MONTE CARLO

The Ultimate Guide to Data Products

How to design and ship data products that are ready for agentic AI

Contents

01 Introduction

The data product has never been a fixed idea. Rather, it has been steadily expanding since the day it was coined as a term, and AI has accelerated that expansion faster than definitions can keep up with.

Data is the foundation businesses now run on. The data ecosystem stopped being an afterthought some years ago and became an essential part of the technical architecture. This is roughly when the idea of a "data product" arrived to help businesses solve a specific and recognizable problem: the most important data assets in the company were being built without real discipline, certainly with none of the discipline applied to software that's used by a paying customer. For a long time, data products had no named owner, no service level agreement, and no way to, for example, differentiate between a trustworthy table and an abandoned one.

The fix, then, was to borrow the product playbook. Teams assigned ownership, wrote SLAs, and documented terms and definitions. This helped them certify what data was considered trustworthy, and organizations began to treat the executive dashboard and the churn model with similar rigor as software shipped externally. That approach has proven effective.

The definition evolved

However, it has not held still. Data products started as dashboards and curated tables. But soon, machine learning models joined the category, and then the data mesh conversation pushed the concept toward domain ownership and self-service. Then, without much announcement, the category absorbed a set of assets data teams had never classified as products: retrieval indexes, semantic layers, feature stores, the document collections that an AI assistant searches. These are business-critical assets with real consumers and real failure modes, and they have become mission-critical in the AI transformation era. In most organizations, however, they have no owner and no SLA.

In the agentic AI world, three major evolutions are taking place with regards to how we think about and treat data products:

What counts as a product	Who does the consuming	What failure looks like
The boundary moved past the warehouse. If an agent retrieves from it, it is part of the product, including the index, the document store, and the tool response, whether or not anyone has called it one.	The discipline assumed a human reader who would notice something odd and ask about it. Increasingly the consumer is software that retrieves, reasons, and acts without pausing to wonder.	A broken pipeline throws an error. A data product feeding a broken agent returns a fluent, confident, plausible answer instead, and keeps returning it.

The data consumer has changed

Go back and read the fine print of any data product SLA written in the last five years. It sets freshness targets, uptime, and quality thresholds, and it was drafted with a specific picture in mind: a human analyst opening a dashboard, seeing an error, and filing a ticket. The SLA was a promise and the human was the enforcement mechanism. That is an old model that has changed completely as data consumers switch from humans to agents, changing the entire trust infrastructure we've worked hard to build to ensure data integrity over the last number of years.

> **THE BIG SHIFT**
>
> A data product consumed by a human has a built-in error detector. A data product consumed by an agent does not. Remove the human and the SLA becomes a static document that goes unnoticed, right up until a customer checks it for you.

Agents do not sanity-check. They retrieve, reason, and act with total confidence in whatever they were handed. A stale table that would have triggered a human response and investigation now produces a wrong answer to a customer, a mispriced quote, or an incorrectly routed claim, at machine speed and machine volume. Data problems get propagated all the way to the end user because agents cannot effectively pick up when something is "off" in the way that humans always could. Add to that the multiplicative effect of their speed and the interactions and handoffs between multiple agents, and the risk of producing bad outputs that customers notice becomes very high.

52% of AI practitioners find out about production issues through customer or user complaints rather than through their own monitoring.

Agents in Production: The Builder's Perspective, 2026 · n=260

This evolution continues

It would be convenient to describe this as a transition with an endpoint. Data products were built for people; now they are built for agents; adjust once and you are done. That is not what is happening. Agent autonomy is increasing, the range of systems agents can reach is widening, and the assets they treat as inputs keep multiplying. Anyone offering a settled definition of a data product right now is describing a snapshot.

That argues for building on the parts that have survived every previous shift. Ownership, certification, documented meaning, and measurable reliability have outlasted the data warehouse, the data lake, the data mesh, and the first wave of AI. They will outlast this one. What changes, now, is the standard each has to meet.

That is exactly what this guide covers: the discipline of building data products for consumers that act on them directly. Ownership, certification, documented meaning, and measurable reliability are each held to the standard an autonomous consumer demands, alongside the two requirements

that standard makes non-negotiable: context and lineage.

02 What a data product is

A short definition, a wide scope, and a standard higher than most organizations currently apply.

> **DEFINITION**
>
> A data product is a critical data asset: a key dashboard, a machine learning model, a curated table, or a retrieval index, developed with the same reliable processes and architecture you would use to build a product for an external customer.

Note the scope that definition covers. A vector index your support agent retrieves from is a data product. As is the semantic layer your text-to-SQL agent resolves metrics against, as well as the feature store feeding a pricing model. These are business-critical assets with consumers, dependencies, and failure modes, which means they need owners and SLAs like anything else. In most organizations they have neither.

Benefits of developing an AI-ready framework for data products

Trust, which is now critical	A way to measure the data team	Focus and capacity
Stakeholders trust data until something goes wrong, and rarely fully afterward. When an agent sits on top, that trust is not just reputational; it determines whether you are permitted to automate the workflow at all.	Incidents, time to detection, time to resolution, downtime and its cost. Hard numbers for work that is otherwise invisible to the business, and the basis of any credible AI readiness conversation.	Data engineers have historically absorbed a large share of quality work by hand. The product framework automates the tedious part so that time goes to work that moves the business.

The capacity argument deserves a callout in 2026. The reason to automate quality work is not that engineers dislike it, but rather that manual review does not survive contact with agent volume, and most organizations are about to discover this the hard way.

58% of AI executives expect agent counts to increase substantially or more within twelve months, yet **62%** still rely on human review as their primary means of verifying AI outputs.

State of AI Reliability, CDO Magazine x Monte Carlo, 2026

Where the real work begins

You cannot decide that an executive dashboard is a data product any more than you can declare bankruptcy by announcing it. The shift requires actual changes to governance and reliability: ownership someone acknowledged, quality someone verifies, meaning someone documented. The standard is unforgiving, because there is now a consumer that will act on the gap between the label and the reality.

03 Designing for a non-human consumer

Seven design steps for a consumer that cannot ask a follow-up question.

01 **Write SLAs that assume no one is watching**
Shared language, shared metrics, documented expectations, and an SLA that is machine-checkable and actively monitored, because no analyst is going to notice the breach. An unmonitored SLA is a comment, not a control.

02 **Assign ownership at every level**
Organizational, project, and pipeline or table level, extended to retrieval sources and indexes, with a record of which agents depend on each, so an upstream problem reaches the agent's owner and not only the data engineer's queue.

03 **Document for a reader with no context**
Documentation carries discovery and self-service, and agents need it at a higher standard than people do: explicit metric definitions, synonyms, relationships, and worked example queries. An agent cannot infer that ARR excludes one-time fees.

04 **Factor in compliance and governance**
External compliance, internal governance, and one question that decides whether either is real: which agents may access this product, under what circumstances, and is that enforced rather than merely documented?

05 **Certify with tiers that reflect autonomy**
A tier has to encode what an agent is permitted to do with the asset unsupervised, not just how much a person should trust it. See section 04.

06 **Demonstrate value with KPIs**
Breadth of consumers and applications, how you are building for the future, business impact today, and data quality, plus agent consumption: which agents depend on this, how often, and what happens to them when it fails.

07 **Consider DataOps and Agile methods**
Merging data engineering and data science practice makes reliable data easier to ship, and it puts monitoring config where it belongs: versioned alongside the agent that depends on it.

The through-line across all seven: anything that depends on a person noticing has to become a system noticing. That is the whole discipline in one line.

04 Certification when the reader is an agent

A certification tier has an important job: to tell a consumer how much autonomy it may take with an asset. When that consumer is a system, the tier has to be written for a system.

Tiered certification is familiar enough: gold, silver, bronze, assigned to assets, each tier carrying quality and freshness commitments. It works on a human, who can read the label and calibrate, treating a bronze table as directional and a gold one as decision-grade.

An agent does not calibrate. Given a table, it uses the table, which means the tier has to describe permission rather than confidence.

Tier	What it signals about the asset	What an agent may do with it
Gold	Decision-grade. Validated, monitored, owned, high freshness guarantee.	An agent may act on this autonomously, including customer-facing and financial actions. Requires continuous monitoring, lineage to source, and a tested rollback path.
Silver	Reliable for analysis. Monitored, owned, reasonable freshness.	An agent may use this for reasoning and retrieval but not as the sole basis for an irreversible action. Human confirmation on consequential steps.
Bronze	Directional. Use with judgment.	Not available to autonomous agents. Available in supervised contexts where output is reviewed before it reaches anyone outside the team.

Tier	What it signals about the asset	What an agent may do with it
Uncertified	Use at your own risk.	Should be unreachable by production agents. If an agent can retrieve it, the tier is decorative; the access control is what makes it real.

THE TEST TO RUN

Pick a production agent and ask which certification tiers it can actually reach. If the answer is "all of them, whatever is in the warehouse," your certification scheme is documentation rather than governance, and the tier labels are describing a discipline you are not yet enforcing.

This is where the majority of readiness conversations we see get uncomfortable, which is a reason to have that conversation early. Certification without enforcement is really just documentation. Enforcement requires knowing, per agent, what it can reach, which most organizations discover they cannot answer.

63% of AI practitioners who felt pressured to deploy AI faster than their teams were ready to support have already discovered an agent accessing data or systems they were not aware it could reach.

Agents in Production: The Builder's Perspective, 2026 · n=260

05 Context and lineage as product requirements

Two properties that indicate a data product is safe for agents to use.

WHERE MONTE CARLO LEADS

Monte Carlo monitors the data retrieved by RAG and tool calls, the context layer, in combination with agent behavior, trajectories, and outputs. Agent lineage then ties every output back to its source, so a bad answer traces to the pipeline, table, or index that caused it.

Requirement one: the context layer is part of the product

When your data product feeds an agent, the product boundary is wider than the curated table. It includes the retrieval path: the index the agent searches, the embeddings behind it, the document store it falls back to, the tool calls it makes into live systems. All of that is input to the output your business is accountable for, and almost none of it is covered by a conventional data quality

program.

The failure this creates is specific and common. An agent returns a wrong answer, but the model is fine, the prompt is fine, and the orchestration is fine. The retrieval step pulled from an index that stopped refreshing many days ago. Nothing in the pipeline monitoring fired, because the pipeline was healthy; the index was not part of the product as anyone had defined it.

64% of AI practitioners say context and data quality (freshness, retrieval, drift) is their single biggest visibility blind spot.

Agents in Production: The Builder's Perspective, 2026 · n=260

Requirement two: lineage from source to agent step

For agent-consumed data products, lineage is the difference between a fifteen-minute incident and a two-day one, because the first question in every agent incident is the same: was this the model, the prompt, or the data? Without lineage, that question is answered by elimination.

47% Only 47% of AI practitioners say their systems are easily traceable end-to-end when something goes wrong. **53%** describe agentic incidents as not easily traceable, requiring manual effort or stitching multiple tools together.

Agents in Production: The Builder's Perspective, 2026 · n=260

Source-to-agent step	Source vs. model attribution	Anomaly propagation
Map the table or pipeline through its transformations to the specific agent, tool, or retrieval step that consumed the output.	When quality drops, distinguish a data problem from a model or prompt problem as a first-class answer, not an investigation.	An upstream anomaly triggers a linked alert to the owners of affected agents, not only to the data engineering queue.

These are not new engineering programs; they are the completion of your existing data product definition. If the asset needs an owner, an SLA, and documentation, then context and lineage are what those three commitments require when the consumer cannot notice problems on your behalf.

06 Best practices

Six essential truths to keep in mind when designing and deploying data products that are AI-ready.

01 · Not everything has to be a data product

Do not minimize the value of focus. Sometimes you need a quick dashboard and that is sufficient. Promoting everything to "product" status is how the framework becomes bureaucracy.

02 · Do not do everything at once

Building a data product takes real effort. Start small, launch, gather feedback, iterate. Do not spend a quarter building something that ends up disused.

03 · It need not be a universal source of truth

Data products exist to let specific teams accomplish specific goals. One canonical product to serve every question is usually a way to serve none of them well.

04 · You do not have to be decentralized

Data products are a core data mesh principle, but a centralized team can build excellent ones. The organizational model is not the point; the discipline is.

05 · Scope the product to what the agent reaches

If an agent retrieves it, it is in scope. This includes the index, the document store, and the tool response. A product boundary drawn at the warehouse edge will not illuminate where your failures come from.

06 · Ownership must scale past the platform team

As agents spread, the people building them are increasingly not engineers. If every monitor needs a specialist, coverage will never keep pace with how fast agents ship.

That last tip is the one most likely to determine whether any of your efforts work at scale. Agent trust cannot bottleneck on a central ML or platform team. Non-technical AI practitioners need to stand up monitoring, create monitors in natural language, and interpret the results themselves. Otherwise ownership stalls behind a queue while agents keep shipping.

07 From marketplace to agent registry

Self-service discovery was built so people could find data products. The same problem now exists for non-human consumers, and it is the same solution with stricter requirements.

A data product marketplace is a centralized, standardized place for consumers to find, publish, and manage data products: an internal storefront. The three actions it has always needed to support:

01 **Find.** Explore and discover products, read contracts, preview samples, request new products, leave feedback.

02 **Publish.** Owners make products visible with descriptions, definitions, performance KPIs, and usage instructions.

03 **Manage.** Owners set standards, governance, and policy; monitor performance; get notified of incidents; maintain a changelog.

All three are important. But the consumer browsing your catalog is frequently software, which changes what the catalog has to expose. A human reading a product page can interpret a vague description and investigate the truth behind it. An agent resolving a metric cannot.

01 **Machine-readable contracts**
Definitions, freshness guarantees, and quality commitments an agent can retrieve and reason about, not prose on a wiki page.

02 **Certification exposed as permission**
The tier is not a badge on a page; it determines what an agent is allowed to do with the asset. See section 04.

03 **Live health, not last-reviewed date**
Current status of the product and its upstream dependencies, queryable at the moment of retrieval.

04 **Declared agent dependencies**
Which agents consume this product, so an incident can reach their owners and impact can be assessed in minutes.

05 **Programmatic and agent-native access**
API and MCP-compatible interfaces, so agents and coding assistants can query the registry directly rather than through a UI built for people.

If your marketplace project stalled, this is probably the argument that restarts it. The business case was always efficiency and discoverability, which was real but rarely urgent. The case now is that agents need a governed way to discover and evaluate the data they use, and the alternative is agents reaching whatever they can reach.

08 Agent trust for data products

Your data products are only as valuable as they are reliable, and with an agent consuming them, reliability is something you either measure continuously or discover when a customer reports a problem.

Everything in this guide describes a product you can design. This section is about knowing whether it is working, which is a different problem and the one where most data product programs fail.

What to measure

Dimension	The question	What it means with an agent consuming
Coverage	What share of this product and its retrieval path has active monitoring?	Uncovered surface is where agent failures originate, usually the index, not the table
Detection	Do you find issues before consumers do?	52% of AI practitioners learn about issues from users; agents remove the internal early-warning layer
Attribution	Can you tell a data problem from a model problem?	Determines whether an incident takes fifteen minutes or two days
Impact	Which agents and workflows depend on this product?	Impact analysis is impossible without declared agent dependencies
Reversibility	Can you stop an agent acting on a broken product?	36% of AI practitioners cannot disable or roll back a failing agent within minutes

What this is worth

The financial case for treating data products this way is well documented, and you should have the numbers on hand when this work competes for a quarter:

- **357% ROI** over three years, with **$1.5M+** in losses avoided through detecting data quality issues.
- **6,500 hours** saved annually in data personnel time.
- **80% reduction** in data and AI downtime.
- **65% reduction** in redundant data product creation, which is the quiet cost of a catalog no one can navigate.

Source: Forrester Total Economic Impact™ of Monte Carlo

A 65% reduction in redundant product creation is what happens when discovery and certification actually work: teams stop rebuilding assets that already exist because they could not find or trust the original.

> **WHERE TO START**
>
> Take your three most business-critical data products. For each, identify every agent that consumes it, every retrieval source in its path, and who is authorized to stop those agents if the product breaks. The gaps that exercise reveals are your roadmap, and in our experience it surfaces at least one dependency the team did not know existed.

09 Frequently asked questions

The short answers.

What is an example of a data product?

An airline's flight tracking system combining real-time GPS, flight manifests, and historical arrival data. A CRM syncing across marketing tools. A model forecasting stock returns. Increasingly: a retrieval index, a semantic layer, or a feature store.

What are the types of data products?

Dashboards, reports, machine learning models, curated tables, and unified schemas for specific business use cases. Vector indexes, semantic layers, and feature stores belong on that list too, and these frequently have no owner because no team classified them as products.

What does a data product do?

It turns raw data into something usable by combining datasets with product practice, business logic, and access controls, aligned to a business goal. With an agent consuming it, it also has to carry enough context and lineage to explain an output after the fact.

What is a data product in a data mesh?

A domain-specific, decentralized asset built for self-service use, adhering to governance and quality standards so it can be used across teams without central control. The mesh model is compatible with agents, but it raises the bar on federated governance: a domain team shipping an agent-consumed product needs enforced certification.

Why do you need data products?

To increase trust, demonstrate measurable value, reduce ad-hoc requests, and free the team for higher-value work. The agent-era reason is more direct: an agent acting on an uncertified, unmonitored, unowned data asset is an operational risk you cannot currently see.

Does this replace our data mesh or governance program?

No. It completes them. Ownership, certification, documentation, and lineage are things those programs already committed to. What changes is that committing on paper and enforcing informally stops being an option.

A human consumer provided a margin for error. An autonomous one does not, so the margin has to be built into the product itself.

MONTE CARLO

Trust your agents in production.

Monte Carlo is the agent trust platform. Built on seven years of enterprise-scale data observability, it connects the context layer to the agent layer, so you can trace a bad output back to the pipeline that caused it, validate how your agents actually behave, and catch upstream data issues before they ever reach production.

If this guide reflects how you think about the data products your agents depend on, we would welcome the chance to show you how Monte Carlo covers them, in your environment, on your agents.

Trusted by 400+ enterprises including Amazon, PepsiCo, CNN, Nasdaq, and JetBlue.

Request a demo

Or see it in action at montecarlo.ai