentiment. And that's something we couldn't find, where an AI provider currently in the space or outside was looking at all of those areas, not just automations.

WHITNEY KIDD
SVP OF INNOVATION & TECHNOLOGY, PREISS

Find the return before you build

Most AI business cases are written before anyone has looked. The order should be the other way around: explore what the business already knows, find where the money is actually going, then build.

This step is usually skipped, and skipping it is expensive. Teams pick a use case because a vendor demoed it or a competitor announced it, not because they measured where their own cost and cycle time sit.

Once your context is connected, you can interrogate it in plain language before committing engineering time to anything. Which communities generate the most repeat contact, and about what. Where cycle time is worst and why. Which escalations were avoidable. Which regions do the same job five different ways.

These questions used to require a data project each. Now they are a conversation, which means you can afford to ask twenty of them before choosing what to build. The choosing is the highest-leverage decision in the whole program.

Exploring is also how adoption starts. When people get an answer by asking, in the surface they already work in, the platform stops being a rollout and becomes a habit. Habit is what makes the workflows you build later actually get used.

Questions worth asking first

1. Where does the same issue come back more than once?

2. Which handoffs add days but no decisions?

3. What do our best-performing communities do differently?

4. Which residents told us they were unhappy before they left?

5. Where do teams disagree about what the number means?

A note on sequencing:

Explore is a step, not a phase you finish. Teams that keep exploring after they start building find their second and third use cases faster than teams that stopped, because the questions get sharper as the context gets richer.

Step 3: Create

Create is where your ideas become AI agents. Build the reports, profiles, scores and workflows your teams use every day and publish them to your world.



In a world where property management technology tends to offer the same solution out of the box, the ability to have something more tailored really stands out.

MIKE GOMES
CHIEF EXPERIENCE OFFICER, CORTLAND

The four things you build

Almost every housing use case worth the effort resolves into one of four shapes. Learn the four and your teams can build the rest themselves.



Reports

Talk your way to the report you need.

Getting, sharing and reporting information stops being a wait. The full context of the business is built in, and anyone can build one by describing it.

Returns show in:

- Time from question to answer
- Decisions made on current rather than month-old information



Profiles

Your own cheat sheet on any resident, community, deal or asset.

A reusable summary set up to show exactly what a given job needs. Marketing, customer experience and asset management each get their own view of the same record.

Returns show in:

- Ramp time
- Escalation handling
- Every conversation starting with the full story



Scores

Find the needle in the haystack.

A buried signal, risk or sentiment or health or performance, turned into a comparable number that surfaces what needs attention, often before it becomes a problem. You decide what matters.

Returns show in:

- Retention
- Early intervention
- An always-on, read on experience instead of a survey taken once a year



Workflows

Agents that do real jobs.

Workflows are agentic. They reason and act on behalf of the business, drawing on all your interlinked context, and you build them by describing the job rather than coding it. Alerts watch for what needs attention and flag it.

Returns show in:

- Cost to serve
- Cycle time
- Consistency across regions

Build once, reuse everywhere

Most teams can build a prototype. The return is usually lost at launch, when production requirements arrive all at once and the early gains disappear into rework.

One consistent pattern changes that by settling the hard parts early, so each new capability builds on what already works rather than rediscovering it. The hard parts are always the same: grounding in trusted sources, permission-aware retrieval, human checkpoints, an audit trail, and a way to see cost and quality per workflow.

Solve them once at the platform level and they stop being project scope. A new use case inherits them. That is the mechanism behind the compounding: not that the fifth workflow is simpler than the first, but that four fifths of it already exists.

The compounding effect

Use case 1



One quarter.
Mostly foundations.

Use case 3



Weeks.
Foundations are reused.

Use case 5



Days.
Built on what exists.

 Reused  New build

A useful diagnostic:

If your second AI use case took roughly as long as your first, you do not have a platform. You have two projects that happen to share a vendor.

Match autonomy to the cost of being wrong

Every workflow needs a level of independence. That choice drives both the return and the risk, and it should be made deliberately, workflow by workflow, rather than set once for everything.



A simple rule for deciding

Ask what happens if this runs wrong a hundred times before anyone notices.

If the answer is an awkward email, let it run within policy. If the answer is a resident harmed, a regulator interested, or a decision that cannot be unwound, keep a person in the loop and design the workflow so that person has everything they need to decide in seconds rather than minutes.

Human oversight is not the opposite of leverage. A well-built approval step that takes four seconds still delivers most of the gain, and it is the thing that lets you deploy widely with confidence.

Responsible by design.

In housing, AI touches people's homes. Keep humans accountable for consequential decisions, keep every action traceable to its source, honor the access boundaries your organization already enforces, and be able to explain any output to a resident, an owner or an auditor. These are choices made at build time, and they cost far less than retrofitting them.

Set the minimum bar before you ship

Before any workflow goes live across a portfolio, six things should be true. Settle them once and the return is protected as you scale beyond the first use case.

☐

Trusted sources

Know which systems are in scope, which are explicitly out, and who confirmed it.

☐

A named owner & adoption owner

Have two people accountable for the logic and for signing off changes to it, one in the region instead of head office.

☐

Permissioning

Enforce role-based and property-level access on every answer and action.

☐

Traceability

Ensure every output can be traced back to the source it came from on demand.

☐

Human checkpoints

Have written down what this workflow may never do without a person.

☐

Cost attribution

Have usage and spend tracked to this workflow, not absorbed into one platform bill.

Settling these six is a one-week exercise the first time and a fifteen-minute exercise every time after. That ratio is the whole argument for doing it at the platform level rather than per project.

Holding the return as usage grows

Early wins become durable returns when oversight is built in rather than added. As workflows move from one region to the portfolio, three costs start to move, and only one of them is the platform.

What goes up



Volume cost.

More units, more conversations, more reasoning. Honest and predictable, and it should track your unit metric closely.



Oversight load.

More workflows means more to watch. If oversight is manual, this grows in a straight line and eventually eats the gain.

What comes down



Cost per new use case.

Falling sharply is the best single indicator the platform is working.



Integration spend.

Approaches zero, because there is nothing new to integrate.



Time to production.

Go from quarters to weeks to days.

The one to watch

Oversight load is where returns quietly die. A team running four workflows can watch them by hand. A team running forty cannot, and if the only way to know a workflow has drifted is for someone to notice a complaint, you will find out late and expensively.

This is why observability belongs in the platform rather than in a project plan. Quality, latency, volume, cost and exception rates attributed per workflow mean the fortieth workflow costs about as much to supervise as the fourth.

Three questions for a quarterly review

01. Which workflows moved the unit metric, and by how much?
02. What did the last use case cost against the first?
03. Where is adoption below half, and who owns fixing it?

Guardrails keep you moving

Controls are usually framed as a brake. In practice they are the reason you can go faster, because a capability nobody trusts never gets deployed widely enough to matter.

01 **Permissioning that mirrors your organization.**

Role-based security and property-level permissioning built into every feature rather than bolted on. A regional manager sees their region. A site team sees their community. The AI inherits the same boundaries the rest of your stack already enforces.

02 **Human accountability on consequential decisions.**

Anything affecting a resident's tenancy, standing or legal position keeps a named person in the loop, with the context they need to decide quickly. Judgment stays with people. The preparation does not.

03 **Traceability as a default.**

Every answer traceable to its source and every action to its trigger, its inputs and its approver. This is what turns "the AI said so" into something you can explain to an owner, a resident or an auditor.

04 **Governed self-service.**

Business teams build their own features and workflows inside the controls, not around them. Freedom to build and consistency of control are not in tension when the control lives in the platform.

05 **Model-agnostic by design.**

The reasoning layer is replaceable. As models improve or economics shift, you change the engine without rebuilding what sits on top of it, and without renegotiating who owns what your business knows.

06 **Your IP stays yours.**

The logic your teams encode, the scores they define, the workflows they build: assets on your balance sheet rather than configuration inside somebody else's product.

None of this makes a compliance product, and it should not be read as one. It makes a platform your legal, risk and operating leadership can say yes to, which is a precondition for the return rather than a feature of it.

Your first 90 days

The path to a defensible return runs through lowering the recurring cost of each new capability. Start narrow, prove the pattern, then let it compound.

WEEK 1

Connect the context, and baseline one workflow

Connect the sources your first workflow needs, and while you are there connect the ones the next three will need. Pick somewhere handoffs and rework are common: inquiry response, maintenance triage, renewal outreach. Measure today's cost to serve, cycle time and touchcount before you change anything. Without a baseline there is no return, only an anecdote.

WEEK 3

Explore before you commit

Ask twenty questions of your own context before choosing what to build. Confirm the workflow you picked is genuinely where the cost sits, and set permissioning and access rules now, at the platform level, so no later use case has to negotiate them again.

WEEK 5

Build it with a person approving

Ship with a person in the loop and full traceability. Resist the urge to let it run unattended early. You are buying evidence, and the approval log is the evidence.

WEEK 7

Win adoption in one region before widening

Name an owner on the ground. Get to four fifths adoption in one region rather than a fifth across five. A proven pattern in one region is a template. A thin deployment everywhere is a write-off.

WEEK 9

Build the second workflow and time it

This is the real experiment. If the second takes a fraction of the first, the platform is working and the business case writes itself. If it takes as long, find out why now rather than after use case six.

Defer multi-agent coordination, complex routing, and settling on a single model. None of these pay until you have a working pattern to reuse, and all of them are easier to get right once you do.

Next steps



Travtus is the most transformational technology we've implemented since online leasing.

ADAM BYRLEY
CHIEF OPERATING OFFICER, PREISS

Your competitors are buying AI features. The opportunity is to become AI-native, where intelligence, decisions and workflows are centralized and every team works from the same understanding of the business.

That is not a bigger version of a pilot. It is a different architecture, and the returns look different too: not a one-time saving in one department, but a falling cost for every capability you add, on knowledge and IP that stay yours.

Travtus is the everyday AI platform on which you build your own AI for housing, in a sentence. Put what your business already knows to work as the answers, workflows and actions your teams use every day.

Where to go next:

Book an executive briefing with us

Book a working session on your portfolio, your baseline and your first workflow.

See Explore on your own context

Ask your business a question and get an answer, not another dashboard.

Run your unit-metric baseline

We'll help you size the return on a workflow before you commit to anything.

travtus.com/case-studies

Read how Cortland, BH, Continental, Preiss and Beam Living built with our platform.

www.travtus.com

© 2026 Travtus. Everyday AI™ is a trademark of Travtus. Quotations are from published Travtus case studies. Figures reflect results measured across Travtus enterprise deployments and are not a guarantee of future performance.