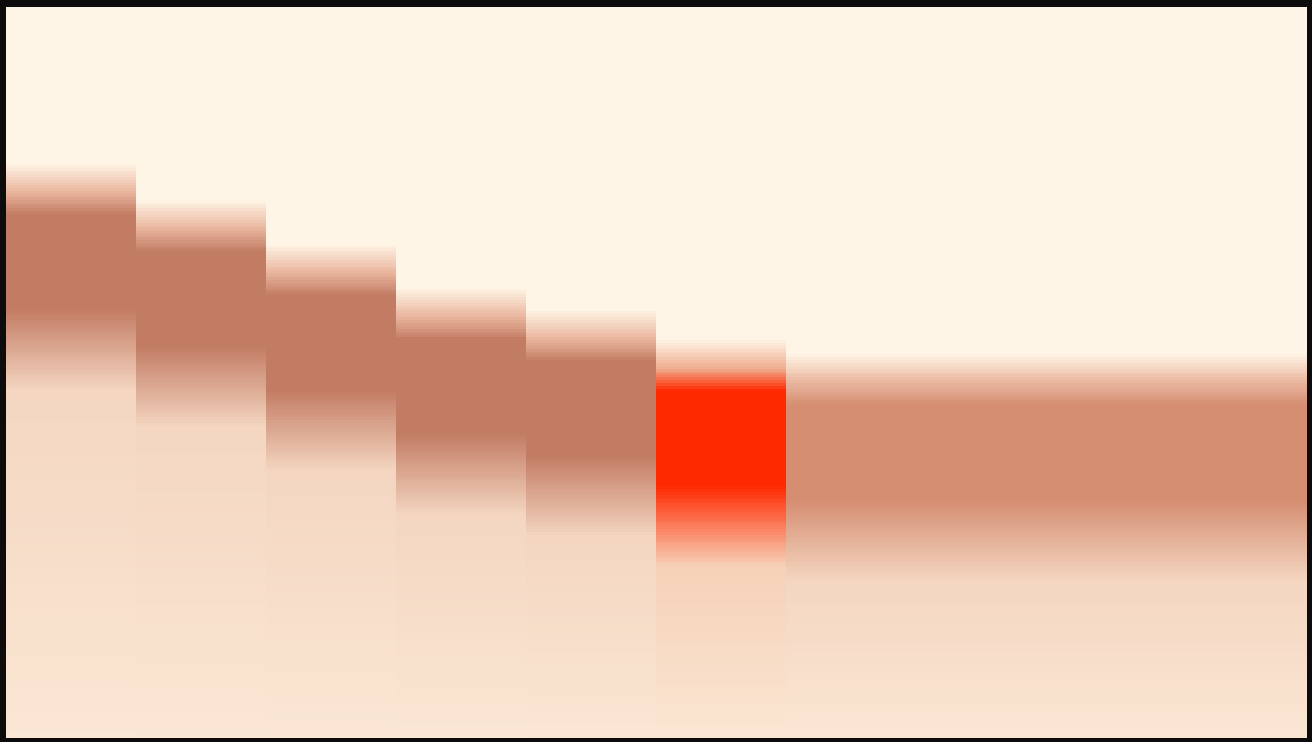
The data platform is becoming part of the agent runtime.

When an AI agent can change an order or confirm a price, the data it reads becomes part of a customer commitment. Leaders need to decide what the data platform should serve, which systems have authority to act and who keeps the service dependable.



WQA

Read online ↗

Executive summary

1. When a live workflow depends on data from the platform, agree what the data means, how current it must be and who restores access when it fails.
2. Use analytical data, a shared operational view or a direct source according to what the decision requires. The system that can enforce the commitment confirms it.
3. Give source, platform and business teams distinct responsibilities. Expand shared infrastructure when it reduces recurring work or risk and the business can sustain it.

In this Perspective

Treat operational context as part of the service 3

Separate serving facts from making commitments 5

Design for facts that change during the decision 7

Assign authority and ownership before scaling 9

Invest where the same capability earns its keep 11

Sources and further reading 13

Treat operational context as part of the service

Confirm a cancellation too early and the item can still ship. The customer may have to return it and contact support again to resolve the refund. Send every uncertain request back to the service team and even routine order changes can become expensive. The commercial decision is what the retailer can safely promise, which system can confirm it and what the agent should say while confirmation is pending.

Consider a fictional retailer that fulfils orders from shops and warehouses. A customer asks to cancel one item from an order split across two shipments. The order system identifies the line and its fulfilment route. The warehouse knows whether it can still stop the item. The payment service holds the status of the charge. A customer channel can show the last known order status while the warehouse is already packing it. Before confirming cancellation, the order workflow has to obtain the fulfilment decision and record what will happen to the payment.

This is a change in what the business asks its data platform to do. Historical cancellation data helps leaders decide which fulfilment routes to improve. Current order facts help resolve a customer request while there is still time to act. Once a live workflow relies on those facts, their availability and accuracy affect service performance. The platform team takes on an operating commitment even when the order and payment systems continue to own the transactions.

The operating cost becomes visible

A failed overnight report may delay a planning meeting. A failed context lookup in a customer conversation can immediately add work for service staff or leave a customer uncertain about a decision. The difference changes how the business funds and operates the dependency. For each fact the workflow needs, name its provider, when it must be available, what counts as usable and which decisions stop when the service cannot meet that standard.

Availability alone gives an incomplete picture. A service can return a response quickly while serving a fulfilment status too old for the decision at hand. Pair response time and availability with the proportion of requests served within the agreed freshness limit. Measure the effect on the workflow too. A system may technically recover while leaving a backlog of customers who still need a decision. The workflow owner needs to see that backlog and decide how it will be cleared.

Agree these expectations with the teams that have to meet them. The platform team owns the delivery of the data and the recovery of that service. Source teams make changes and corrections visible. The business owner sets the customer commitment and decides when the workflow should pause or pass to a person. A shared service needs an operating budget, on-call coverage and a way to resolve competing priorities.

References 6

Separate serving facts from making commitments

The system of record owns the business state and the rules for changing it. An operational context service gives the caller the facts it needs for a specific task. An analytical platform combines and retains data for planning, measurement and investigation. These responsibilities can sit on shared infrastructure, and some managed platforms support all three. Keep the responsibilities separate even when one vendor supplies the technology. A single product can expose very different guarantees across its query, replication and transaction services.

Use a direct read from the source when a single system owns the answer and can meet the latency and availability the task needs. Put an action behind a business API that applies the source's rules. There may be little reason to add a serving layer for a low-volume workflow with two stable calls. A dedicated read model becomes more attractive when several channels repeatedly make the same expensive joins or apply the same interpretation to different records. A materialised view can reduce that work while introducing a new dependency on refresh and reconciliation.

Historical order volumes and demand by fulfilment route can come from the analytical store when its refresh schedule and access controls meet the need. That can support decisions such as how to staff a service queue. Once a workflow relies on the data, give its users a clear expectation of when it will be available and what to do if it is late.

References 1

Make the facts usable outside their source

A field called shipped may mean a carrier label was created or that the carrier has taken custody. Neither interpretation tells the agent whether a warehouse can still stop an individual item. Give the context service vocabulary that matches the decision. Return whether a cancellation can be requested for the specific order line and keep that separate from its final status. If the request is accepted, show whether the payment change is pending or complete.

Preserve enough provenance to explain how the answer was assembled. Capture which records contributed, their effective times and the policy version that governed the result. Where sources disagree, specify which source has authority for that decision. The agent can say it cannot yet verify cancellation while an operator can tell whether the warehouse feed is behind or the fulfilment team has rejected the request. Keep internal identifiers and detailed sources within the teams that need access.

Give absence a meaning as well. An unavailable source, an out-of-date view and a denied request call for different action. Define the response states alongside the successful field list and test them with the people who handle exceptions today. Do this for the decision the data serves. A generic order status is useful across channels. A promise about a particular line needs more precision.

References 2

Design for facts that change during the decision

A weekly trend in cancellation demand can stay useful for planning. The fulfilment status of an individual line can change while a customer is asking about it. Define freshness for each critical fact in terms the workflow can use. Record when state changed at source and when the serving layer received it. A new ingestion timestamp can make an old fact look current. Measure lag and the share of decisions affected by stale inputs alongside response times.

Events can keep a shared operational view current. Until an update arrives, the view can disagree with its source. The customer channel may still show an item awaiting fulfilment after the warehouse starts to pack it. Design for the guarantees the source can provide. Use stable identifiers, reconcile against authoritative state and test replay during recovery. When the projection falls behind, the agent can explain that it is checking the order without promising a cancellation.

Make the commitment through the workflow that can enforce the current rules. A cancellation request can lose the race to packing even after a recent read. The order workflow should return the outcome for that line and preserve the rest of the order. If an acknowledgement is lost, a retry should recover the existing result rather than creating a second cancellation or refund request. This keeps the final guarantee with the business operation and lets the read path optimise the earlier enquiry.

References 3

Choose what still works when a source fails

A context service can make a workflow easier to build while giving it more ways to fail. A slow fulfilment request may slow the entire response. An unavailable payment feed can make an apparently complete answer unreliable. Decide which inputs are essential to each step and give each source a bounded time to answer. The agent can still explain the policy when fulfilment is unavailable. Where the order workflow has accepted the cancellation but cannot yet verify payment, communicate the confirmed order change and describe the refund separately.

Decide how much work the fallback can carry. If a source outage sends too many requests to an already busy service queue, restrict the actions the agent offers and show customers what they can still do. Restoring access to the source is one step. Resolving the requests that accumulated during the outage is another.

Escalation is useful only when it allows somebody to resolve the situation. A service colleague should receive what the customer requested, what the system confirmed and which dependency or rule prevented the change. They should be able to verify the relevant detail without reconstructing an entire agent session. The service owner should track whether deferred cases are resolved without extra customer contact. That gives the business a way to decide whether fallback is genuinely protecting the service or steadily expanding manual work.

Assign authority and ownership before scaling

A commerce agent may read an order on behalf of a signed-in customer and invoke a cancellation tool under a service identity. The tool credential identifies the calling workload. The request context identifies the person it represents and the order line it can act on. A trade customer using a buying portal has a similar identity problem with different rules. The person may belong to several buying accounts, and the relevant contract depends on which account is purchasing. Carry the identity and scope through the read and the action so both services can enforce their own decisions.

Apply authorisation where protected facts are served and where state changes are accepted. Retrieval tools and shared indexes need the same care as operational APIs. Review what can cross customer, case and regional boundaries, and how an access change takes effect in replicas and caches. A broadly privileged credential can make an impressive demonstration easy. It also creates a mismatch between what the workflow can show and what it can defend once several teams rely on it. Design the service to return only the fields required by the task.

Allow a source owner to explain which rule allowed a read or rejected an action without relying on a model's description of its reasoning. For a sensitive action, keep the business approval or escalation in the authoritative workflow and make the agent convey its outcome. If a user changes role or consent is withdrawn, decide how quickly each part of the path must recognise the change. Test that expectation through retrieval, cached data and the write interface. Treat the need to remove access as an operating requirement alongside the need to grant it.

References 4

Give operators enough evidence to recover

An operator should be able to follow a disputed customer commitment from the request through the context lookup and authorisation check to the system that accepted or rejected it. An order cancellation and a trade quote will have different business records. Both need a way to connect the decision, the version of the facts and the system response. Record enough information to investigate while limiting access to personal information and conversation text.

Telemetry conventions can help connect spans for agent and tool activity across services. The trace still needs business semantics. A successful tool request can return stale eligibility or a request that is still pending. Instrument the states the team cares about, including denied, stale, unavailable, conflict and confirmed. Test realistic sequences in which packing starts during the interaction or payment confirmation is delayed. Evaluate whether the workflow honoured the order change and what it told the customer.

Decide who responds when these measures deteriorate. The source owner should be able to confirm the record and its data contract. The serving team should know whether lag, a schema change or an access failure prevented it from supplying useful context. The workflow owner has to decide whether to pause actions, switch to a human path or accept a temporary reduction in capability. Use a regularly exercised runbook to make those handoffs explicit. A trace is valuable when it helps an accountable person resolve a decision or correct the next one.

References 5

Invest where the same capability earns its keep

Begin with the business outcome and its current baseline. For order cancellation, track eligible requests completed, repeat contacts, items shipped after cancellation was confirmed and refunds that require manual correction. A pilot that clears the chat queue while moving work into reconciliation can appear efficient. Include the cost of running the source integrations, monitoring the data and intervening when the workflow cannot complete.

A second workflow should test what can genuinely be shared. In a fictional B2B buying portal, an agent can use prior orders to suggest products for reorder. The price and pack size it can quote depend on the current agreement, buying account and product terms. The pricing service must confirm the terms of a quote. Order cancellation and quote preparation may both need to express what a fact means, when it was observed and whose authority applies. Their business rules and final decisions still sit with different systems.

Fund a shared capability where evidence from several workflows shows that teams repeatedly solve the same serving problem. One candidate is a common way to expose account scope, provenance and freshness across services. Another is a shared, operated view of operational facts when several channels need the same semantics. A direct source integration remains a sound choice when it already meets the service objective. If the shared layer adds a failure point, requires a new team or makes changes slower, count that cost in the investment case.

Expand when the next workflow can show a measurable gain in delivery effort, consistency or reliability. Give it a way to exit if the shared service does not meet its needs. Make that decision jointly. The data leader funds the reliability that workflows need. The workflow owner measures what customers and staff gain from it. Each subsequent workflow should make that trade-off easier to judge.

Decide what belongs in the shared platform

Take a workflow where a wrong commitment has a visible consequence. Walk through an ordinary request, a change in source state during the interaction and the loss of a required service. Repeat the exercise with a second workflow. Ask the people who own the process, data and systems to agree the answers below before deciding which responsibilities to keep local and which to share.

Use the result to decide whether to keep a direct integration, build a shared read service or improve the authoritative system first. Agree what the next workflow needs to demonstrate before committing further investment.

- Name the decision and the commitment. Identify what the agent can suggest, what it can request and what only the authoritative system can confirm.
- List the required facts. Give each an owner, meaning, source, freshness requirement and response when it is absent or disputed.
- Choose the read path. Use a direct API, a projection or an analytical source according to the latency, reliability and semantics the task needs.
- Set the authority boundary. Specify whose request the agent represents, who can see each fact and where the final action is authorised.
- Make the operating commitment explicit. Agree the fallback, who restores service, how the business measures the impact and what would justify further investment.

What does this change for your organisation? Talk to us ↗

Sources and further reading

1. Microsoft Azure Architecture Center · Materialized View pattern
2. W3C · An Overview of the PROV Family of Documents
3. Microsoft Azure Architecture Center · Event-Driven Architecture Style
4. Google researchers, published by USENIX · Zanzibar: Google’s Consistent, Global Authorization System
5. OpenTelemetry · Semantic conventions for generative AI agent spans
6. Google Site Reliability Engineering · Service Level Objectives

The data platform is becoming part of the agent runtime.

WQA

Intelligence, engineered.

Data platforms
Published 27 November 2026

Please consider the environment before printing this document.

© 2026 WQA UK Ltd. All rights reserved.