Orium + commercetools + CONTENTSTACK

The Software Buyer's Guide to Agentic AI

What changes when software is presumed to be cheap to build, and a demo proves less than it used to.

THE POWER OF MACH

Table of Contents

Foreword

Holly Hall

Managing Director,
MACH Alliance

Every few years a new MACH Alliance resource sets out to answer the question enterprise buyers are asking, not the one that is easiest to write about. Our ebook Evaluating AI Vendors for Composable Commerce Environments answered the first of those: How do you evaluate an AI vendor on its own terms? This guide answers a harder one, raised often enough by the buyers in our community to earn its own resource.

Agentic AI has stopped being a line on a vendor slide. It shows up on both sides of the table now, in the products under review and in the process of reviewing them, and it has reopened a question most organizations thought they had settled: What should we buy, and what should we build?

We were glad to see commercetools, Contentstack, and Orium take this on together, and we reviewed it for the same thing we review everything published under the MACH name— does it hold up without a vendor's logo attached? It does. Nothing in these pages tells you what to buy. It tells you what to ask, and why the answers carry more weight than they used to.

Bring these questions into your next evaluation.

Executive Summary

Buyers now presume software is cheap to build, and that presumption, not any single product capability, is what's actually changed about procurement. It cuts both ways: a buyer who can prototype can't claim a requirement was unknowable later, and a vendor with the same tools can't defer the product-to-environment gap to a scoping exercise after signature.

Architecture, not the build decision itself, predicts whether an organization can actually use agentic AI. A legacy system that can't support agentic workloads is a second project, whether or not anyone chose to build one.

Evaluations need to get narrower and more concrete: a prototype is a procurement activity that forces internal alignment before the first vendor call, and a short paid pilot with two finalists tells you more than five demos ever will.

The product itself is the hardest new problem. It doesn't behave the same way twice, it can act rather than just answer, and it keeps changing after signature, and no one has a settled definition yet of what a trustworthy version of that looks like.

Most purchases now carry AI capability nobody asked for. Size the review to the actual risk, then put real teeth behind it: pricing at scale, change control, and liability all belong in the contract, not a conversation.

The organization has to be ready, not just the vendor. That means a bigger committee that convenes earlier, and an honest answer for whether the team you have can actually run, and own, what you bought.

Introduction

Most enterprise buyers this year are replacing something they built themselves, or something that's simply aged past repair, and the board wants to know why it can't just be rebuilt in-house with AI coding tools instead. Every vendor on the shortlist is already demoing some kind of agent, and the team drafting requirements is probably using AI to do it too.

None of that pressure existed the last time most procurement processes were actually built.

Software buying has failed the same way for years: evaluations run wide across a long vendor list and shallow on the criteria that actually matter; teams start comparing options before they've agreed internally on what they're optimizing for; and budgets cover the license while leaving out the team required to actually run the thing.

MACH Alliance research shows that among enterprise IT decision-makers, 88% report real barriers to AI implementation, hitting an average of three at once, with data privacy and security the single most-cited concern, ahead of ethical concerns, integration complexity, and a shortage of in-house skills. Misalignment between IT and the rest of the business remains one of the most common obstacles to executing a software strategy at all.

The single biggest change is the belief about what software costs to build. Leadership teams assume that cost is falling for everyone at the table, which raises what each party can be asked to show. A buyer who can prototype a requirement cannot claim, six months into implementation, that the requirement was unknowable, nor can a vendor whose delivery teams have the same tools push the gap between product and environment into a post-signature scoping exercise.

That's the biggest change to the process, but there's actually a harder change sitting underneath it: what's actually being evaluated has changed, too. An agentic product doesn't behave the same way twice. It reasons through the problem each time rather than following a fixed path, so one clean demo only shows you what happened once, not what it will reliably do. What you're actually buying is everything else it might do the rest of the time.

This guide is organized around those two pivotal shifts: why you are buying at all, what you owe before the first vendor call, and what your vendors owe you. Assessing the AI inside a product is its own body of work, and the MACH Alliance ebook Evaluating AI Vendors for Composable Commerce Environments covers it well. It doesn't cover what agentic AI does to the rest of the purchase, which is where this guide comes in.

88%

of enterprise IT decision-makers report real barriers to AI implementation

3

average number of barriers hit at once — this isn't a single obstacle, it's a pile-up

№1 concern

Data privacy & security, ahead of ethical concerns, integration complexity, and in-house skill shortages

What You Are Buying

Your board is asking about building, and the question deserves a real answer.

Someone on your executive team has watched a working prototype appear in an afternoon and has drawn a reasonable inference: *If that's what a small team can do now, why are we writing a seven-figure check?*

That pressure to build, arriving from above, will cost you credibility with the people asking if you dismiss it as hype. So let's take apart that inference carefully, because two different things are being conflated.

The first is the speed of producing code that runs. That has changed, visibly, and it's what most of the public conversation is about.

The second is the practice of delivering software that an enterprise can depend on: code review, test coverage, security hardening, release discipline, observability, someone on call at three in the morning. That practice is maturing around agent-driven delivery, and the maturation is real. It is also considerably less far along than the first thing, and it's the one that determines whether you can replace a vendor.

Most organizations are in the first two years of building agentic applications at any scale, and the field has not standardized around what good practice looks like. Speed-of-first-draft has improved faster than the discipline required to operate the result. The honest answer to your board, therefore, is that the presumption is directionally right, but the timeline is wrong. Saying that plainly is step one, but being able to demonstrate where building is the better call is step two, and that's what the rest of this section is for.

> "The prototype is never the hard part anymore. The hard part is who's on call for it in eighteen months, and whether the team that built it fast is the same team responsible for keeping it running.

Everett Zufelt
VP Agentic Systems, Orium

What "build" means, and where it is the safer answer

"Build" covers at least three different situations:

01
Point Solution
USUALLY SAFE TO BUILD

02
Scaled Capability
NEEDS REAL MATURITY

03
Vendor Extension
CASE-BY-CASE

Each of those circumstances deserves its own answer. Point solutions are often safe to build in-house. Things like testing tools, validation utilities, and dashboarding are already working well for teams that have tried it. By contrast, scaled capabilities for the whole org—such as loyalty logic or search and recommendations—are really only feasible for teams with real maturity behind them. Common failure points here tend to be specific, and include things like an underestimated enterprise

data strategy or the difficulty of hitting performance at scale. The third category—small extensions on existing vendor capabilities—are unique enough each time out that a blanket rule won't work here and they'll need case-by-case evaluation. The more useful question isn't typically whether the team can build it, but whether or not it needs to be built in the first place.

"Buy" remains close to the only responsible option when the thing in question is a core transactional or system-of-record platform (commerce, DXP, ERP, payments); externally facing and outside your own network perimeter (WAF, CDN); something the rest of the business depends on for stability; or something that touches regulated or sensitive customer data.

The reasoning behind that has held for a long time: in-house teams have lacked the breadth of hard-won experience that comes from solving the same problem across hundreds of implementations, the kind of knowledge that shows up in security hardening, scaling under real load, and integration quality that only reveals its gaps once something is live. AI-generated code and infrastructure have narrowed that gap unevenly. They have not closed it.

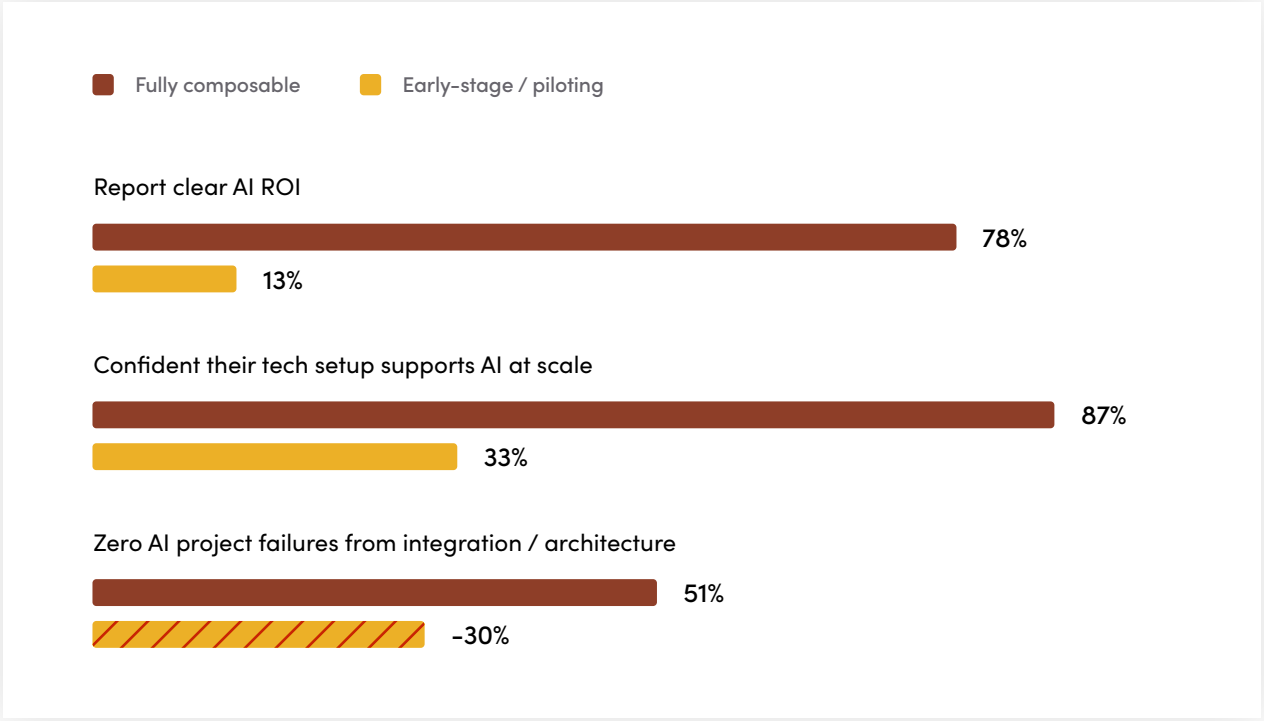
A quick pressure test to help you determine buy vs build: Does the thing in question touch core stability, sensitive data, or the outside world? A yes to any of the three is a strong signal to buy. A no to all three, paired with a team that has the maturity to support it, is where building becomes the more efficient path.

> "Preparedness at machine speed starts where software and infrastructure are built, not where breaches are caught.
>
> Alex Shamsutdinov
> VP CISO at Upwork, AIUC-1's After Mythos

Your existing foundation narrows the decision

The architecture you already have shapes this decision more than most teams expect. MACH Alliance research shows organizations with mature composable architectures are six times more likely to report clear AI ROI than those still in early planning stages, 78% versus 13%. Confidence that a team's tech setup can support AI at scale follows the same pattern: 87% among organizations that have fully implemented composable, against 33% among those still just planning it. Fully implemented organizations also report a significant acceleration in AI deployment, with 94% seeing faster implementation as a result.

Read that correlation carefully before you take it into a steering committee. Organizations with mature composable architectures tend to be further along



on funding, staffing, and executive attention, and any of those could be doing some of the work the architecture appears to be doing. The pattern is consistent enough across measures to take seriously, but it's not clean enough to present as proof that architecture alone causes the outcome, and a skeptical CFO will find that seam before you do.

What the data supports is narrower and still useful: a legacy system that cannot support agentic workloads adds a second project to the first one, and that second project is rarely in the budget. That shows up directly in failure rates, too: 51% of fully composable organizations report zero AI project failures caused by integration or architecture problems, against roughly three in ten among organizations still piloting or in the process of adopting it.

Take a deprecated, on-premise commerce platform and ask what it would actually take for an agent to operate against it, like calling an interface to create

> "The gap always shows up in the same place: checkout, pricing, and inventory edge cases that only surface after years of production traffic across thousands of businesses. A prototype can look finished long before it's actually ready to run a real storefront.

Preseetha Pettigrew
VP Global Partnerships, Contentstack

promotions or pulling in enriched product data from other systems in real time. What's missing is rarely a single feature and it's usually foundational: modern API performance, cloud-native scaling, and the patterns that let a system integrate cleanly with anything built after it. Reaching parity with an actively developed alternative that way takes real time, against a target that keeps moving.

Most buyers are not choosing between a clean legacy system and a clean composable one. If you sit somewhere in the middle, the answers to these three questions take you further than a replatforming debate.

1

Which capability do you expose first?

Usually, that means starting with something that only looks up information rather than changes it: checking inventory, pulling an order status, that kind of thing. It can't do damage, since nothing gets altered if the agent gets it wrong, but it lets you confirm the connection actually works before you ever trust it to create or change something for real.

2

Where does the boundary go?

Somewhere you control, in front of the legacy system and not inside it, so the interface you publish outlives whatever sits behind it. That layer becomes the thing you maintain and the thing every vendor integrates against.

3

What sits in front of the boundary?

New capabilities, agent-facing interfaces, and anything you expect to change. Behind it stays the system of record you are not ready to move, on a migration timeline you control.

The AI surface you did not ask for

And now we get to the situation almost no procurement process is built for, and almost every process is facing today. You're buying an order management system or a commerce or DXP platform. AI is not the reason for the purchase—it's not in your requirements document, and no one on the evaluation team asked for it—but every vendor on the shortlist ships it anyway.

That capability now has to be assessed, *whether or not you intend to turn it on*. Someone owns the question of what data it reaches. Someone prices it, including what happens to the price as usage grows.

First, separate the capability you are buying from the capability that came along with it, scoring the vendor on the thing you set out to buy and assessing the AI surface separately, without letting a weak agent disqualify a strong platform solution or vice versa.

Next, establish whether it can be turned off—and what you lose if it is—getting the answer in writing since "you can just not use it" and "it's on by default and calls out to a model provider" are frequently the same product described by two different people at the same vendor.

And finally, decide once, centrally, what your standing requirements are, so that a purchase that now always carries an AI review doesn't mean re-litigating that review from scratch every time.

What You Owe

Prototype to align before you prototype to compare

The second failure mode from this guide's opening list is the expensive one: teams compare options before they've agreed internally on what they're optimizing for. Vendor conversations then become the venue for an internal disagreement, which is the most costly place to have one.

A working prototype fixes this, and not for the reason people usually give. Its value is less about testing a vendor than about forcing your own organization to be specific. When merchandising, engineering, and finance look at something running, disagreements that survived three requirements workshops surface in ten minutes. The prototype is therefore a requirements artifact that you ultimately throw away.

This distinction matters because it keeps the build-versus-buy question from contaminating the evaluation. Prototyping to align is a procurement activity. It doesn't commit you to building anything, it doesn't need to scale, and it shouldn't be staffed as though it were a product. Teams that miss this either refuse to prototype because "we decided to buy," or accidentally talk themselves into a build because the prototype looked good on a laptop.

Do it in this order:

1. First, prototype internally to establish what you actually require and where your teams disagree.
2. Next, take the resolved version into vendor conversations as a specification.
3. And finally, ask vendors to demonstrate against it, on your data.

That third step is where the burden shifts to them, and it's worth arriving with something concrete enough to make a generic demo impossible.

This ordering doesn't put vendors at the end of a queue. Internal alignment comes first because it's yours to do, and nobody else can do it for you. Solution design still happens with your finalists, jointly, working against the specification you arrive with. What stops is using vendor conversations to discover what you wanted in the first place.

What your own AI put in the requirements

If your team drafted the requirements with AI, that draft needs the same scrutiny you're about to apply to a vendor.

Invention is the visible failure: a requirement reads plausibly, nobody in the business ever actually asked for it, and you pay three vendors to price it anyway. Omission is the expensive one, because a model writes what's common across its training, not what's specific to your operation. The integration nobody documented, the regulation that applies only in one market, the workflow your warehouse built around a limitation twelve years ago— none of it appears unless a person puts it there.

The same caution applies to scoring vendor responses with AI. A model comparing three proposals tends to reward the one written most fluently, which measures the vendor's writing, not their fit.

To handle these likely failure points, implement two controls: have a named person from each affected function confirm that every requirement traces back to something real, then run a separate session to identify what the document is missing. The prototype above surfaces both faster than another review cycle would.

The sequence collapses

Buying used to run as two sequential decisions: choose the software, then separately choose who implements it. That sequence existed because building something real to test a hypothesis used to be too slow and too expensive to do before a contract existed. Once real work becomes cheap enough to produce during an evaluation, the two decisions no longer need to happen one after the other.

The Sequence, Then and Now

	THEN		NOW
THE DECISION Software and implementation partner	Evaluated one after the other, from two separate blank pages	→	Evaluated together, since real work can exist before either decision is final
THE PROOF POINT What a buyer sees before signing	A scored RFP and a set of requirement documents	→	A working prototype, built against the buyer's own data and constraints
THE EXPOSURE Who carries the risk of a wrong estimate	The buyer, once work starts and the real requirements surface	→	Both sides, earlier, while the cost of being wrong is still small

Just over half of organizations report getting from idea to production in three to six months, up slightly from the year before. Treat that as a signal about the market, not a target for your own environment. A prototype on synthetic data can be assembled in days; a prototype on production customer data in a regulated enterprise involves a data-sharing agreement, a security review, likely a privacy impact assessment, and a non-production environment realistic enough to mean something— you need to budget for that directly. To move quickly through this step, you need to build the pathway before you actually need it, with legal and security agreeing in advance on what a vendor evaluation is allowed to touch.

Moving fast cuts in both directions, which is the actual point: a pilot that can be stood up quickly can be shut down just as quickly. Walmart's experience with OpenAI's Instant Checkout, widely reported in the press, is a visible example. The retailer joined as an early partner when the feature launched in 2025. Within a few months, live traffic showed checkout completed on Walmart's own site, converting ahead of checkout completed natively inside the AI product, and Walmart stepped back from the integration around the same period OpenAI moved away from native in-app checkout more broadly.

The takeaway here isn't a failure of agentic capability, but two important lessons sit in that example— and the second is the one usually missed. A live pilot against real traffic surfaced something no demo would have. And a partner deprecated a capability a buyer had already built against, on the partner's own timeline, not the buyer's. Build that same willingness to test, measure, and reverse into your own evaluation. Then ask every vendor what happens if they change direction, because that risk never shows up in a demo.

Who belongs in the room

The buying committee needs to get bigger, and it needs a perspective from outside your own vendor conversations.

On who: Someone needs a view of this technology that isn't tied to a single vendor's pitch. Roadmaps and market claims move faster than a team can track when it only runs an evaluation like this once every few years. That can be an internal role dedicated to staying current, or outside expertise brought in for the purpose. What matters is that the person has watched this play out across more than one organization.

On when: executive sponsorship needs to be in the room from the start. Introducing it at sign-off is too late; by then, governance is rubber-stamping a decision that's already been made rather than shaping it.

The reason for both is the same. These purchases touch more of the business than a comparable purchase did five years ago, and change management has become part of the decision itself, not something addressed after the contract is signed. A committee sized for a narrower kind of purchase, one where governance could wait, is the wrong shape for what's actually being evaluated now.

The Committee, Then and Now

	THEN		NOW
THE DECISION-MAKERS			
Who sits on the committee.	Day-to-day users, a security reviewer, IT for architecture	→	The same, plus whoever owns change management, plus a perspective not tied to one vendor
THE PROOF POINT			
When change management enters.	After signing, as a downstream phase	→	During the evaluation, as part of the decision itself

What the evaluation itself costs

The third failure mode from this guide's opening list is a budget that covers the license and omits everything around it. An evaluation run the way this section describes carries costs of its own, and they belong in the same budget, not a surprise found mid-cycle. Earmark budget for:

- A short paid pilot with two finalists
- Legal and security time for a data-sharing agreement and a privacy assessment
- A non-production environment holding data realistic enough to actually prove something
- Review of the AI capability bundled into a purchase scoped for something else
- Outside expertise, if the perspective needed isn't already in-house

None of that is large against the price of the software itself, but all of it is large against a line item of zero, which is what it usually gets. Put a number on it before the evaluation starts. The alternative is finding that number mid-cycle, and cutting the verification budget to pay for it.

What Your Vendors Owe

The product is the harder case.

Agentic AI can be a tool that helps you buy well, and it can be the product under evaluation. The second case is harder, for three reasons that don't arise with traditional software.

First and foremost, it doesn't behave the same way twice. The same input can produce a different output, because the system reasons through steps, and the steps vary. For traditional software, a demo that goes well is reasonably stable proof the product does what it says. Here, it's one observation, and the vendor chose which one you saw.

Secondly, it doesn't just answer, it acts. A traditional tool tells you something and waits but an agent can place the order or send the message or update the record. The evaluation question stops being whether the output is correct and becomes what happens when it's wrong, and who's watching when it acts. (CSA CEO Jim Reavis has proposed a six-level autonomy taxonomy running from no autonomy through full self-direction, worth a look for anyone wanting more granularity than a simple assist-versus-act split).

And finally, it's harder because what you buy keeps changing after you buy it. The underlying model, the tools it can call, and the guardrails around it can all shift after signature, sometimes with no version number attached to the change. Evaluating once, at the point of purchase, tells you little about what the system will be doing in a year.

No single definition is settled yet, though frameworks are starting to emerge—AIUC-1's six control areas among them—alongside what the MACH ebook covers in the sections ahead. That's why the rest of this section is about *method* more than *criteria*: how much to look, where to look, and what to write down.

> "The hardest question we get from buyers isn't whether the AI can do something. It's whether it will still be doing the right thing, in the buyer's own brand voice, three releases from now. That's a different kind of proof than a demo can offer.
>
> Preseetha Pettigrew
> VP Global Partnerships, Contentstack

Telling real from demoware

The same tools that create genuine agentic capability make a convincing and unsubstantiated demo cheap to produce. Verification has to work differently as a result.

From here on out, references must be current, meaning any reference the vendor offers must be from the last 12 months. At the pace this category moves, a reference call about what a vendor delivered eighteen months ago says little about what's being sold now.

Depth over breadth also matters more now. Verifying one vendor properly means testing the same request repeatedly and checking references against what's being sold today, not last year. Spread across five vendors, that thins to watching five demos. Concentrated on two, it can stretch to a short paid pilot with real data— realistic with two finalists, close to impossible with five. If policy requires a minimum number of bidders, score the long list cheaply on documentation, published interfaces, and pricing, and save the real verification for the two that matter.

It's more important than ever to ask the right questions, and that means operational questions over aspirational ones. "Can it do this?" has lost most of its value; "Show me it doing this, on our data, with

our constraints, today" is the version that actually produces information.

Know the data path, too. A system that can only be operated through its own screens can still look impressive in a demo. The real test is whether its data interfaces are programmatic and standards-based enough for an agent, yours or theirs, to act through. Displaying data is a much lower bar.

One more thing worth checking, since vendors rarely bring it up unprompted: a demo where the AI suggests something and a person clicks "approve" is a completely different capability from one where the AI just does it, no click required. Ask directly which one you're actually looking at.

Closing the gap belongs in the evaluation

If software is becoming less expensive to build, that presumption applies to your vendors as well, and it impacts what you can reasonably ask of them.

The gap between what a product does out of the box and what your organization actually requires has traditionally been handled after signature, as a scoping exercise, with the cost landing on the buyer once the real requirements surfaced. That arrangement made sense when producing working software was slow. It makes much less sense now.

Ask vendors to demonstrate the gap-closing path during the evaluation. Not to describe it, and not to price it, but to actually show it working against your specification.

Be precise about what you're asking for, since the obvious version of this request produces a bad outcome. Asking a vendor to write custom code faster gets you forked, unsupportable customization, which is the thing composable architecture exists to prevent. Vendor gap-closing is gated by product roadmap and multi-tenancy far more than by typing speed, and no amount of AI assistance changes that.

The defensible version asks about extension rather than modification. First, show the extension points, running, with your requirement implemented through them. Next, show what your own team

> "If a vendor can't show you the extension points running, live, against your own data, that's the tell. The commitment has to be an open architecture a buyer's own team can build on, not a roadmap promise.

Dirk Hoerig
Founder and Chief Innovation Officer, commercetools

could build in the vendor's environment without waiting on their roadmap, asking specifically about runtime access. And finally, show which parts of your requirement the product will never cover, and say so early.

That third answer is the most valuable one you can get, and the willingness to give it is a strong signal. A vendor who names the boundary in an evaluation is telling you something a vendor who defers everything to implementation is concealing.

How much scrutiny does this purchase warrant

Not every AI surface deserves the same review. Treating them alike is how a purchase you actually needed ends up stuck while people argue about how thorough to be. The size of it should match the risk, not a fixed checklist.

Five questions set the scale:

1. Can it act, or does it only suggest? A system that writes to a record is a different proposition from one that drafts something a person sends.
2. Whose data does it reach, and does any of that data leave your perimeter?
3. Acting as whom: what credential does the agent hold, what's that credential scoped to do, and how does it get revoked?
4. What's the blast radius of a wrong action, and how long before someone notices?
5. Can you disable it and still get the product you came for?

Those answers sort most purchases into three tiers.

> "Not every AI surface touching your content needs the same scrutiny, but the ones that can publish under your brand without a human checking first absolutely do. That's usually where buyers under-scope the review

Preseetha Pettigrew,
VP Global Partnerships, Contentstack

Sizing the Review

TIER	WHEN IT APPLIES	WHAT THE REVIEW LOOKS LIKE
Record and move on	Suggests only, no sensitive data, cleanly disabled	Document what the capability does, confirm in writing that it can be turned off, note it in the file
Standard review	Acts inside bounded workflows, reaches internal data	Run the full product assessment, bring security in, get the answers in writing before signature
Review and pilot	Acts on customer or financial data, hard to reverse, cannot be disabled	Everything above, plus a short paid pilot on your own data and the contract terms below

Two things decide whether this survives contact with your process. The first is where it sits: set the tier at shortlist, once you know which vendors ship what, and run a standard review before finalist selection, rather than leaving it until pre-contract, which puts an open-ended question in front of a signature date. And the second is who calls it: one named owner, sitting outside the team that wants the purchase, since sizing exposure is a judgment call, and the people who need the software are the wrong people to size their own.

The common failure here isn't too little review. It's letting a needed purchase sit still while people argue about how much review it deserves.

Where the product assessment itself belongs

This guide is about the purchase. Assessing the AI inside a product is a separate body of work, and the Alliance has already published it.

Evaluating AI Vendors for Composable Commerce Environments was developed by MACH Alliance Ambassadors Danielle Diliberti, Dylan Valade, and Krishna Gangu, written by and for end-user brands. It sets out six evaluation areas covering architecture, data behavior, transparency, change management, support, and governance, alongside use cases, red flags, and a decision framework for internal use. Use it as the assessment instrument.

What this guide adds is where that assessment sits in a cycle that was never designed to include it: who runs it, at what threshold, and when. Run it twice. Once during the evaluation, and again as a standing requirement, because the thing you assessed will not stay still.

There is a reason to converge on a single instrument. Nearly nine in ten enterprise IT leaders believe standards and certifications for AI in composable, agentic environments are still missing, most acutely around governance, interoperability, and explainability. Until that changes, buyers carry the gap themselves. Working from a shared assessment and asking vendors what they are doing to help build an emerging standard gets further than every buyer maintaining a private checklist.

What belongs in the contract

A product assessment tells you what a thing does today. It won't settle what happens when the deal changes, the model changes, or something goes wrong. Those are commercial questions, not product ones.

Commercial exposure

Is pricing tied to a metric that stays stable, or one likely to move sharply as usage grows? A model that looks reasonable today can produce a very different number once usage triples.

Ask the vendor to model cost at 3x and 10x current usage.

Change control

A product that changes after signature, sometimes without a version number attached, requires commitments around that change.

- What notice do you get when the underlying model changes?
- Can you pin a version, and for how long?
- What's the deprecation policy for a capability you've built against, and what does the exit path look like if the vendor changes direction?

→ See: Walmart's experience with Instant Checkout, above

Liability for agent actions

When an agent acts incorrectly, whose responsibility is it? The vendor's, yours, or unresolved? This isn't hypothetical, though it's also not agentic-specific

€35M

or 7% of global turnover — max fine for non-compliance

Dec 2027

full enforcement begins — obligation is binding now

The EU AI Act classifies high-stakes financial decisions — credit scoring, insurance risk pricing — as high-risk, requiring human oversight and detailed logging. Under Article 2, the Act applies to any provider or deployer whose system is placed on the EU market or whose output is used within it, regardless of where the company is based.

When nobody will sign

All of the above assumes leverage you may not have. On the model-change notice in particular, every finalist may decline, since none of them controls the model provider. That answer is honest, not evasive, and pressing harder won't produce a commitment the vendor can't keep.

When a commitment is unavailable, buy differently, and don't proceed as though you got it. Narrow the initial scope to workflows you can afford to have change underneath you. Shorten the term, so renewal becomes your leverage. Stage the rollout behind observed behavior, and hold the autonomy down until you've seen enough. And take exit rights over assurances: the right to leave with your data on a defined timeline is worth more than a promise about a model that nobody can audit.

A vendor who tells you plainly what they can't commit to, and why, has given you more to work with than one who agrees to everything.

The Last Mile

A vendor can clear every question in this guide and the purchase can still fail, because the organization wasn't ready to manage it.

This closes the loop back to where the guide started: not the product, but the organization buying it. Evaluating whether a platform can do something is a different question from evaluating whether an organization can actually run it.

Both questions need to be answered before you invest in any solutions, tools, or products. Up to this point, the focus has been on external evaluation: does it deliver on the promises of the marketing team, or will it crash and burn when the first edge case creeps up? Now the focus shifts inwards: is your organization capable of managing a tool that passes the evaluations you just put it through?

“

Most organizations can answer who built a system. Far fewer can answer who owns it a year later, especially once the person who built it has moved to the next project. That answer needs to exist before signature, not get improvised after the first incident.

Everett Zufelt
VP Agentic Systems, Orium

This can be the toughest part of the evaluation for some orgs, because it requires honesty, clarity, and sometimes a bit of self-directed tough love. But understanding your own organization's capabilities and limitations up front will save enormous headaches and heartaches down the line. So take the time to reflect on these questions, discuss with other stakeholders internally, and go from there.

Five questions, in the order they tend to bite:

1

Is your data fit for an agent to act on?

An agent working from poor data doesn't produce a poor report, it produces poor actions, at machine speed, across your catalog or your customer base. Ask what the vendor's system does when data is wrong or missing: does it degrade visibly, or fail silently?

2

Who owns the agent in production?

Not who *owns* the platform, but who's accountable when it takes a wrong action at two in the morning? Who gets paged? And what are they authorized to do? Settle this before signature; it's the difference between an incident and a crisis.

3

How will you know it's misbehaving before a customer tells you?

A non-deterministic system needs behavioral observability, not just uptime monitoring. What gets logged, what's reviewable, and can you actually reconstruct why the agent did what it did?

4

Can the team run it as it exists today, or does the plan depend on hiring and upskilling that hasn't happened yet?

If upskilling is the plan, it's worth knowing where that actually ranks: organizations that successfully accelerate AI adoption prioritize alignment on change management, data quality, and upskilling staff ahead of simply buying more tooling—all three outrank additional tooling as investment priorities.

5

How does the vendor support you through their own pace of change?

Onboarding matters less than what happens in month eighteen. Ask whether ongoing adoption looks like a partnership or a recurring upsell.

A platform that outpaces the organization buying it is not a win. Evaluate the technology and your capacity to use it well together, not one after the other.

Close

None of this is a reason to slow down. It is a reason to be specific about what is actually being evaluated, and honest about what is genuinely new because of agentic AI versus what has simply been true about buying software all along.

The questions in this guide will not produce a single right answer for every organization. They are meant to structure a conversation that is currently happening without much shared language. Bring them into the next evaluation, adapt them to the purchase in front of you, and treat a vendor's willingness to engage with them seriously as a signal in itself.

Glossary

Agentic AI

AI systems built to plan, decide, and take action toward a goal with limited human input at each step — distinct from generative AI that primarily produces content for a human to review and act on.

AI Agent

A specialized AI system with a defined role, context, and objective, able to independently plan, reason, and complete tasks using data and API access, including interaction with other agents.

AIUC-1

An independent standard for evaluating AI agents against six control areas: data & privacy, security, safety, reliability, accountability, and society.

Assistive vs. autonomous

Assistive AI supports a human who makes the final decision and takes the final action. Autonomous AI takes the action itself. Most agentic AI on the market today is closer to assistive than truly autonomous— a distinction worth testing for directly rather than assuming.

Buying committee

The full group of stakeholders involved in a purchase decision— traditionally IT, security, and end users, now expected to include whoever owns change management for the purchase.

Composable architecture

A system built from independently replaceable components connected through open interfaces, rather than one vendor's single, all-in-one platform.

Demoware / vaporware

A capability that looks real in a controlled demonstration but isn't yet reliable, available, or built for production use.

Human-in-the-loop

A design principle requiring a person to review or approve an AI system's action before it takes effect, as opposed to fully autonomous execution.

MACH

Microservices, API-first, Cloud-native, Headless— an architectural approach built for flexibility and speed of change, as opposed to monolithic, tightly coupled systems.

Non-human identity

A way of authenticating and governing an AI agent's actions distinctly from a human user's, so systems can tell the difference between a person and an agent acting on that person's behalf.

Prompt injection

An attack that manipulates an AI system's instructions, often through content it processes, to make it act outside its intended boundaries.

Runtime access

The degree to which a buyer's own team can build on, extend, or operate within a vendor's AI or agentic environment, as opposed to being limited to what the vendor exposes.

SPONSOR

The MACH Alliance is the global industry body for open, composable, and connected enterprise technology – the foundation and the framework for the agentic era.

PARTNER

commercetools, the global digital commerce AI company, helps enterprises transform commerce for the AI era. Built API-native from the start and AI-first by design, our technology enables agentic shopping experiences, automates commerce operations and gives businesses the agility to modernize faster, innovate continuously and operate more intelligently.

PARTNER

Contentstack is redefining how modern digital experiences are built and managed. As the pioneer of the Agentic Experience Platform (AXP), Contentstack brings together structured content and brand governance (Content Cloud), real-time customer data, omnichannel personalization (Data Cloud) and autonomous AI orchestration (Agent OS) into one unified system.

PARTNER

Orium is the trusted operations and transformation partner for the autonomous commercial era—helping enterprise brands build and operate the composable, AI-ready commerce, content, and data systems that make increasingly autonomous execution possible. We work alongside clients not just to implement these systems but to run and evolve them, keeping people setting direction while agents extend what the organization can execute.


Orium