slalom | salesforce

Behind the Curtain

The headless architecture guiding the Agentic Enterprise

One intelligence layer for every interface

The arrival of headless in enterprise software has changed the economics of building.

Creating a custom portal, app, or agent is faster and more accessible than ever, making the cost of building no longer prohibitive. And because the experience layer is no longer locked to the system of work, organizations can begin from the experience they want to provide, then assemble the right pieces for each part of the process.

What follows identifies the key decision points in a headless journey, drawing on insights from Slalom architects building cutting-edge headless experiences, and ends with sample reference architectures behind two headless workflows.

Twenty-six years in the making

For the #1 Agentic CRM, headless isn't new. Twenty-six years ago Salesforce launched the world's first enterprise web APIs, exposing an enterprise application over the internet as a programmable service. APIs were built into how the company operated from the start, and the capabilities have only grown more robust.

Headless 360 expands on this API leadership, opening Salesforce to every agent while inheriting the governance and controls organizations expect. The platform now ships 60+ MCP tools, 100+ agent skills, and 220+ CLI commands, alongside the 4,000+ APIs already in place.

slalom | salesforce

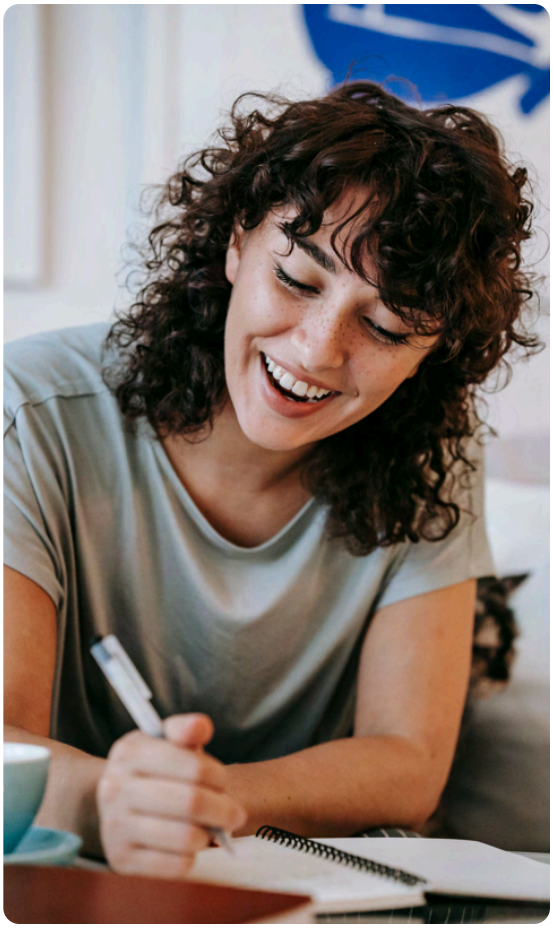
“Salesforce has been API-first since its inception, so working with data outside the UI has always been possible. What’s new is this layer built specifically for agents, running on the same platform and inheriting the governance organizations already have.”

Frank Klensch

Managing Director, Enterprise Technical Architecture & Engineering, Slalom

This enables organizations to build a shared intelligence and governance layer that sits apart from any single interface. Companies that have adopted agentic transformation run an average of 1,057 business applications, according to the 2026 MuleSoft Connectivity Benchmark Report. A customer might start in a portal, a service representative in Service Cloud, and an account manager in a Slack channel.

Beneath all three interfaces, the intelligence layer interprets the request, decides which steps it can complete on its own, calls the systems that hold the answer, and routes to a person when policy requires one. Every step is recorded. The interface can change without rebuilding the foundation.



Slalom + Salesforce: Building with Anthropic's engineering patterns

Slalom's accelerated engineering frameworks support Claude Code and skills, and the Agentforce agents and Slack plugins highlighted in this piece were built using Anthropic's models applied to Salesforce development.

The decisions made early on determine whether a headless build succeeds or stalls. Five of them separate a working prototype from a production system:

01 What outcome are we driving?

02 What should the agent decide?

03 Where does the data live?

04 How will it stay governed?

05 Who will keep it running?

But before any of that work starts, there first needs to be alignment on what headless is, what it is not, and what it means for your organization.

What headless means in practice

Headless is an architecture pattern, not a product. It refers to the data, business logic, and workflows that already live in your applications being exposed via APIs, MCP tools, and CLI commands, so that any application, agent, or interface can call them directly.

Headless 360 is Salesforce's implementation of that architecture pattern. It packages the platform's APIs, MCP tools, over 100 prebuilt agent skills, and CLI commands together with the identity, permissions, and audit controls that already govern the org. An agent or outside application can act on Salesforce data without stepping outside those controls, with the Salesforce UI remaining as one of the available interfaces.

slalom | salesforce

“Headless is a shared intelligence and governance layer decoupled from any single interface. The interface can change without rebuilding the intelligence layer every time.”

Allen Mann

Senior Director of Enterprise AI Innovation, Slalom

Those agent skills function as structured prompts that give a coding agent explicit guardrails for building against a specific org, including authentication, single sign-on, and login. A developer can pull them from a GitHub repository into Claude Code, Cursor, Agentforce Vibes, or their tool of choice and the agent builds the same way every time rather than improvising. The same capability reaches business users as a downloadable plugin, so someone who will never open a repository can authenticate into Sales Cloud and query their own book of business. In both cases, sharing rules and permissions remain in effect.

"Headless 360 is Salesforce, wherever you want it to be. Companies have been building headlessly on this platform for years. What's new is that the building blocks now ship with it, so teams stop assembling the plumbing themselves and go straight to building the experience."

Ryane Bohm

Senior Director,
Product Marketing,
Salesforce

Below is an example of a business user request and the corresponding Headless 360 workflow running behind the scenes:

1. Surface

A sales representative, working in the conversational interface of their choice, asks: "Which deals over $500,000 are at risk of slipping this quarter? Create follow-up meetings."

2. Agent reasoning

The agent interprets intent and selects the MCP tools the request requires. This is the probabilistic part of the workflow, and it runs inside a defined boundary of what the agent is permitted to do.

3. System of work

Salesforce is queried with the requester's security applied. Opportunities, forecast category, and last activity are read in place, with no export and no copy. Every sharing rule and field-level permission that governs this person in the standard interface governs them here.

4. Action

At-risk deals are flagged and follow-up meetings are created. Each write passes through the same flows and validation rules that would fire if a person clicked the button.

5. Render

The answer returns to the interface where the question was asked.

Underlying all five stages sits the shared intelligence layer, which routes and governs the request; and Salesforce's Trust Layer, through which every LLM request passes with zero data retention and a full audit trail. The example request also touches other systems already running in most Salesforce environments: Customer 360 for the record and the business logic, Data 360 for context, Agentforce for the reasoning, and Slack or Tableau for the system of insight.

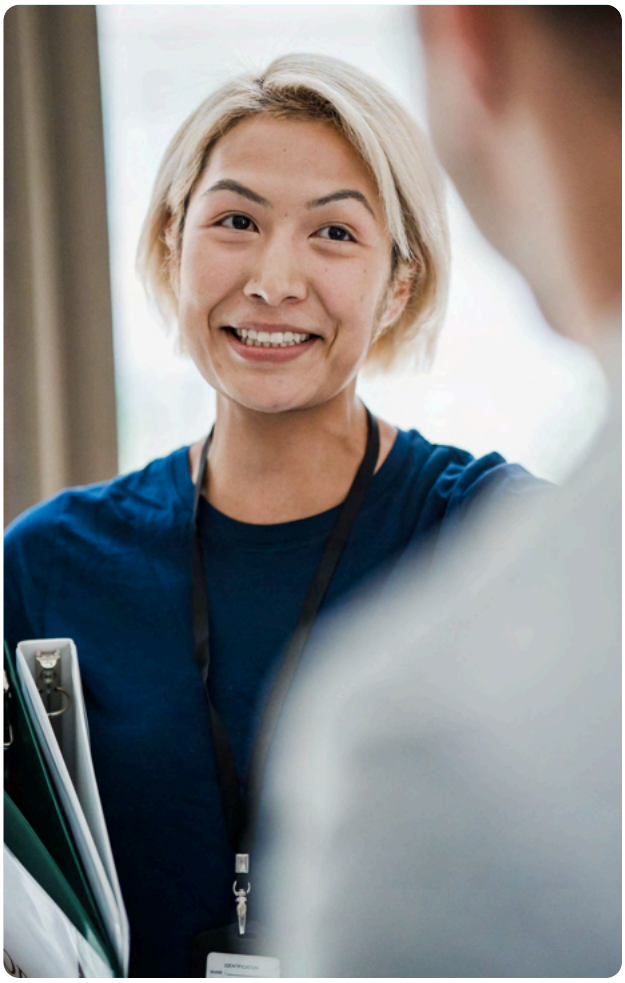
This pattern works because governance travels with the request. Approaches that bypass the governed system of work may require teams to recreate or federate permissions, audit trails, and business rules elsewhere; when Salesforce remains in the execution path, many of those controls can be inherited. That distinction is what separates an architecture that meets stringent compliance requirements from one that works only in a demo. The sections that follow are the decisions that determine which one an organization ends up with.

Decision 1:
What outcome are we driving?

Start from the outcome, never from the technology.

This is the decision all of the others depend on, and it is the one organizations are most likely to skip.

Slalom sees this pattern often: organizations acquire a technology and then search for the problem it solves. These projects can meet every success measure set at kickoff and still have limited adoption a year later, because the desired outcome isn't defined from the start.



The solution is to start with the business outcome, then design the experience and technology around it. Describe what your target users should be able to do and how they should experience it, then work out the components to deliver it. While Salesforce can supply the whole stack, it also works alongside the data platform, hyperscaler, and tooling an organization already runs. The right mix depends on what is already in place and what's needed to meet user needs.

Five questions to help define the outcome:

01 **What result are we driving toward?** Name it in terms the business already measures.

---

02 **Who does that work today?** Group people by the workflow they perform, not the license they hold.

---

03 **What does their day actually look like?** The answer determines what is realistic to ask of them.

---

04 **What should happen without anyone doing anything?** This is where headless earns its place, because work that used to require a human to intervene can now happen in the background.

---

05 **Where does the work land?** Service teams and operations staff working queues may be best served by the Salesforce UI, which is designed for moving through high volumes of records quickly, while sellers, field technicians, and relationship managers who work in meetings and messages may choose to have Salesforce brought into Slack or email. Customers and partners usually need a purpose-built portal or an agent they can address directly.

Slalom works through these questions with clients before anything gets built: helping understand how people actually work today, identify where an agent changes the workflow, and define the outcome in terms the business can measure.

The interface itself is no longer a constraint. Salesforce has demonstrated a working CRM assembled in Claude Code and connected headlessly to the Salesforce Platform, arriving with the metadata, the permissions, and populated data already in place.

Slalom recently worked with an organization where most of the intended users do not spend their day inside the CRM interface. High performers doing high-touch client work simply weren't taking time to pause and update records, because the task sat outside the way they work. Rather than redesign the interface, the organization started from the experience it wanted those teams to have, which was one where the record kept itself. The CRM now comes to them, capturing the right data at the right moment in the background of the tools they already use.

slalom | salesforce

> “The aim is getting the software to work from the perspective of the business rather than getting people to work from the perspective of the software.”

Mike King

Sr. Director of Enterprise Architecture, Slalom

Slalom is taking this further by building an experience where nobody enters anything at all. Transcripts, call logs, meeting activity, email, and Slack feed a structured data model, so the system registers on its own that a call occurred and that a new contact entered an account hierarchy.

Extracting and storing structured information based on these unstructured data sources is what makes insights possible. An agent tasked with identifying the next step on an opportunity will struggle against raw call logs.

First moves:

01 **Name the business result** in terms the organization already measures.

02 **Analyze your target users** by the work they perform rather than the license they hold.

03 **Establish where each group already works,** because that answer shapes how the outcome gets delivered.

04 **Check for multiple logins.** If the design requires someone to leave the tool they work in, revise it.

Decision 2:
What should the agent decide?

Separate the probabilistic work from the deterministic.

Some steps are probabilistic; that is, they require reasoning before taking action. Others are deterministic; these already have a correct answer the business settled years ago: pricing rules, prescription approvals, credit checks, discount thresholds, all baked in over time with security and validation built in.

Agentforce defines and governs agentic reasoning, while deterministic work stays where it already lives. When an agent calls an MCP tool to update an opportunity stage, every flow and validation rule fires exactly as it would if a person clicked the button, because the business logic lives directly in the platform.

Consider a customer return that arrives outside the 30-day window. According to the organization's built-in rule, the return generally can't be accepted; but a clause added years ago exempts Platinum customers from that rule, so credit is issued and the case closes. An agent calling that workflow inherits the rule and the exception together, without anyone restating either one.

Multi-step sequences need additional controls. Headless 360 allows guided determinism across an agentic workflow through the use of existing Salesforce logic, so a sequence running heedlessly produces consistent, reliable outcomes.

Teams that separate probabilistic from deterministic work in this way only need to define the reasoning once, because the rules underneath do not move. Every new surface inherits both.

Steps can also switch from reasoning to rules-based once a team identifies a consistently right answer. At the volumes enterprise workflows run, a process that works most of the time introduces operational risk, and the fix is often to pull the decision out of a free-text prompt and encode it.

Slalom helps clients design that boundary: what is safe to automate, what has to stay deterministic, and where accountable human judgment belongs.

First moves:

01 **Separate the workflow** into steps that require reasoning and steps that have an answer your business has already defined.

---

02 **Inventory what you already trust,** since existing flows and approval processes are assets in this architecture.

---

03 **Resist rebuilding logic that already exists.** A rule enforced today is enforced when an agent calls it.

---

04 **Revisit the boundary as the workflow matures.** Decisions the model handled early may belong in code once the right answer is known.

Decision 3:
Where does the data live?

Work backward from the outcome to the data.

A lack of data readiness risks stalling more agentic projects than any other single factor. Starting from the outcome and working backward makes data prep more manageable.

Questions to ask when getting started:

01 What context does this agent or workflow need to do its job?

02 Which systems hold that context today?

03 Is the data structured in a way that meets governance and visibility requirements?

04 Does it need another logic layer on top before an agent, or a human in the process, can consume it?

05 How do we make it available?

Many enterprises hold their data infrastructure in cloud data platforms like Snowflake or Databricks. Zero copy makes that data available to the Salesforce ecosystem without moving it, along with prebuilt paths into Marketing Cloud, Service Cloud, Loyalty Management, and Commerce. Routing through Data 360 puts the data through a trusted context layer along the way, and can become the bridge connecting Salesforce data to the rest of the enterprise.

But **available** data and usable data are two different things. For data to be usable, an agent must be able to decipher which version of a record is correct when systems disagree and audit where an answer came from. Data 360 harmonizes structured and unstructured sources and federates them in real time. Informatica's MCP server adds the catalog layer, exposing discovery, classification, lineage, ownership, data quality, and master data resolution as tools an agent can call directly.

Slalom works with clients to determine which capabilities to keep, rationalize, modernize, or introduce, then defines the role Salesforce plays in delivering the outcome. The result is an architecture built for the workflow rather than layered onto whatever is already there.

The Data 360 MCP Server extends trusted Data 360 context beyond the Salesforce UI, exposing nearly 200 Data 360 APIs through open standards. With the Data 360 MCP Server, authorized agents can execute tasks like building semantic models, transforming data, generating calculated insights, intelligently mapping fields, inspecting identity graphs, creating audience segments, activating campaigns, and standing up entirely new data streams—all through natural language.

The Data 360 MCP Server will also includes prebuilt skills for common workflow,s which will become available soon.

On a recent manufacturing engagement, Slalom added AI tooling behind the systems people were already using rather than moving anyone to a new interface. Work continues in the client's project management tool, then an agent observing from the cloud platform they already run picks it up, does its part, and pushes updates back into the system of work. Nobody logs into a separate portal to get the benefit, and the record still lives in Salesforce.

First moves:

01 **Scope the data to the outcome** rather than to the enterprise.

02 **Establish which system holds the source of truth,** so an agent never has to choose between conflicting records.

03 **Favor patterns your teams already run.** New ones cost more than they look like they will.

Decision 4:
How will it stay governed?

Inherit the security model rather than rebuilding it.

In Slalom's research, 46% of people using agent-enabled workflows report greater accountability and governance of AI outputs, and 53% report a greater focus on outcomes and oversight. That level of accountability depends heavily on the permission model underneath it.

Salesforce's security model was built for that. HIPAA compliance, PCI compliance, and FedRAMP certification all rest on the ability to tie granular permissions to individual humans who authenticate into the platform. Enforcement happens at the API level rather than the UI level, so a browser request, a REST call, and an MCP tool all run the same checks.

Headless does not bypass those controls. Every action inherits the profile, sharing rules, and field-level security already configured in the org, whether it originates in the UI or in an agent running somewhere else.

If data is reachable elsewhere, it can be tempting to serve occasional users through a shared connection instead of their own login. But decoupling permissions from individual users transfers responsibility for the security model, and its certifications, to wherever the data now resides.

slalom | salesforce

"Governance is not something you add to an agentic architecture. It is either inherited from the platform the work already runs on, or it gets rebuilt from scratch in every place the data lands."

Mike King

Sr. Director of Enterprise Architecture, Slalom

The security reviewer test

Permissions define what an agent can reach. Auditability determines whether anyone can verify what it did. The practical standard is whether a security reviewer can open any completed workflow and see what system was allowed to do what, why the action occurred, who approved it, and what happened afterward. An architecture that cannot answer those questions is not governed, regardless of what controls exist on paper.

Producing an effective audit trail means adhering to four principles, each enforced at runtime:

01 **Every action has an authority ceiling.** Tools and humans carry specific permissions, and the boundary of what an agent may do is defined before it is deployed.

---

02 **Policy is checked before action, not after.** Validation happens on the way in.

---

03 **Consequential exceptions route to an accountable person.** When an agent has no answer or reaches its ceiling, that outcome reaches a named human rather than going unrecorded.

---

04 **Every important transition leaves a record.** Telemetry is built in from the start, because what is not logged at the time cannot be reviewed later.

For regulated industries, each LLM request routed through Salesforce's Trust Layer carries zero data retention, automated toxicity and bias detection, and full audit trails.

First moves:

01 **Pressure-test any design that decouples access from individual users,** and establish what would have to be built to reproduce the controls given up.

02 **Define the authority ceiling for each agent** before it is deployed.

03 **Run the security reviewer test against the design** before anything is built.

04 **Confirm which workflows route their reasoning through Agentforce,** since those inherit Trust Layer governance automatically and others may not.

Decision 5:
Who will keep it running?

Plan for day two before day one.

A first working version comes together quickly. Bringing it to the standard required to put it in front of customers or regulators is a different exercise, and it requires two kinds of operational work.

The AI-specific work covers monitoring agent behavior, running evaluations, tuning prompts, and replacing models as better ones become available. It has little in common with traditional CRM system maintenance. These processes continue to change as the technology changes and as people interact with them, and teams need to be staffed for that work before launch. Cost considerations are an important factor here as well. Every call to an LLM carries a price, through credits or tokens, so the run rate is something the team must take into account and manage to prevent any surprises.

slalom | salesforce

"Running an agentic workflow is not the same discipline as maintaining a CRM. Teams that plan for day two before they build are the ones still improving the workflow a year later."

Frank Klensch

Enterprise Architecture and Service Delivery Innovation, Slalom

The engineering work determines whether the system handles the required concurrency, withstands a denial of service attempt, and remains maintainable in five years. It also raises the operating model question that gets skipped most often: when the system fails outside business hours, who is notified, and how do they know what to do?

Slalom's conversations on headless today are often with technology companies that have established practices for observability and logging. As headless reaches teams building their first agentic workflow, the operating model must be defined explicitly: who owns the agent, how failures are detected and recovered, where escalations route, how cost is managed, and how the workflow keeps improving after launch. Slalom helps establish that model with clients before the build, when those decisions are cheaper to make.

First moves:

01 **Define the support model before the build,** including who will own the agent after go-live.

02 **Document the failure states** and the response to each one.

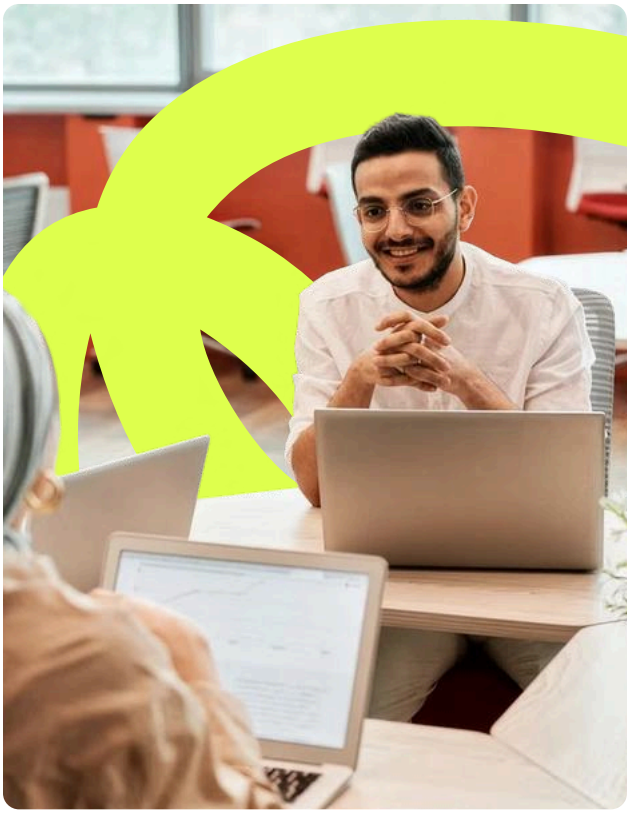
03 **Confirm the team is staffed** for evaluation and tuning before launch, not after.

Under the hood

The reference architecture for two headless builds

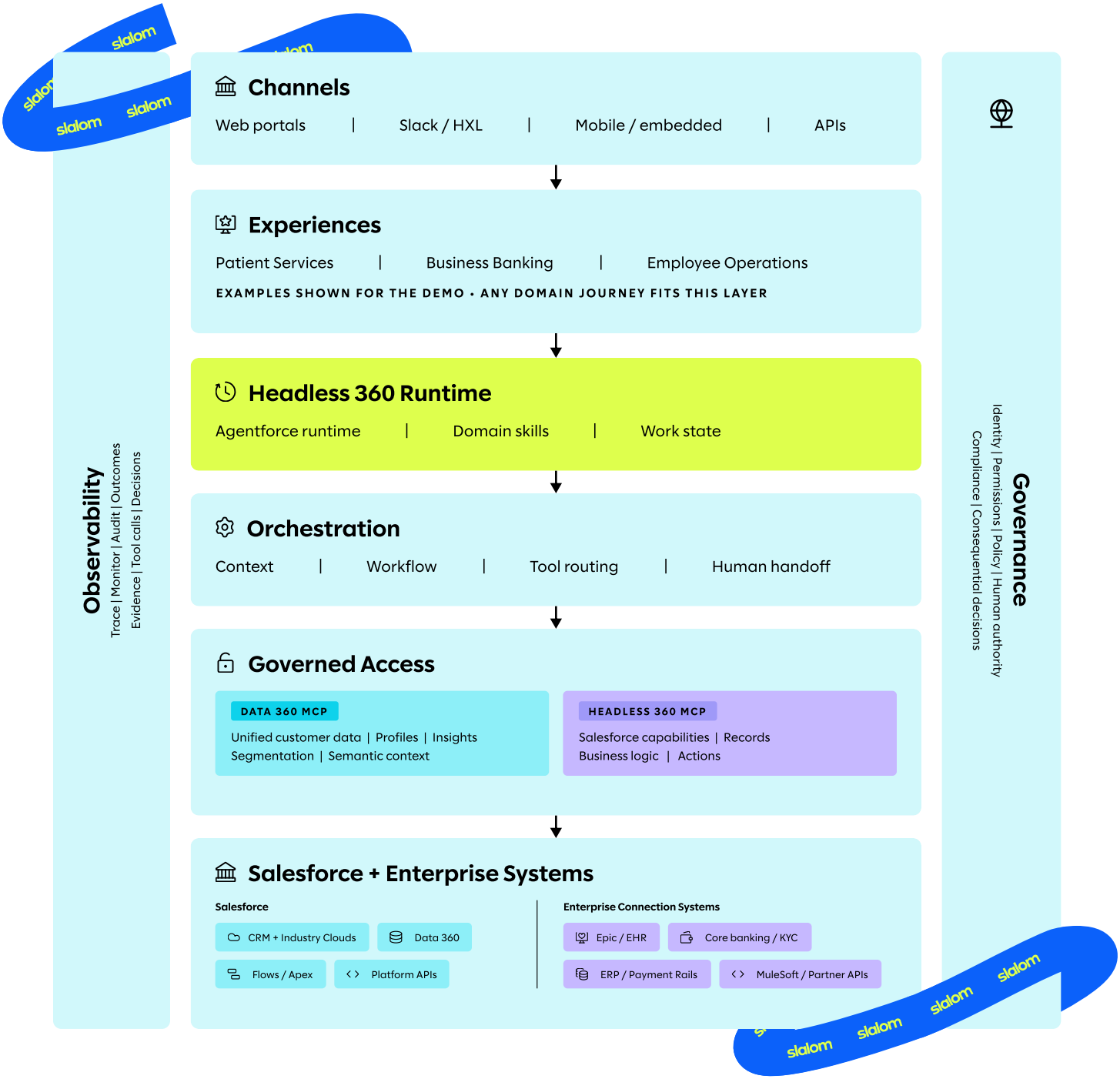
At Dreamforce this year, Slalom demonstrated multiple reference scenarios across the Patient Services and Business Banking industries. Each demo build was chosen for its complexity. Patient benefits and commercial lending both operate under heavy regulatory constraints, with approval steps and policy exceptions at nearly every stage, so an approach that holds in these industries tends to also work well in less demanding environments.

Included on the next page is the reference architecture for these demo headless builds. This is provided here as an example rather than blueprints. The right architecture for any organization depends on what they're currently running and what they're trying to change.



While each build has its own intelligence layer, the structure is the same: context assembled from governed sources, reasoning inside a defined boundary, a human brought in on exceptions, and evidence recorded at every step. What differs is the policy being checked and the system of work being called.

The reference architecture

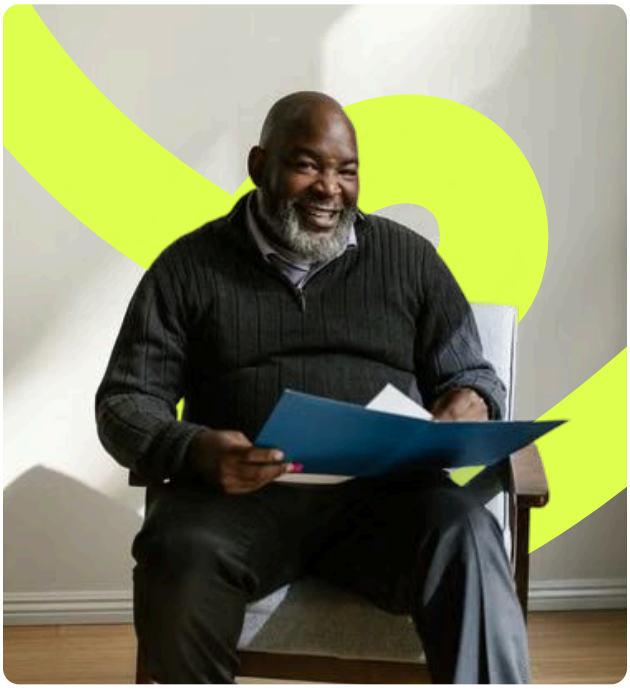


Demo scenario: Patient services

Patient access work sits between a provider, a payer, and a patient, and almost none of it is a single transaction. Benefits change, prior authorizations expire, and a coverage answer that was correct last week may not be correct today. The people doing this work spend their day chasing the most up-to-date information across systems that often disagree with each other, which makes it a natural fit for an approach that brings the answer to them rather than asking them to go find it.

In this demo scenario, the patient services agent reasons about administrative status rather than clinical judgment, and every exception routes to a person before anything is committed. The context lives in several places at once, so discovery MCP servers assemble it from governed sources rather than the agent querying systems directly.

In practice, a patient checks coverage in a secure portal, the agent establishes who they are and where the exception sits, an Agentforce agent on the Atlas reasoning engine determines there is enough context to create a work item, and the Headless Experience Layer packages that into an approval a patient services manager handles in Slack. The benefits API executes the deterministic work with the same validation applied to any other caller, and observability captures the path so the sequence can be reconstructed.



Demo scenario: Business banking

Commercial lending runs on policy thresholds. A request below the line can be approved automatically, a request above it needs a person, and the difference is often a matter of a few thousand dollars. Relationship managers spend their time assembling the case for the exception rather than making the decision, which is work an agent can do without touching the decision itself.

In this demo scenario, the policy being checked is a lending threshold rather than a benefits eligibility rule, and the system of work is the lending platform rather than the payer. A business owner requests working capital for seasonal inventory above the automatic approval threshold, context is assembled, a recommendation is formed, an accountable human is brought in, the deterministic system executes, and evidence is recorded at every step.

Each headless build was measured against the manual process it replaces. In patient services, work that normally takes about an hour completed in roughly 90 seconds. Once an organization has worked through a headless build, it has a pattern it can apply to the next workflow. It also has an intelligence layer, with its governance already established, that the next workflow can tap into.

Building headless with Slalom

Headless work spans many systems: the data platform, hyperscaler, integration layer, existing automation tools, and more. When designing across all of those systems to create a headless experience layer, clients can benefit from the right expertise.

What to look for in a headless implementation partner:

01 They keep business value in focus through the whole engagement, not only at kickoff.

---

02 They evaluate your business rather than scoping to the product they know best.

---

03 They recommend architectures that take advantage of investments you have already made.

Slalom is a top-ranked partner across Salesforce, AWS, Azure, GCP, Snowflake, and Databricks, with 14,000+ certifications across the Salesforce ecosystem. Slalom has been building AI on the Salesforce platform since 2022, and its enterprise AI innovation practice now deploys that work in the industries with the strictest governance requirements.

Slalom is also investing heavily in the Salesforce Forward Deployed Engineer (FDE) program and holds FDE partnerships with several hyperscalers. This model puts smaller cross-functional teams on an engagement, staffed with practitioners encouraged to build depth in data platforms rather than remaining inside a traditional Salesforce developer lane. When the formal FDE motion doesn't apply, Slalom runs the same model as Accelerate teams. Both work by co-engineering, embedded in the client's team rather than positioned above it.

Slalom's accelerated engineering frameworks support Claude Code and skills and stay agnostic across providers. The Dreamforce builds highlighted in the previous section applied Anthropic's patterns to Salesforce development, covering Agentforce agents and plugins for Salesforce and Slack.

Deciding what outcome to drive, what an agent should handle, where the data lives, how it stays governed, and who keeps it running is the work Slalom's architects do on every headless engagement. The experiences that will define the next decade are being designed by organizations working through exactly these questions.

The Evaluation Checklist

The questions below consolidate the First Moves from each section into one actionable checklist. Review each one before kicking off a headless initiative.

Outcome

- [ ] Have we named the business result we're driving toward in terms the business already measures?
- [ ] Have we analyzed target users by the work they perform rather than the license they hold?
- [ ] Do we know where each group already works?
- [ ] Does any part of the design require someone to leave the tool they work in?

Workflow composition

- [ ] Have we separated the steps that require reasoning from the steps that have an answer the business already defined?
- [ ] Which existing flows and approval processes can this reuse?
- [ ] Are we rebuilding any logic that already exists?
- [ ] Do we have a plan to revisit the boundary as the workflow matures?

Data

- [ ] Is the data scoped to the outcome rather than to the enterprise?
- [ ] Which system holds the source of truth?
- [ ] Are we extending a pattern the team already runs, or introducing a new one?

Governance

- [ ] Have we pressure-tested any design that decouples access from individual users, and established what would have to be rebuilt?
- [ ] What is the authority ceiling for each agent?
- [ ] Has the design passed the security reviewer test?
- [ ] Which workflows route their reasoning through Agentforce?

Operations

- [ ] Who owns the agent after go-live?
- [ ] What are the failure states, and what is the response to each?
- [ ] Is the team staffed for evaluation and tuning as ongoing work?