

Agentforce Multi-Agent Orchestration: How Multiple AI Agents Work Together

Alpesh Kanubhai Patel
Information Technology(Salesforce Developer)
Abingdon, Harford

Abstract - As organizations extend Salesforce Agentforce across more business processes, a single agent is often asked to cover responsibilities that belong to different teams, data domains, and security boundaries. Salesforce's response is multi-agent orchestration, which it lists as generally available in the Summer '26 release (June 2026). In this model, a primary agent acts as the single point of contact for a user and delegates work to specialist agents, each grounded in its own data and equipped with its own actions. This paper explains how the pattern works in Agentforce: description-driven routing through the Atlas Reasoning Engine, context handoff between agents, deterministic control through Agent Script subagents, and delegation to external agents through the open Agent2Agent (A2A) protocol. It presents four illustrative implementation scenarios, a decision framework for choosing between subagents, specialist agents, and external agents, and a practical approach to testing routing with Agentforce DX and monitoring production behavior with Agentforce Observability. The paper also examines limitations, including misrouting, latency, debugging complexity, and agent sprawl, and proposes governance controls built on least privilege, minimal context sharing, and human-in-the-loop approval for high-impact actions. The intended audience is Salesforce administrators, developers, architects, consultants, and technology decision-makers planning to scale beyond a single agent.

Keywords - Salesforce Agentforce, Multi-Agent Orchestration, Primary Agent, Specialist Agents, Subagents, Agent Script, Atlas Reasoning Engine, Agent2Agent (A2A) Protocol, MuleSoft Agent Fabric, Agentforce DX, Agentforce Observability, AI Governance, Human-in-the-Loop, Enterprise AI

I. INTRODUCTION

Most Salesforce teams begin with Agentforce in the same way: one agent, one channel, and a small set of jobs and actions. That approach works until the agent's scope starts to grow. A service agent that answers order questions is later asked to handle refunds, then warranty claims, then contract renewals. Each new responsibility adds instructions, actions, and edge cases to the same agent, and its routing decisions become harder to predict and harder to test.

Multi-agent orchestration addresses this problem by replacing one oversized agent with a coordinated team. A primary agent talks to the customer or employee, while specialist agents each own a narrow domain and work behind it. Salesforce made this capability a headline feature of its Summer '26 release, which became available on June 15, 2026 [2].

This paper explains how the pattern works in Agentforce, where it fits, where it does not, and what administrators,

developers, and architects should plan for before splitting one agent into many. Section II defines the core concepts and terminology. Section III describes the mechanics of routing, context handoff, deterministic control, and cross-platform delegation. Section IV presents illustrative implementation scenarios. Sections V through VII cover benefits and limitations, implementation considerations, and security and governance. Section VIII answers common questions, and Section IX concludes.

II. BACKGROUND AND TERMINOLOGY

A. Multi-Agent Orchestration in Agentforce

In Agentforce, multi-agent orchestration is a design in which one agent owns the conversation and delegates work to other agents that own specific domains. The user interacts with only one agent; the others operate behind it. Salesforce lists Multi-Agent Orchestration as generally available in the Summer '26 release [3]. Because availability can vary by edition, region, and license, teams should confirm it in the current release notes for their own org before committing a design to it.

B. The Primary Agent

The primary agent is the front door. It reads the incoming request, determines the intent, and decides which specialist should handle it. Salesforce describes it as the single entry point for user interactions, so the user does not need to choose the right bot or repeat information [1]. A useful analogy is a triage nurse rather than a surgeon: its main responsibilities are understanding and routing, followed by presenting the final answer in a consistent voice.

C. Secondary (Specialist) Agents

Secondary agents perform the domain work. Each one is grounded in its own data and carries its own set of actions. When a specialist finishes, it returns the result to the primary agent, which responds to the user [1]. A billing agent, for example, might only have actions for invoices and payment plans; it never needs to know how to reschedule a field service visit.

D. Subagents Within a Single Agent

One terminology point frequently causes confusion. Inside a single agent, the units of work were previously called topics. The Agent Script reference notes that topics were renamed subagents in April 2026 [4]. As a result, delegation happens at the three levels summarized in Table I.

TABLE I. LEVELS OF DELEGATION IN AGENTFORCE

Level	What it is	Typical use
Subagent	A job inside one agent, with its own instructions, actions, and reasoning	Splitting related work, such as Order Status and Returns
Specialist agent	A separate agent that a primary agent delegates to	A domain with its own owner, data, permissions, or release cycle
External agent (A2A)	An agent on another platform reached through the Agent2Agent protocol	Work that lives in a partner or third-party system

A practical rule of thumb is to start with subagents and promote a subagent to its own agent only when it needs a different owner, different data access, or an independent testing and release schedule.

III. ARCHITECTURE AND MECHANICS OF ORCHESTRATION

A multi-agent request moves through four stages: understand, route, execute, and respond. Fig. 1 follows one request through these stages using an illustrative design with one primary agent and four specialists.

A. Description-Driven Routing

Salesforce states that the Atlas Reasoning Engine selects a specialist by reviewing each agent's description, instructions, and available actions [1]. This has a direct practical consequence: agent descriptions are no longer documentation; they are routing logic. A vague description such as "Helps customers with account questions" will overlap with many other agents. A precise one, such as "Handles invoice disputes, payment plans, and refund status for B2C accounts; does not handle shipping," gives the router clear signals.

B. Context Handoff Between Agents

The purpose of a single front door is that the customer does not repeat themselves. Salesforce's Summer '26 announcement frames this as shared context across channels, so the user never restates the problem to a new agent [2]. Designers should still be deliberate about what is passed. A specialist usually needs identifiers and a few facts, such as a verified contact ID and a case number, rather than the entire conversation and every variable.

C. Deterministic Control with Agent Script

Not every routing decision should be left to a language model. Agent Script is Salesforce's language for defining

agent behavior, and it combines deterministic logic with LLM reasoning [4]. The elements most relevant to orchestration are:

- A `start_agent` block that serves as the entry point and classifies the request.
- `subagent` blocks, each describing one job with its own reasoning and actions.
- `@utils.transition` to `@subagent.<Name>` for handing off between jobs.
- `available` when conditions that hide a route until a rule is satisfied.
- `run` statements and `if` logic for steps that must always execute.

Listing 1 shows a simplified billing router written in the style of Salesforce's published examples [5]. The action and Flow names are illustrative. The key design choice is the `available` when guard: the model cannot route to invoice help until identity has been verified, regardless of how the customer phrases the request. That is an enforced rule rather than a suggestion embedded in a prompt.

D. Actions as the Execution Layer

Agents still act through familiar platform building blocks. In Listing 1 the action targets a Flow; actions can also invoke Apex, prompt templates, and APIs. For administrators, orchestration therefore does not require a move to code. A well-built autolaunched Flow remains one of the safest ways to give a specialist agent a narrow and testable capability.

E. Delegation Beyond Salesforce: A2A and MuleSoft

Salesforce states that Agentforce supports the Agent2Agent (A2A) protocol for delegating tasks to third-party agents [1]. A2A is an open standard that Google launched in April 2025; the Linux Foundation began hosting it on June 23, 2025, with Salesforce among the founding members [10].

In A2A terms, the calling agent is the client and the receiving agent is the remote agent. Agents advertise their capabilities through an agent card and exchange tasks, messages, and artifacts [9]. The remote agent does not expose its internal prompts or tools, which matters when that agent belongs to a partner or vendor.

On the integration side, MuleSoft provides an A2A connector for Mule 4; its Anypoint Exchange listing marks it as a beta connector subject to Salesforce beta terms [11]. MuleSoft Agent Fabric adds an agent registry, a broker for routing, a visualizer, and gateway-based governance for agents across the enterprise [12].

```
start_agent billing_router:
  description: "Greetes the customer and routes billing requests"
  reasoning:
    instructions: ->
      | Identify whether the customer needs identity checks,
      | invoice help, or a payment plan.
  actions:
    go_to_verify: @utils.transition to @subagent.Verify_Customer
      available when @variables.verified == False
    go_to_invoices: @utils.transition to @subagent.Invoice_Help
      available when @variables.verified == True

subagent Invoice_Help:
  description: "Explains invoice lines and opens disputes"
  reasoning:
    instructions: ->
      | Look up the invoice with {@actions.get_invoice}.
  actions:
    get_invoice: @actions.get_invoice
  actions:
    get_invoice:
      target: "flow://Get_Invoice_Details"
```

Listing 1. Simplified Agent Script billing router (illustrative; action and Flow names are examples, syntax adapted from Salesforce's published examples [5]).

The customer talks to one agent; specialists work behind it

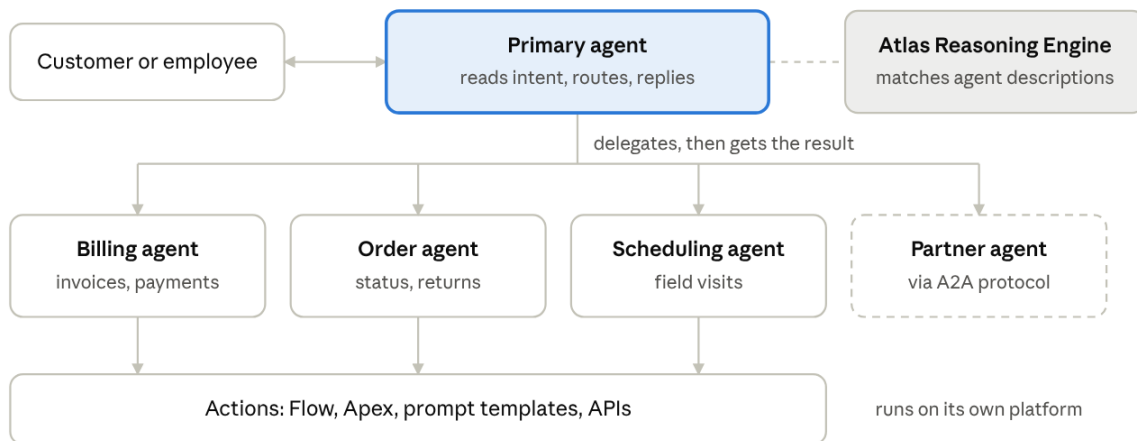


Fig. 1. Illustrative multi-agent design: the customer interacts with one primary agent, which delegates to specialist agents and, through A2A, to a partner agent. Agent names are examples, not Salesforce-provided agents.

IV. PRACTICAL IMPLEMENTATION SCENARIOS

The scenarios below are illustrative designs rather than customer case studies. They show where dividing work across agents pays off and how the pieces map to Salesforce features.

A. Customer Service with a Billing Specialist

1) *Situation:* A subscription company's service agent handles order status well, but billing questions keep derailing it. Billing requires different data, stricter identity checks, and policies owned by a different team.

2) *Design:* The primary service agent keeps greeting, verification, and general FAQ subagents. A separate Billing agent owns invoice lookups, payment plan setup, and dispute creation through autolaunched Flows that read only the Invoice and Payment records its running user may see. Refunds above a set amount create a case for human review instead of completing automatically.

3) *Outcome:* The billing team can change its instructions and actions, rerun its own tests, and deploy without touching the service agent.

B. Employee Help Desk Across HR and IT

1) *Situation:* Employees ask a single Slack agent about laptops, password resets, leave balances, and benefits.

2) *Design:* A primary employee-help agent routes to an IT agent and an HR agent. The HR agent sits behind stricter permissions because it handles personal employee data. The Summer '26 release also introduced an IT Service Domain Pack with more than 50 prebuilt IT agents for Slack, Teams, and the IT Service Desk portal [2], which can reduce how much of the IT side a team builds itself.

C. Sales Handoff from Qualification to Quoting

1) *Situation:* An inbound lead agent qualifies website prospects, who then ask about pricing and product fit.

2) *Design*: The primary agent manages the conversation. A qualification specialist scores fit using lead and account data, and a quoting specialist calls a Flow that builds a draft quote. A human seller, not the agent, approves and sends the final price.

D. Reaching a Partner System Through A2A

1) *Situation*: A manufacturer's warranty data lives in a partner's platform, which runs its own agent.

2) *Design*: The Agentforce primary agent delegates warranty lookups to the partner's agent over A2A, with MuleSoft in the middle to apply gateway policies and logging. Because the Mule 4 A2A connector is labeled beta [11], this pattern suits pilots more than critical production paths until it reaches general availability.

E. Selecting an Orchestration Pattern

Table II maps common situations to a recommended starting pattern.

TABLE II. CHOOSING AN ORCHESTRATION PATTERN

If the situation looks like this	Start with
One team, one data domain, several related jobs	One agent with several subagents
Separate owners, data access, or release cycles	A primary agent plus specialist agents
Work handled by an outside system that has its own agent	A2A delegation, often through MuleSoft
Steps that must always run in a fixed order	Deterministic Agent Script logic or a Flow

V. BENEFITS AND LIMITATIONS

Multi-agent orchestration trades one kind of complexity for another. Each agent becomes simpler, but the system around the agents becomes more important.

A. Benefits

1) *Smaller, clearer agents*: Each specialist carries fewer instructions and actions, which makes its behavior easier to predict and review.

2) *Independent ownership*: Billing, HR, and IT teams can each own their agent's instructions, tests, and release timing.

3) *Tighter data access*: A specialist needs permissions only for its own domain, which supports least-privilege design.

4) *One experience for the user*: The user deals with a single agent instead of choosing among several bots [1].

5) *Reach beyond Salesforce*: A2A support allows a Salesforce agent to delegate to agents on other platforms [1].

The last two benefits reflect how Salesforce positions the feature. How well they hold in practice depends on design quality and testing discipline.

B. Limitations and Trade-offs

1) *More hops, more failure points*: Every handoff is a routing decision that can go wrong, and a misroute can appear as a confident but incorrect answer.

2) *Overlapping descriptions*: If two specialists both claim "account questions," routing becomes unpredictable.

3) *Latency and consumption*: Additional reasoning steps and agent calls can increase response time and platform usage. Teams should review their contract and current Agentforce pricing before scaling out.

4) *Harder debugging*: When an answer is wrong, teams need to know which agent, subagent, and action produced it. Without tracing, diagnosis becomes guesswork.

5) *Agent sprawl*: Creating agents becomes easy, and agents without clear owners accumulate. Each new agent should be treated like a new application with an owner and a retirement plan.

6) *Uneven maturity*: The core feature is GA in Summer '26, but surrounding components, such as the MuleSoft A2A connector, may still be in beta.

VI. IMPLEMENTATION CONSIDERATIONS

The difficult part of multi-agent orchestration is not enabling it. It is drawing sound boundaries between agents and proving that routing works.

A. Defining Agent Boundaries

Before configuring anything in Agentforce Builder, teams should write a one-page charter for each agent. Table III shows an example for a Billing agent.

TABLE III. EXAMPLE AGENT CHARTER FOR A BILLING AGENT

Charter item	Example
Owens	Invoice questions, payment plans, dispute creation
Does not own	Shipping, returns, account changes
Data it may read	Invoice, Payment, related Account and Contact
Actions	Get invoice, create payment plan, open dispute case
Human handoff	Refunds above a set limit; any legal or fraud claim
Business owner	Billing operations lead

The "does not own" entry matters as much as the "owns" entry, because it becomes part of the agent description that the router reads.

B. Combining Low-Code and Pro-Code Development

Agentforce Builder remains the primary configuration surface for most administrators. Agentforce DX extends Salesforce DX so agents can be stored in version control, authored in Agent Script, previewed, tested, and deployed as metadata, with teams moving between Builder and Flow Builder on one side and VS Code and the Salesforce CLI on the other [6]. For multi-agent work, source control is worth the setup: changing one specialist's description can change routing for the whole team, so those changes should be reviewed like code.

C. Designing Narrow, Declarative Actions

Each specialist should receive a few well-named actions rather than one general-purpose action. Autolaunched Flows are a good default for record reads and updates. Apex is appropriate for complex logic, callouts that need custom handling, or bulk-safe processing that Flow handles poorly.

D. Testing Routing Behavior

In a multi-agent design, a correct answer from the wrong agent is still a defect, so tests should verify where a request lands as well as what comes back. Agentforce DX provides CLI commands for this: `sf agent generate test-spec` creates a YAML file of test cases, `sf agent test create` registers the test in an org, and `sf agent test run` executes it [7]. Salesforce's Summer '26 materials also describe an enhanced Testing Center with parallel batch testing, AI-generated test cases, and custom evaluators [3]. A practical routing test set includes:

- **Clear requests** for each specialist, such as "Why was I charged twice?"
- **Ambiguous requests** that could fit two agents, such as "I need to change my order and my card."
- **Out-of-scope requests** that no agent should accept.
- **Guardrail cases**, such as an unverified user asking for invoice details.
- **Handoff cases** where the correct outcome is a human.

E. Production Monitoring

Salesforce describes Agentforce Observability as providing session tracing across agents, quality scoring, and dashboards that can be filtered by subagent, action, and intent [8]. Three signals deserve particular attention: misroutes, handoff rates to humans, and sessions in which users rephrase the same request several times.

F. Phased Rollout

- Start with one primary agent and one specialist.
- Validate both in a sandbox with a routing test set.
- Pilot in one channel with a limited audience.
- Review traces weekly and refine agent descriptions.
- Add the next specialist only after routing is stable.

VII. SECURITY AND GOVERNANCE

More agents mean more identities, more permissions, and more paths for data to travel. Governance must scale with the number of agents.

A. Least Privilege per Agent

Each specialist should receive only the object, field, and record access its job requires. The Billing agent does not need to read HR cases, and the HR agent does not need payment data. Teams should review the running user and permission sets for every agent and remember that Flows running in system context can bypass sharing rules they believe are in force.

B. Minimal Context Sharing

When the primary agent hands work to a specialist, it should pass identifiers and the minimum facts required. Forwarding full transcripts can expose personal or payment details to agents that have no need for them.

C. Governing External Agents

An agent reached through A2A sits outside the org's security model and should be governed like any other third-party integration:

- Confirm how it authenticates and what it may do with the data it receives.
- Route traffic through a gateway that applies policies and logging; MuleSoft Agent Fabric uses Flex Gateway to enforce policies on agent-to-agent and agent-to-tool traffic [12].
- Check contracts and data residency before sending regulated data to another vendor's agent.

D. Human-in-the-Loop Controls

Teams should decide in advance which outcomes an agent may complete alone and which require a person. Refunds above a threshold, contract terms, eligibility decisions, and actions with legal or financial consequences are common candidates for human approval. These controls belong in the design, through approval processes, case creation, or escalation to a service representative, rather than in a sentence inside a prompt.

E. Platform Guardrails and Operational Oversight

Salesforce's Summer '26 materials list rate limiting, access management, observability, and built-in security guardrails as enterprise controls for multi-agent orchestration [3]. These are a base layer; an organization's own ownership model, review cadence, and audit process determine whether the system remains trustworthy. A lightweight governance checklist includes:

- A named business owner and technical owner for every agent.
- A written scope for every agent, including what it does not do.
- Permission reviews and routing test runs before each release.
- Documented and tested human handoff rules.
- Scheduled review of production traces and a retirement plan for unused agents.

VIII. FREQUENTLY ASKED QUESTIONS

A. What is Agentforce multi-agent orchestration?

It is a way to build a team of Agentforce agents in which a primary agent talks to the user and delegates tasks to specialist agents. Salesforce lists it as generally available in the Summer '26 release [3].

B. How does the primary agent choose a specialist?

Salesforce states that the Atlas Reasoning Engine compares each agent's description, instructions, and available actions to choose the best fit [1]. Clear, non-overlapping descriptions are the most important factor a team controls.

C. How does a subagent differ from a specialist agent?

A subagent is a job inside one agent; Salesforce renamed topics to subagents in April 2026 [4]. A specialist agent is a separate agent that receives work from a primary agent.

Subagents suit related jobs under one owner, while separate agents suit differing ownership, data access, or release cycles.

D. Can Agentforce agents work with agents outside Salesforce?

Yes. Salesforce states that Agentforce supports the A2A protocol for third-party agents [1]. MuleSoft also provides an A2A connector, currently labeled beta [11].

E. Is Apex required for a multi-agent design?

Not necessarily. Many specialist actions can be built with Flow, and agents can be configured in Agentforce Builder. Apex and Agent Script become valuable for complex logic, strict deterministic control, or source-controlled development.

F. How can routing be tested?

Build a set of utterances covering clear, ambiguous, out-of-scope, and guardrail cases, and verify routing as well as answers, using the Testing Center or Agentforce DX commands such as `sf agent test run` [7].

G. When should multiple agents be avoided?

When one team owns the work, the data is shared, and the jobs are closely related, a single agent with several subagents is usually simpler to build, test, and govern.

IX. CONCLUSION

Multi-agent orchestration gives Salesforce teams a structured way to grow Agentforce beyond a single overloaded agent. A primary agent keeps the experience simple for the user, while specialist agents keep each domain small, owned, and testable. Agent Script adds deterministic guardrails where model-driven routing is not appropriate, and A2A extends delegation to agents outside Salesforce.

The capability itself is only part of the work. Results depend on precise agent descriptions, narrow actions, least-privilege access, routing tests, and regular review of production traces. Teams that start with one primary agent and one specialist, prove the routing, and add agents only when each has a clear job and a clear owner are best positioned to benefit while keeping risk under control.

REFERENCES

1. Salesforce. (2025). Agentforce Multi-Agent Orchestration. Salesforce. <https://www.salesforce.com/agentforce/multi-agent-orchestration/>
2. Salesforce. (2026, May 11). Summer '26 Release: 10 Innovations Bringing the Agentic Enterprise to Life. Salesforce Newsroom. <https://www.salesforce.com/news/stories/summer-2026-product-release-announcement/>
3. Salesforce. (2026). Summer '26 Release in a Box. Salesforce. <https://www.salesforce.com/en-us/wp-content/uploads/sites/4/documents/PDF/release-in-a-box-summer-26-v3.pdf>
4. Salesforce. (n.d.). Agent Script Reference. Salesforce Developers. <https://developer.salesforce.com/docs/ai/agentforce/guide/ascript-reference.html>
5. Salesforce. (n.d.). Agent Script Example: Customer Support. Salesforce Developers. <https://developer.salesforce.com/docs/ai/agentforce/guide/ascript-examples-customer-support.html>
6. Salesforce. (n.d.). Build Agents with Agentforce DX. Salesforce Developers. <https://developer.salesforce.com/docs/ai/agentforce/guide/agent-dx.html>
7. Salesforce. (n.d.). Test an Agent with Agentforce DX. Salesforce Developers. <https://developer.salesforce.com/docs/ai/agentforce/guide/agent-dx-test.html>
8. Salesforce. (n.d.). Agentforce Observability. Salesforce Blog. <https://www.salesforce.com/blog/agentforce-observability/>
9. Salesforce. (n.d.). What Is the Agent2Agent (A2A) Protocol? Salesforce. <https://www.salesforce.com/agentforce/ai-agents/agent2agent-protocol/>
10. Linux Foundation. (2025, June 23). Linux Foundation Launches the Agent2Agent Protocol Project to Enable Secure, Intelligent Communication Between AI Agents. <https://linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents>
11. MuleSoft. (n.d.). A2A Connector - Mule 4. Anypoint Exchange. <https://www.anypoint.mulesoft.com/exchange/com.mulesoft.connectors/mule4-a2a-connector>
12. Salesforce Architects. (n.d.). MuleSoft Agent Fabric Deep Dive. Salesforce Architects. <https://architect.salesforce.com/fundamentals/mulesoft-agent-fabric-deep-dive>