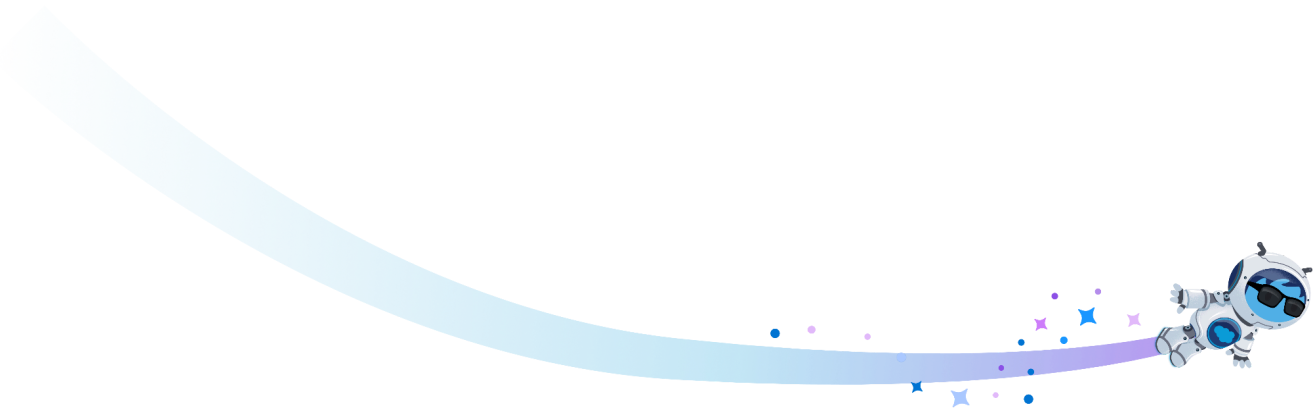
Agentforce Implementation Guide

Customer-Facing Agents





© Copyright 2000–2025 Salesforce, Inc. All rights reserved. Salesforce is a registered trademark of Salesforce, Inc., as are other names and marks. Other marks appearing herein may be trademarks of their respective owners.

Last Updated: Apr 16, 2026

Table of Contents

Get Started with This Guide..... 5
- What are Agents?..... 5
- The Agent Development Lifecycle..... 5
- A Real World Example..... 6
- Sign Up for a Salesforce Org..... 6

Ideate Your Agentforce Solution..... 8
- Identify Your Use Case..... 8
- Define Your Agentforce Use Case..... 9
- Define the Technical Requirements..... 12
- Salesforce Considerations for Agentforce Projects..... 13
- Define the Agent's Guardrails..... 14
- Design Your Agent..... 17
- A Strong Foundation..... 20

Build Your Agent..... 21
- Review Required Editions and Permissions..... 21
- Set Up Einstein Generative AI..... 23
- Create Your Agent..... 24
- Get to Know Agentforce Builder in Agentforce Studio..... 25
- Customize Your Agent's Settings..... 29
- Ground the Agent's Responses in Makana's Information..... 30
- Configure Your Agent with Subagents and Actions..... 32
- Add Permissions to the Agent User..... 58

Test Your Agent..... 61
- Planning and Preparation..... 62
- Core Testing..... 67
- Iteration and Refinement..... 85

Deploy Your Agent..... 92
- Before You Begin..... 92
- Learn About Connections..... 92
- Route Messaging Conversations to and from the Agent..... 93
- Prepare Your Service Team For Escalated Conversations..... 104
- Deploy the Agent to a Website..... 106
- Deploy the Agent to Staging or Production..... 110

Monitor Your Agent..... 112
- Before We Begin..... 112

Introducing Agentforce Observability..... 112
Set Up and Configure Agentforce Analytics, Optimization, and Session Tracing..... 113
Review Agent Performance in Agentforce Studio..... 114
Dig Deeper into Agent Performance with Agentforce Optimization..... 115
Review Sessions and Intents to Identify Areas of Improvement..... 116
Monitoring Strategy Methods..... 117
Building a Monitoring Strategy..... 119

Get Started with This Guide

In this guide, you'll learn how to launch a customer-facing agent using the Agentforce Service Agent template. This guide walks you through the process of creating an agent by following the five major stages of agent development: Ideate, Build, Test, Deploy, and Monitor. You can use this guide for a variety of solutions:

Goal	Instructions
I just want to build and launch an agent fast.	Skip the Ideate chapter and use the Build, Test, and Deploy chapters to launch an agent. After launch, you can review the Monitor chapter to refine your agent.
I'm new to building agents.	Walk through the entire guide in order.
I need help building out a continuous development cycle.	Review the Test and Monitor chapters in order to refine your agent.

What are Agents?

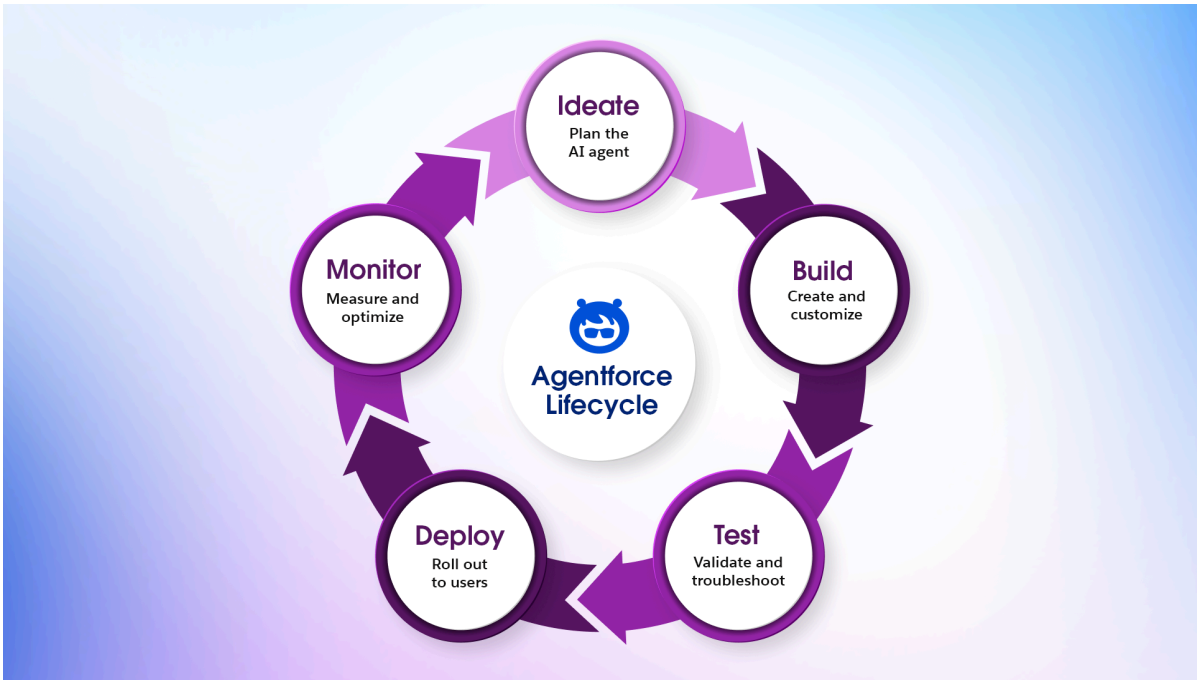
Agents are goal-oriented, autonomous AI assistants that perform tasks and business interactions. They can initiate and complete a sequence of tasks, handle natural language conversations, and securely provide relevant answers drawn from business data.

Some agent types are best at assisting and collaborating with a Salesforce user in the flow of work. Other types can also act on behalf of a user or customer, based on the use cases and guardrails an admin specifies. Some agent types are available in your Salesforce interface, and you can add some to your customer channels.

Agents respect standard Salesforce access controls such as licenses and permissions, so they always act securely. Agents also include AI guardrails designed to help meet ethical standards and are integrated with the Einstein Trust Layer, a secure AI architecture natively built into the Salesforce Platform.

The Agent Development Lifecycle

The development lifecycle for an AI agent generally follows five key stages: ideate, build, test, deploy, and monitor. Here's an overview of what happens during each stage.



- **Ideate**: Define the agent's business value use case, scope, and required capabilities. Address concerns and risks upfront by integrating risk management into the planning process to ensure the agent meets ethical, legal, regulatory, and security requirements.
- **Build**: Create the agent, customize it, and add the necessary data sources.
- **Test**: Validate the agent's performance, accuracy, and adherence to security and compliance standards. Troubleshoot and resolve any issues.
- **Deploy**: Configure the agent's connections and make the agent available to users on the right channels.
- **Monitor**: Measure the agent's performance and health, collecting feedback and data to identify areas for optimization and continuous improvement.

This guide walks you through each stage of the agent development lifecycle.

A Real World Example

In this guide we follow a fictional company, Makana Medical Devices, as they work step-by-step to implement an Agentforce Service Agent in their organization.

Makana Medical Devices is a medical device company that develops, manufactures, and distributes a wide range of diagnostic, treatment, and patient monitoring instruments. They've already integrated generative AI into their Salesforce ecosystem and are now looking to build their first agent with Agentforce to enhance customer support.

Sign Up for a Salesforce Org

To follow along with the steps in this guide, you can sign up for a free Salesforce Developer Edition org.

1. Go to the Developer Edition signup page.
2. Fill out the form and submit it.
3. Look for a welcome email from Salesforce in your inbox.
4. Verify your account using the verification link in the email.
5. When prompted, set a new password and select a security question for your Developer Edition org. Then complete the login process.
6. Keep track of your username and password so that you can log in again later from the Salesforce login page.

When you're done signing up for a Salesforce org, you can start the agent development process.

Ideate Your Agentforce Solution

The first step in developing an AI agent is ideating your Agentforce solution. Planning an AI agent is crucial for several reasons. It ensures that your AI project aligns with your company's overall AI strategy and business goals. It also helps you define the agent's work and assess its potential impact on the organization.



Addressing concerns and risks upfront by integrating risk management into the planning stage can accelerate projects and ensure they meet ethical, legal, regulatory, and security requirements. Without proper planning, an agentic AI project might fail to deploy or be scrapped entirely.

When planning an agent, here are the things to consider:

- Use case definition and scope
- Data and technical requirements
- Risks and guardrails
- Business processes and agent design

Although careful planning is important, try to strike a balance between experimentation and planning. When you're getting started with Agentforce, hands-on learning and prototyping are essential for understanding AI agents and testing out ideas.

In this chapter, we walk through an ideation exercise for Makana Medical Devices. If you want to get straight into creating an agent, skip ahead to the Build chapter.

Identify Your Use Case

An Agentforce use case is an application of AI technology where an AI agent takes an action or series of actions that accomplish a goal or job-to-be-done on behalf of your employees, customers, or organization.

As you begin collecting ideas for Agentforce use cases – whether through crowdsourcing, business process review, user research, or market analysis – you might notice that some ideas lend themselves well to autonomous AI applications, and some don't. When evaluating potential use cases for autonomous AI, consider the following questions:

- **Value**: Why are you delegating this work to an AI agent? Is the agent faster or more accurate? Will it provide a better experience?
- **Work**: Can you describe the work the AI agent will do? Do you fully understand all the business processes involved in that work? Are there business processes that your agent must follow, that are particularly complex, or must follow a specific sequence?
- **Decision-making**: Does the work involve decisions and steps that can be completed without direct human input or judgment? Are there well-defined policies, rules, and constraints that the AI agent can follow alone?
- **Risk**: Can the AI agent operate within the security, legal, ethical, and regulatory requirements that apply to the work?
- **Data**: Is the data capable of supporting the work the AI agent will do?

As you decide on your answers, it's important to understand what agents can do. Often the easiest way to get started is by building an agent and seeing how its capabilities match with the goals you have for implementation.

Makana's Business Challenge

Makana's service team receives a high volume of repetitive inquiries about product information, troubleshooting, and company policies. These inquiries consume a significant amount of time, leading to longer wait times for customers and a heavy workload for the support team. By implementing an AI service agent, Makana aims to offload these routine tasks, allowing support reps to focus on complex, high-value cases that require a personal touch and expert knowledge. This will improve customer satisfaction and operational efficiency.

Makana comes up with a few use case ideas for their agent. The two most promising ideas are: responding to common customer questions and streamlining case management.

Define Your Agentforce Use Case

After identifying your use case ideas for Agentforce, flesh out each of the use cases so your organization can assess and prioritize them. Remember, this stage of the planning process is about the project goals, not the technical solution.

Identify the Jobs To Be Done

First, describe the work the AI agent will do. Many organizations use the Jobs to Be Done framework to outline the role of the agent and the tasks it will perform. Be sure to think deeply about the work and its expected outcomes. It's an essential step in understanding how the agent can impact your organization, customers, and employees.

Here are the jobs to be done for Makana's top two use case ideas.

Work or Job to be Done	Tasks
FAQs	Answer customer inquiries related to:

	• Company policies • Product information • Troubleshooting
Case Management	Assist customers with cases: • Look up cases • Summarize case details • Add a case comment • Update the case status • Create a new case

Determine the Scope

After identifying the work you want the AI agent to do, the next step is figuring out the right amount of work. What's the appropriate scope? Your minimum viable product (MVP) should be the smallest unit of work that it makes sense to deliver. An iterative approach allows you to validate your assumptions, demonstrate value, manage the level of risk, and develop a plan for scaling the AI solution. Gradually expand your scope and use cases as your organization evolves in AI maturity.

Let's take a look at how Makana might scope its use cases for FAQs and Case Management.

Work or Job to be Done	Tasks
FAQs	Version 1 (MVP): • Handle FAQs for a single, high-volume product: the CGM-3000, Makana's continuous glucose monitor. • Support English language customers. Capabilities Added to Version 2: • Handle FAQs for the rest of their patient monitoring product lines. • Add support for additional languages. Capabilities Added to Version 3: • Handle FAQs for other types of products, such as treatment and diagnostic devices.
Case Management	Version 1 (MVP): • Look up cases. • Summarize case details. • Create a new case.

	Capabilities Added to Version 2: • Add a case comment. Capabilities Added to Version 3: • Update the case status.

Define the Business Value

When you fully understand the scope of the work the AI agent will perform, you can establish the business value of the use case. Be sure to set specific, measurable goals and focus on outcomes.

Here's the business value for Makana's use cases, which aligns with the company's top objectives.

Work or Job to be Done	Tasks
FAQs	• Improve the deflection rate by 40% • Improve customer satisfaction scores by 10%
Case Management	• Reduce case handling time by 25% • Increase the accuracy of initial case routing by 15%

Evaluate Data Readiness

Next, evaluate your data readiness for the use case. For AI agents to perform optimally, they must be powered by trustworthy, high-quality data that's relevant to the business context. Don't commit to a use case until you've confirmed that the data can support it.

Makana's AI implementation team does a cursory investigation into their data readiness for the FAQs and Case Management use cases, and here's what they find.

Use Case	Data Sources	Data Readiness
FAQs	PDF	The documentation for Makana's CGM3000 product is stored in a PDF. The PDF is clear and well-formatted, and it doesn't have any complicated tables that might confuse the large language model (LLM).
Case Management	Salesforce CRM	Makana uses Service Cloud, and they have strong data governance practices in place.

		The data quality in Service Cloud is sufficient for the case management use case.

Keep in mind that when you're defining your use case, you're not conducting a full-scale data audit. It's more of a preliminary gut-check to determine the use case's feasibility and ease of implementation. You'll dig deeper into data readiness when you define the technical requirements for the project.

Assess and Prioritize the Use Cases

After defining your Agentforce use cases, your organization can assess the feasibility and impact of the projects and then incorporate them into your prioritized AI roadmap. See AI Strategy for more information about the factors to consider when refining your AI backlog and which prioritization frameworks you can use.

Define the Technical Requirements

When planning an AI agent, spend some time gathering technical requirements up front so you can build the right solution faster. Key factors to explore include: data, channels, routing and escalation, and security controls.

Data

Your organization needs to be data-ready to start an AI project, which means the data for the project is accurate, complete, available, accessible, and securely governed. When defining the data requirements for your Agentforce use case, be sure to assess the data quality, migrate and integrate data, establish data governance, and create a plan for data analytics. To learn more about data requirements, see AI + Data: Project Planning.

Channels

Consider the channels where the AI agent will operate. How will the agent engage with employees or customers? How will the agent be presented to users? See Deploy Your Agent to Channels to learn more about the different options.

Makana's new agent will be deployed on an Enhanced Web Chat channel that they can embed on an Experience Cloud site.

Keep in mind that you can prototype an AI agent in your sandbox before configuring the channels if you want to test out your ideas for the agent's design. But you need to establish your channel strategy before deploying to production, and thoroughly test the agent's performance on each channel.

Routing and Escalation

AI agents sometimes need to escalate conversations to a service representative. Some reasons for escalating to a rep can include company policy, brand requirements, security measures, risk management, or regulatory compliance.

During the planning process, define all the ways in which human decision-making and oversight will occur during the AI agent's work. Clear guidelines about when and how reps should step in will make it easier for your organization to configure the right guardrails as you start prototyping your agent.

Security Controls

When you start planning your AI agent, consider permissions and access for admins, employees, customers, and the AI agent itself.

- **Admin Access**: To create and manage AI agents in Salesforce, you need the **Manage AI Agents** user permission and the required permissions for your agent type, or the **Customize Application** user permission.
- **Customer Access**: If your agent is deployed to external channels, such as your website, there are probably certain actions you don't want an AI agent to perform on behalf of customers unless they're verified. The way you design your authentication solution for Agentforce depends on the security and identification requirements for your particular use case. See Maintain Trust with Agentforce Actions.
- **Agent Access**: Some AI agents, such as service agents, operate as an agent user, and the agent user has a dedicated user profile and role in Salesforce. The actions the AI agent can take depend on the permissions it's assigned. See Best Practices for Agent User Permissions to find out how to control what data your AI agent can access.

Salesforce Considerations for Agentforce Projects

Your AI agent must coexist with your current Salesforce configuration and architecture. Consider Salesforce-specific factors that can influence your Agentforce implementation.

Licensing and Provisioning

Each Agentforce project is unique, so the Salesforce products and features you need vary depending on your use case. Check with your Salesforce account executive to confirm the licensing requirements for the specific AI agent you want to build.

Billing and Consumption

Agentforce uses a consumption-based pricing model. For more information on how usage is billed, see Agentforce Pricing. You can also refer to your contract or contact your account executive.

Requirements

To use Agentforce, these products and features must be enabled in your Salesforce org.

- Lightning Experience
- Einstein Generative AI
- Data Cloud

Existing Automation

With Agentforce, your AI agents use agent actions as tools to accomplish specific tasks. Those agent actions are built on top of existing Salesforce Platform technology, such as flows, Apex, and prompt templates. Start cataloging any existing automations related to your use case that you might be able to repurpose for your AI agent.

Define the Agent's Guardrails

Autonomous AI has inherent risks that you can mitigate including security threats, data breaches, reputational harm, financial loss, bias, hallucinations, and issues with transparency and accountability. The good news is by building agents on the Agentforce platform, you have access to built-in guardrails and controls to help mitigate risk. When planning an AI agent, it's important to discuss potential risks with key stakeholders for your Agentforce use case and use the platform to plan risk mitigation strategies. By creating a robust guardrail strategy, your team can be confident that the solution is addressing the most common AI risks.

Frame Conversations About Risk

As you approach conversations in your organization about AI risk, you can use the People, Business, Technology, and Data Framework. These categories and considerations can help you come up with possible risks and concerns related to your Agentforce project.

Category	Considerations
People	• **Empowerment**: Roles and responsibilities, hiring, training, and upskilling • **Culture and practice:** Human-centered AI design, change management, adoption
Business	• **Value**: Benefits, objectives, KPI, and metrics • **Operations**: Org structure, capability management, processes and workflows, AI governance, DevOps strategy
Technology	• **AI tooling**: AI infrastructure, applications, APIs, prompts, security safeguards • **AI models**: Model selection, training considerations, management, cost

Data	• **Quality**: Fit for use, accuracy, completeness, accessibility, recency, and more • **Strategy**: Data management, infrastructure, governance, analytics

Identify Risks and Concerns

Makana uses this framework to discuss risks related to their use cases. Let's take a look at the risks and concerns that Makana's stakeholders identified for each category.

Category	Risks
People	• **Rejection**: Customers don't want to talk to the agent because they don't trust it or because they're unsure if they're allowed to use AI. • **Abuse**: Customers are hostile to the agent or try to manipulate it. • **Culture**: Fears about the potential impact of AI on service jobs affect employee morale.
Business	• **Fit**: Agent's scope doesn't fit properly into the business organization or team processes. • **Reporting**: Current team KPIs are invalidated by the introduction of an AI agent to do some of the work. • **Incentives**: Compensation and reward structures are impacted by agent work redirection. • **Operations**: Process for escalation is unclear, inefficient, or frustrating. • **Agent Performance**: Appropriate company policies don't correctly influence the AI agent responses. • **Liability**: Financial and reputational damage due to lawsuits or regulatory fines resulting from potential AI errors.
Technology	• **Accuracy**: Hallucinations degrade the quality of responses or endanger the wellbeing of patients. • **Reliability**: Variability of the agent's generated responses is too broad. • **Audit**: Technology operations can't track the accuracy of agent responses. • **Latency**: Agent can't achieve timely responses.
Data	• **Access**: Data permissions aren't understood or enforced; data might be exposed to customers. • **Privacy**: The required data can't be used according to the privacy policy. • **Compliance**: It's unclear if any customer contractual constraints apply to the data; for example, maybe data can't leave the customer's business country.

	• **Fit for purpose**: Data isn't aligned with the agent's objective, or data rights aren't aligned to the use case. • **Ethics**: Bias in model data could generate inappropriate responses.

Note that this list isn't exhaustive, and every use case involves its own unique risks and concerns.

Define Risk Mitigation Strategies

After cataloging the risks and concerns, come up with mitigation strategies for each risk. As you brainstorm potential guardrails, categorize each guardrail to designate whether it's related to people, business, technology, or data.

Here are examples of potential guardrails for two of the risks that Makana identified.

Category	Risk	Potential Guardrails
People	**Customer Rejection**: Users don't want to talk to the agent because they don't trust it.	• **People guardrail**: Create a communication strategy and conduct education briefings for customers. • **Technology guardrail**: Design the agent to be transparent about the fact that it's AI. • **Technology guardrail**: Configure a welcome message for the agent that sets the right expectations about its capabilities and how it can assist.
Business	**Escalation Issues**: Handoffs from the agent to service reps are inconsistent, inefficient, or frustrating to customers.	• **Business guardrail**: Define the criteria and context for escalation from AI to service reps. • **Technology guardrail**: Configure Agentforce so that a summary of the agent's prior interaction is handed off to the service rep. • **Technology guardrail**: Build an escalation path to a representative triggered by keywords, language, or requests. Clearly state them in the agent instructions.

When you're done with the risk mitigation exercise, document the risks and guardrails for your use case. Capturing risk mitigation activities is important for regulatory compliance, useful for internal audits, and will build trust with key stakeholders.

Design Your Agent

The Makana team is making great progress with their AI agent. They defined the Agentforce use case, considered the project's requirements, and developed some risk mitigation strategies. Now it's time to start designing the agent and thinking about how it will be configured.

Map Business Processes

When planning an AI agent, it's important to define the work the agent will do. One way to define the work is to map the business processes related to your Agentforce use case.

Process mapping is a visual representation of the steps in a business process from start to finish, typically using diagrams like flow charts or swimlanes. It helps define the work an AI agent will do by depicting the sequence of steps, start and end points, and required inputs and data.

As you outline the business processes you want to delegate to an agent, consider these questions.

- What goal is the user trying to accomplish?
- How might the user express their goal or intent? How would they phrase their questions or requests?
- What processes are involved in achieving the goal?
- When does each process begin and end? What steps need to be taken? In what order?
- What company policies, rules, and guidelines apply to the process? What guardrails are relevant to each step?
- At what points in the process are decisions made and how?
- What processes must be followed in order? What processes are more flexible?
- What clarifying or follow-up questions might the agent need to ask?
- When would a user need to confirm something before a step proceeds?
- What data is required? What's optional?
- What input do you need to collect from users? Does that input need to be in a specific format or syntax?
- What are the outcomes of each step or process? How is the output used? How is success measured?
- Are there situations when specific language should be used when communicating with users?
- What questions or paths of conversation should never be pursued?
- Under what conditions should the conversation be escalated to a live service rep?

Even if you don't create formal diagrams for each of your business processes, it's still important to think deeply about the work the agent will do so that you design the right solution.

Identify the Subagents and Actions

After determining your agent's use case and the work the agent will do, you can identify the right subagents and actions for the agent. Agents are made up of subagents, which define the different

jobs an agent can do. Subagents also contain a set of actions, which are the tools the agent can use to do its job.

Salesforce provides some standard subagents and actions for Agentforce out of the box, so you can get up and running quickly. But you can also create custom subagents and actions to give your AI agent additional abilities, so it can perform tasks specific to your business.

When identifying the subagents and actions for your agent, there are two possible approaches: top-down and bottom-up.
In the **top-down approach**, first identify the subagents, then define the relevant actions for those subagents.
In the **bottom-up approach**, first list all of the individual actions for your agent, then group them into related subagents.

If you've identified the jobs to be done and the relevant tasks, the top-down approach will work best. For an example of the bottom-up approach, see Identify the Subagents and Actions in Salesforce Help.

Makana's Subagents and Actions

Makana's implementation team reviewed all the standard subagents and actions for Agentforce, and they found two standard subagents that will work for their use cases: General FAQ and Case Management. The team will need to make some tweaks to the standard subagents and actions, but the ability to use out-of-the-box agent assets will give them a great head start with the development process.

Design the Subagents and Actions

After identifying the right subagents and actions for your AI agent, start planning how to build them. Following best practices when designing subagents and actions helps your agent perform reliably and effectively.

As you plan your subagents and actions, simultaneously build and test your AI agent in a sandbox environment. Try different approaches to figure out what works and what doesn't. The key is to use prototyping and continuous refinement to guide your planning and development efforts.

Design the Subagents

Subagents are an important component of Agentforce because they define the AI agent's goals and provide the context and direction it needs to achieve those goals. Makana plans to customize several standard subagents to get their agent up and running quickly, but you can create custom subagents that are tailored to your business.



Learn More: *Check out this video to learn how to create a custom subagent. For more information about designing subagents, review the different parts of a subagent and follow the best practices for subagent instructions.*

Design the Actions

If none of the standard agent actions suit your needs, you can build custom actions for your AI agent.

When you create a custom action, you build it on top of existing platform functionality that you want to make available in Agentforce – invocable and REST Apex classes, autolaunched flows, prompt templates, external services, and MuleSoft APIs. In Agentforce, that underlying functionality is called a reference action.

Design Considerations for Reference Actions

Here are some factors to consider when approaching the design of the underlying reference actions for your agent actions.

Deterministic or Prompt-Based

When developing the underlying platform functionality for your agent actions, first review the business processes and tasks related to your use case. Then decide whether the process or task should be deterministic, prompt-based, or a combination of both approaches.

- **Deterministic**: Uses an invocable Apex class, REST Apex class, or autolaunched flow to generate output. Actions based on flows or Apex are deterministic and use business logic and rules to produce a consistent outcome.
- **Prompt-based**: Uses one or more prompt templates to generate output. A prompt-based action lets you control how a response is written. It also lets you use reasoning and generative capabilities of an LLM. For example, to generate a summary or perform sentiment analysis, you need to use a prompt template as a reference action. Prompt templates are also used to ground an agent in data, such as knowledge or external system data.

Keep in mind that an action can combine both deterministic and prompt-based approaches. For example, an Appointment Management subagent might include an action that cancels appointments. When a user asks to cancel their appointment, the flow-based action is triggered to complete the cancellation. At some point during that flow, the agent could also launch a prompt template that summarizes the user's reason for canceling.

Inputs and Outputs

In Agentforce, each agent action must have at least one input, which means the underlying flow, Apex, or prompt template must also have at least one input. For example, to look up an order, the input might be the customer's email or order number. During a conversation, the AI agent has the autonomy to gather information and decide if it has all the details required for it to trigger the action and pass in the input.

Each action must also have at least one output. The way you build the action determines what the output is, how it's used, and whether and how it's displayed to users in the conversation. Don't be

afraid to experiment with test actions that aren't fully implemented – they can be a great way to see how your ideas work in execution.

A Strong Foundation

Now you know that planning and designing an AI agent is all about laying down a strong foundation with creative ideation and thorough planning. You need to identify your use case and objectives, think about your data strategy, consider the user experience, and outline your project's technical requirements. Address potential risks and define your business processes to make sure the AI agent aligns with your organization's operational, security, legal, ethical, and regulatory requirements.

Don't make the mistake of taking a "waterfall approach" to agent design. You don't have to develop and deploy with a linear, phased plan. As you consider your Agentforce solution from all the necessary angles, get hands-on, and begin prototyping the AI agent in your sandbox environment. That way, you don't invest too much time up front on a plan that might not work out in the end.

Now that you've planned out and experimented with a new AI agent, let's start making it a reality and learn how you can build an agent that demonstrates real value to your business.

Build Your Agent

Now that Makana has completed their ideation stage, it's time to build the agent. In this stage, you create an Agentforce agent and customize it to better fit Makana's customer support use cases.



Here's what you'll cover.

- Prepare your org for Agentforce by verifying Data Cloud, setting up Einstein generative AI, and enabling Agentforce.
- Create an Agentforce Service agent from the default template and customize key settings.
- Create an Agentforce data library so your agent can access Makana's knowledge and deliver personalized answers to customer questions.
- Configure the agent with subagents and actions. Customize standard subagents that handle frequently asked questions and case management, and create a flow-based custom action.
- Get hands-on experience with the deterministic features of Agent Script that help you build reliable, enterprise-ready agents, including:
  - Conditional expressions
  - Transitions to other subagents
  - Variables to represent conversational states
  - Filters on agent actions
  - Direct references to resources



Note: *You'll need a developer org to fully build, test, and deploy your agent according to this guide. If you haven't already, sign up for a Developer Edition org.*

Review Required Editions and Permissions

Before you get started, review the editions and permissions required to complete the tasks in this guide. The **Admin Permissions Needed** list covers all of the permissions, permission sets, and permission set licenses you'll need to build, test, deploy, and monitor your agent.

REQUIRED EDITIONS
Available in: Lightning Experience

Agentforce Service Agent is available in: Enterprise, Performance, Unlimited, and Developer Editions with Foundations or Agentforce 1 Editions. Access to some standard agent actions requires additional add-on licenses.

Enhanced Chat is available in: Enterprise edition for Service Cloud with the Digital Engagement add-on, Enterprise edition for Sales Cloud with the Sales Engagement add-on, Unlimited edition for Service or Sales Cloud, and Developer edition

ADMIN PERMISSIONS NEEDED	
User Permissions	• Create and Set Up Experiences • Configure Messaging (included in the Messaging User permission set license) • Customize Application • Manage Agentforce Service Agents (included in the Agentforce Service Agent Configuration permission set and the Agentforce Service Agent Builder permission set licence) • Manage AI Agents (included in the Agentforce Service Agent Configuration permission set and the Agentforce Service Agent Builder permission set licence) • Manage Flow • Manage Public List Views • View Setup and Configuration
Permission Sets	• Agentforce Service Agent Configuration • Data Cloud Architect • Prompt Template Manager • Service Cloud User Required for Analytics & Optimization • Access Agentforce Optimization • Tableau Next Limited Consumer
Permission Set Licenses	• Agentforce Service Agent Builder • Data Cloud • Einstein Prompt Templates • Enhanced Chat User • Messaging User • Service User (automatically assigned when you add the Service Cloud User permission set)

Set Up Einstein Generative AI

Agentforce is integrated with Data Cloud and generative AI features so you can build observable and trusted agentic experiences. We'll verify and set up these features to prepare your org for your first agent.

Verify Data Cloud in Your Org

Data Cloud is required for essential Einstein generative AI and Agentforce functionality, such as the Trust Layer, agent event logs, and consumption billing tracking.

If you're preparing a free Agentforce Developer org, Data Cloud is usually provisioned and enabled for you. We recommend verifying your Data Cloud installation. From Setup, in the Quick Find box, enter `Data Cloud`, and then select **Data Cloud Setup Home**. Verify that a home org has been created for you on the Data Cloud Setup Home page, with a home org ID, a home org instance, and a tenant endpoint.

If you aren't using a free Agentforce Developer org, or if Data Cloud installation wasn't successful, turn on Data Cloud.

1. From the Setup Menu, select **Data Cloud Setup**.
   If you don't see this option, refresh your page, or log out and then log back in with your admin user credentials.
2. To enable Data Cloud, click **Get Started**.

Setup can take 15-60 minutes. After setup is finished, you can continue on to the next section.

Enable Einstein Generative AI

Turn on Einstein to use generative AI features, including Agentforce, across Salesforce. If you're completing this guide in a new org, this setting is turned on by default.

1. From Setup, in the Quick Find box, enter `Einstein Setup`, and then select **Einstein Setup**.
2. Enable **Turn on Einstein**, if it's not already turned on.
3. Refresh the page.

After you turn on Einstein, it can take a few minutes to sync Einstein and Data Cloud.

Verify Einstein Trust Layer Settings

Build trust in your agents and other generative AI features. Use the Einstein Trust Layer to personalize data privacy controls that are integrated into the end-user experience.

From Setup, in the Quick Find box, enter `Einstein Trust`, and then select **Einstein Trust Layer**.

Einstein Trust Layer is enabled in your Developer org automatically and includes toxicity detection in LLM responses by default. For Makana's use case, we can continue with the default settings. But if you want to make changes, individual entities or categories can be turned off or on from this page. It can take up to a few minutes for the changes to take effect.



Note: *Pattern-based and field-based data masking for large language models (LLMs) is disabled for Agentforce. Einstein Trust Layer includes several policies and features to help protect sensitive data from misuse or leaks beyond data masking. All information sent to an LLM outside of the Salesforce trust boundary is subject to our zero data retention contract with the LLM provider. Information sent to the LLM is not retained, viewed, or used for training by the provider after the generated response has been sent back to Salesforce. Learn more in Data Masking Limitations in Agentforce.*

Enable Agentforce

Turn on Agentforce to start building, customizing, testing, and launching AI agents.

1. From Setup, in the Quick Find box, enter `Agent`, and then select **Agentforce Agents**.
   If you don't see Agents in Setup, verify that Einstein Generative AI is enabled for your org.
2. To enable Agentforce, turn on **Agentforce**.

After you enable Agentforce, refresh the page to see the New Agent button. The New Agent button launches the guided setup for creating an agent. If you don't see the New Agent button, verify your permissions.

Create Your Agent

Now that you've enabled Agentforce, you're ready to create your agent. Previously, you'd continue in Setup. Now you'll head over to the Agentforce Studio app instead. The Agentforce Studio app is the home of the new Agentforce Builder and a central hub for building, testing, and monitoring your agents.



Learn More: *Access Agentforce Studio and the New Agentforce Builder*

Use the standard Agentforce Service Agent template to get a head start on creating an agent that answers Makana's customer questions and creates and updates cases.

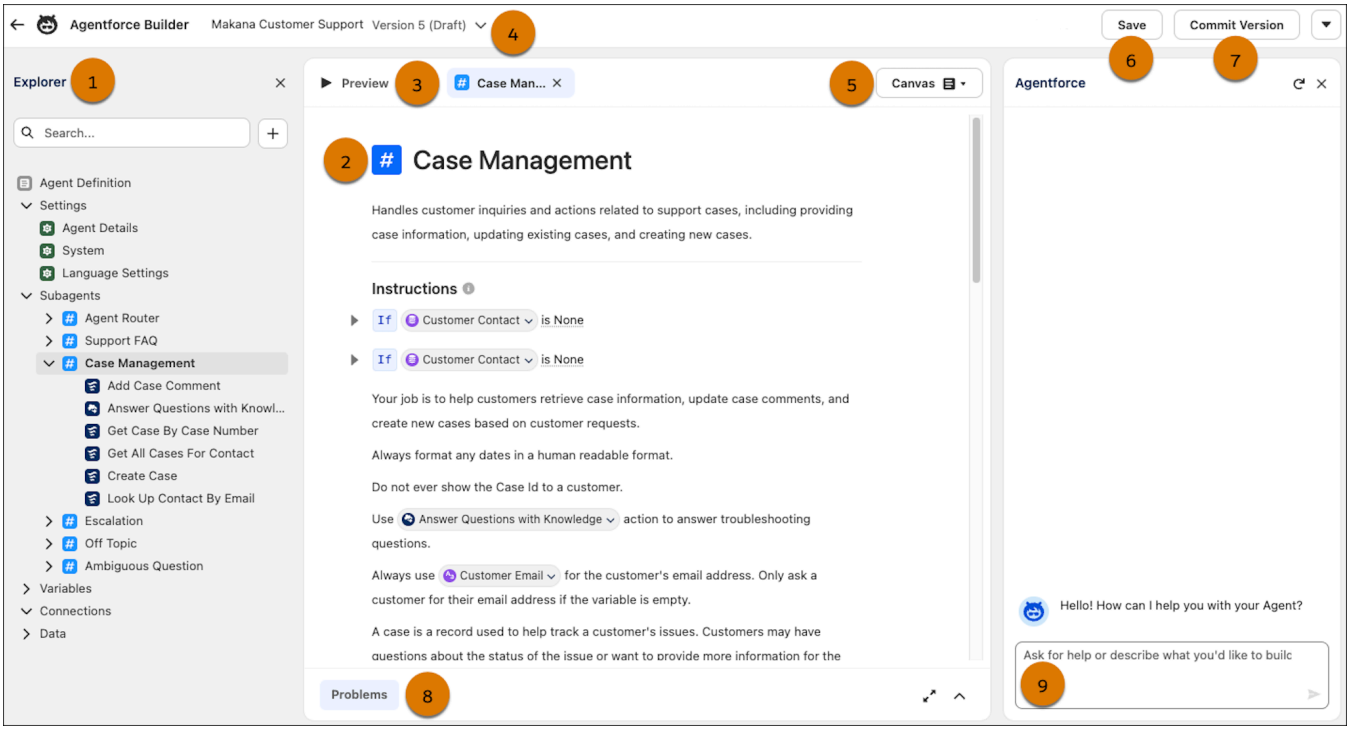
1. From the App Launcher, enter `Agent`, and then select the **Agentforce Studio** app.
2. On the Agents tab, click **New Agent**.
3. Select the **Agentforce Service Agent** template.
   The Agentforce Service Agent template is the standard template for the Agentforce Service Agent type. It includes subagents and actions for handling common customer service use cases, which we can customize for Makana's specific business needs.
4. For the agent name, enter `Makana Customer Support Agent`

5. For the developer name, enter `Makana_Customer_Support_Agent`
6. Select **New User** to create a new user record for the agent.
   The agent user is a Salesforce integration user with the Einstein Agent license type. Because Agentforce Service agents connect to channels that aren't restricted to logged-in users, the agent gets its own user record to securely access data and perform actions that individual end users don't have access to. We'll come back later to this Agent User to assign all of the permissions that the agent needs to do its job.
7. Click **Let's Go**.

Your new agent is opened in Agentforce Builder in Agentforce Studio. Before you make any changes to your agent, take a moment to get familiar with the new builder experience and all it has to offer.

Get to Know Agentforce Builder in Agentforce Studio

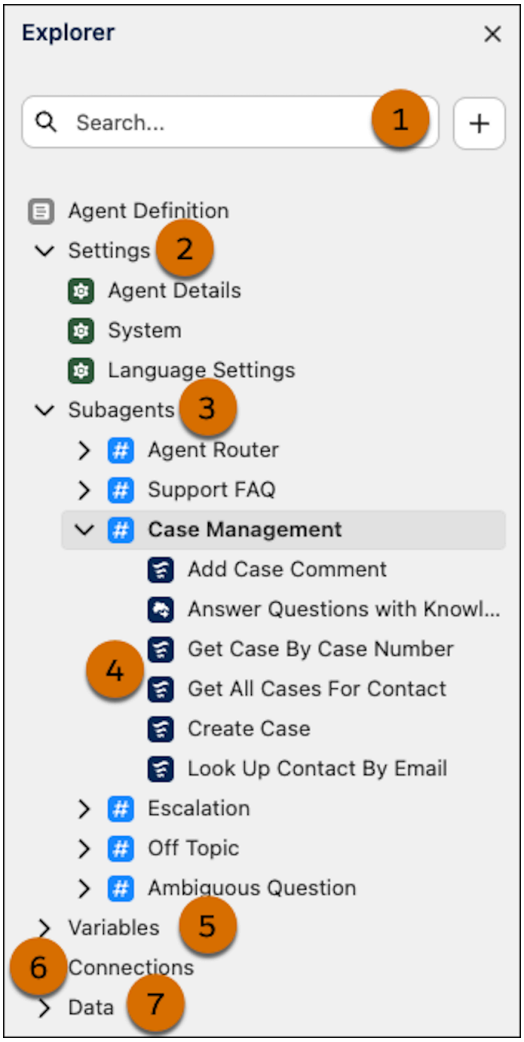
Welcome to Agentforce Builder in Agentforce Studio! Agentforce Builder is where you build, customize, and test predictable, context-aware agents. The Agentforce Builder supports both deterministic logic and natural language prompts, so you can build more complex and reliable agents. For example, you can control when an agent transitions between subagents, chain actions together in sequence, and use variables within conditional logic to control agent behavior based on context.



- Explorer (1): See all the details of your agent, including subagents and their actions, in a single place. You can collapse and expand the Agentforce panel to make room as you build. Use the Agent Definition section to get an overview of your agent and summary of its capabilities.

- Canvas (2): Use the natural language editor to modify your agent. It includes shortcuts for common tasks, including quick actions for adding logic to your agent, and a picker for resources like subagents, actions, and variables.
- Preview (3): Test your agent by inputting utterances. Simulate your agent's performance and view a detailed breakdown of the interaction, including which subagents and actions the agent used to create its response. Get the details in the revamped debugging panel, tracking each moment of the interaction so you can see precisely how the agent spent its time replying.
- Agent Version (4): When you create multiple versions of an agent, verify which version you're working in and easily switch between them.
- Canvas and Script Views (5): Switch between the natural language-based Canvas view, which summarizes Agent Script into easily readable blocks, and the code-based Script view. Your configuration is consistent across both views.
- Save (6): Save the current draft of your agent version.
- Commit Version (7): Save a version of your agent to metadata. You can activate and make the agent version available to your customers. You can't modify an agent version after it's committed. To make further changes, create another version of the agent.
- Console (8): Address errors and warnings that could interfere with your agent's performance, such as syntax errors or unexpected values. When you click an error or warning in the console, the builder opens the element or line with the issue.
- Agentforce Assistance (9): Create, test, troubleshoot, and refine, all with the help of Agentforce. Ask Agentforce questions or direct it to help you build and debug your agent. Tell Agentforce what you want your agent to do, and it converts your request into subagents, actions, and more. You can collapse and expand the Agentforce panel to make room as you build.

Explorer Tour



- Search (1): A way for you to quickly search for the elements of your agent.
- Settings (2): Basic configuration details about your agent.
- Subagents (3): All of your agent's subagents.
- Actions (4): Actions are listed under their respective subagents. Select an action to configure its description, inputs, and outputs.
- Variables (5): Variables associated with your agent. Variables store values that can help you control how your agent performs.
- Connections (6): All of your agent's connections, which define which channels your agent can connect to and how your agent interacts on each channel.
- Data (7): Structured data sources that your agent can reference.

Introducing Agent Script

Agent Script is the language for building agents in Agentforce Builder in Agentforce Studio. Agent Script combines the flexibility of natural language instructions with the reliability of programmatic logic for handling business rules (a powerful combination that we call hybrid reasoning). For example, use expressions in Agent Script to define if-else conditions, transitions between

subagents, and other logic; set, modify, and compare variables; and select subagents or actions together in sequence.

Determinism in Agent Script

Determinism in Agent Script can be particularly helpful when your agent must follow important or sensitive business rules, when instructions are particularly complex, or when instructions must be followed in a sequence. For example, you can use logical expressions to:

- Run one or more actions to complete a prerequisite task or gather information. For example, a subagent that handles merchandise returns can always start by running an action to look up the user's most recent order number.
- Apply instructions for what an agent should do or say based on whether certain conditions are met. For example, check a variable to determine whether a customer is a member of a rewards program and, if so, thank them for their loyalty. Or if not, invite them to join.
- Personalize instructions with contextual information. For example, reference a variable to show customers their current order summary when the conversation starts or if they request it.
- Control routing to specific subagents and actions. For example, add a transition to automatically escalate a VIP customer to a representative before any reasoning takes place.

Building Agents with Agent Script

Agentforce Builder gives you several ways to write Agent Script.

- You can chat with Agentforce and explain what you want your agent to be able to do (for example, "If the order total is over $100, then offer free shipping."). Agentforce converts your request into subagents, actions, instructions, and other expressions. In this guide, Agentforce will do some of the work for you.
- In Canvas view, Agent Script is summarized into easily understandable blocks, which you can expand to view the underlying script. You can edit your agent with the help of quick action shortcuts. Type `/` to add expressions for common patterns (for example, if/else conditionals) and `@` to add resources (for example, actions and variables). In this guide, you'll mostly be working in Canvas view.
- Advanced users can switch to Script view to write and edit script directly, with developer-friendly aids like syntax highlighting, autocompletion, and validation. In this guide, you don't edit your agent's script directly in Script view. But we'll show you side-by-side comparisons of your agent's configuration in Canvas and Script views (your agent's configuration is always consistent across both views) and point you to further resources.

No matter how you build your agent, you're writing Agent Script.



Learn More: *As you work through this guide, you'll be working mostly in Canvas view to add determinism to your agent, so you'll learn more about Agent Script along the way. But we also have lots of resources to help you dive deeper.*

- *Salesforce Help: Building Agents in Canvas View*
- *Salesforce Help: Get Started with Agent Script*
- *Agentforce Developer Guide: Agent Script Common Patterns*

Customize Your Agent's Settings

Specify Makana's agent description, agent-level instructions, and system messages.

Customize Your Agent's Description

When you created your agent from the Agentforce Service Agent template, it inherited the default description from that template. To make sure that you and other agent builders in Makana's org can identify the agent, customize the description to better reflect the agent's capabilities.

1. From the Explorer panel, click **Agent Details**.
   If you don't see Agent Details, expand the Settings section first.
2. For the description, enter `Answers common customer questions about products and creates, looks up, summarizes, and updates customer cases.`
3. Save your changes.

Customize Your Agent's Agent-Level Instructions and System Messages

Now that you've created a basic agent, let's get your agent ready for its first conversation. You'll define agent-level instructions, which provide your agent with guidance on how to interact across all agent conversations. You'll also define system messages, which introduce your agent and handle major errors.

1. From the Explorer panel, click **System**.
   If you don't see System, expand the Settings section first.
2. For agent-level instructions, enter `You're a customer service agent for a global medical device company. Always respond to customer questions and requests professionally. When a customer is frustrated (for example, they can't find an answer to their question), empathize with their frustration and promptly offer to escalate to a service rep.`
   You can think of agent-level instructions as system instructions that apply to every agent interaction. Makana uses this field to define the agent's role and tone to match their expectations for good customer service.



Note: *This won't be necessary for Makana's use case, but you can override agent-level instructions at the subagent level. For example, to provide a different voice and tone. See Common Patterns: Avoid Conflicting Instructions with Instruction Overrides.*

3. For the welcome message, enter `Hi, I'm Makana's customer support AI agent. You can ask me questions about our devices, our shipping policies, and more. I may make mistakes, so review my messages for accuracy. What can I help you with?`
   This message is sent at the start of every agent conversation. Makana makes sure to introduce the agent as an AI agent and not a human, which helps establish customer transparency and trust (and is a legal requirement in some regions). Makana also knows

that their agent, like all other generative AI applications, can make mistakes. So it encourages its users to carefully review agent responses for accuracy.

4. For the error message, enter `Something went wrong. Try again.` Makana keeps this error message generic because it applies only in the case of non-recoverable system errors, such as a lost connection or LLM outage. The agent attempts to handle more minor, recoverable errors naturally as part of the agent conversation.
5. Save your changes.



Learn More: *Makana's agent is English only, so we don't have to adjust any language settings. But if you want to change the language or add more languages to an Agentforce Service agent, this is a good time in the process to get started. Learn more in Salesforce Help.*

- *Update Language Settings*
- *Use Additional Languages in Enhanced Chat Conversations*
- *Agentforce Language Support*
- *Agentforce Considerations*

Ground the Agent's Responses in Makana's Information

Next, to ground your agent's responses in Makana's product information, and customer service processes and policies, we'll create a data library.

Agentforce Data Library connects AI agents to trusted data sources and turns unstructured or semi-structured data into useful, searchable information. When you create a data library, it automates several configuration steps across Data Cloud and Prompt Builder to make it easy to connect your agents to your data. Learn more about data libraries.

Agents use the Answer Questions with Knowledge action to access the data library. When a customer asks your agent a question, the Answer Questions with Knowledge agent action is used to answer the query based on the sources available in the agent's assigned data library. The Answer Questions with Knowledge action is included in the General FAQ and Case Management subagents you'll use with your agent.



Learn More: *For Makana's use case, Agentforce data libraries are a sufficient grounding solution. But for more complicated datasets with unstructured data and long free-text fields, we recommend implementing retrieval augmented generation (RAG).*

Create a Data Library

To add data to a data library, you can use Salesforce Knowledge, upload files, use web search, or use a custom retriever. Makana doesn't use Salesforce Knowledge, so for our use case, we'll upload a PDF that contains Makana's product information about their CGM3000 continuous glucose

monitoring device and customer service processes and policies. If you haven't already, download the PDF.



Note: *The information that your agent returns is only as good as your data. In this case, Makana has already reviewed their knowledge to ensure that their content is specific, organized, detailed, and accurate. Before you add your own source content to a data library, prepare your content for AI and human consumption.*

1. From Setup, in the Quick Find box, enter `Data Library`, and then select **Agentforce Data Library**.
2. Click **New Library**.
3. For the library name, enter `Makana Products and Policies`. The API name is automatically populated.
   Makana is uploading information that's specific to their CGM3000 device, but we're leaving the data library name more general so they can add more files to it later.
4. For the description, enter `Sources for Makana Customer Service agent's product information and customer service processes and policies.`
5. Save your changes.
6. In the Data Type field, select **Files**.
7. Upload the makana_cgm3000_product_info.pdf file from wherever you saved it on your computer. When your file is finished uploading, click **Done**.

   After you've uploaded your files, a search index and a retriever are created automatically and available to view or edit in Data Cloud. You can monitor your data library's progress from Setup. When the status updates to "Your data library is ready to use!", the library is ready to use with your agent. You don't have to wait for your data library to be ready to use to move to the next step.

Assign a Data Library to the Agent

Now that you've created your data library, add it to your agent.

1. You should still have your agent open in Agentforce Builder in a separate tab.
   If you don't have your agent open, from the Agents tab in the Agentforce Studio app, click the name of your agent.
2. From the Explorer panel, click **Data**, and then click **Data Library**.
3. Select the **Makana Products and Policies** library. If you don't see your library, refresh the page.
4. Leave the Show Sources setting disabled. When this setting is enabled, your agent shows the sources for every answer it generates with your data library. For the Makana use case, we'll leave that setting disabled because we're citing a single internal PDF for all of the agent responses generated from your data library.
5. Save your changes.

Add the URL for Makana's Website to Your Trusted URLs

To make sure Makana's agent can share URLs referenced in its data library, add the domain for Makana's website to your Salesforce trusted URLs.

Agentforce enforces your trusted URL allowlist, so you have more control over the links that your agents call, generate, and share. Because we're using an internal PDF and not publicly available knowledge articles for our use case, we're not implementing traditional citations. However, when an uploaded file contains URLs (for example, when a file references a specific page on your company's website to submit a warranty claim), it can be helpful for the agent to include the website in relevant responses. If you don't add the domain for a referenced website to your trusted URLs, it's blocked from agent responses.



Note: *Just as Makana is a fictional company, the Makana website referenced in your data library is also fictional. Any URL that your agent shares won't be functional. We're including the steps for adding a trusted URL because it's a requirement for* ***any*** *URL you plan to use with Agentforce, and we want you to plan for this requirement when you're building your next agent.*

1. From Setup, in the Quick Find box, enter `Trusted URLs`, and then select **Trusted URLs**.
2. Click **New Trusted URL**.
3. For the API name, enter `Makana`
4. For the URL, enter `https://makanamedicalsupplyinc.com`
5. Save your changes.



Learn More: *For each trusted URL in Setup, you can specify Content Security Policy (CSP) directives and Permissions-Policy directives. Learn more about how to limit Interactions with external URLs.*

Configure Your Agent with Subagents and Actions

The subagents and actions that you get out of the box with the default Agentforce Service agent template are a great start for setting up a customer service agent. But standard subagents and actions are meant to be customized to meet the needs of your business. Think of them as templates of best practices for common use cases.

Before you make changes to a standard agent subagent or action, get to know the subagent and actions and review all of their fields.

Subagent Fields
All subagents contain a name, classification description, scope, and instructions.

Field	What It Is	How an Agent Uses It

Name and Description	The subagent name is the API name of the subagent. It's usually 2-3 words that describe what the subagent does in plain language. The description is 1-3 sentences that describe what a subagent does and the types of user requests that should be classified into this subagent.	These fields are used along with a special subagent, the Agent Router, to determine when to use a subagent in a conversation. The agent compares the names and descriptions of all of the subagents assigned to the agent to the user's question or request and recent conversation history. Based on that context and the instructions in the Agent Router, the agent selects the best match.
Instructions	The guidelines that tell your agent how to handle conversations in the context of the subagent, your business case, and the conversation in general. Instructions contain a combination of programmatic logic (written in Agent Script) and natural language prompts.	Instructions are used to create the prompt that's sent to the LLM. After the agent selects a subagent, it begins resolving the reasoning instructions in order, executing any logic and running any actions along the way. When the agent reaches the end of the instructions, the agent sends the resolved prompt to the LLM to start the reasoning process.
Actions Available for Reasoning (Reasoning Actions)	The actions an agent is allowed to take during reasoning with the LLM. They include references to traditional agent actions, such as Answer Questions with Knowledge. But they can also include utilities, which include transitions to another subagent, setting variables, or escalations to a representative.	Reasoning actions are sent to the LLM along with the prompt created from the reasoning instructions. The LLM chooses which actions and utilities to run based on the user's message, the recent conversation history, the resolved reasoning instructions, and any logic defined in the reasoning actions.



Learn More: *Subagents are fundamental to defining your agent's behavior. Want to go deeper? Learn more in Salesforce Help.*

- *Agent Subagents*
- *Subagent Classification and Routing*
- *How Subagent Instructions are Resolved to Build a Prompt*
- *Video: How an Agentforce Agent Resolves Instructions to Build a Prompt*

Agent Actions

Each action in a subagent is a task that your agent can perform when the conversation is routed to that subagent.

An agent action includes:

- A natural language name and instructions that tell the agent how and when to use the action, how to retrieve required inputs, and how to format and use outputs.
- The Salesforce functionality that the agent action calls to get information or perform a task, called a reference action. For example, an agent action can call an API to retrieve data, a flow to update a record, a prompt template to generate a response, an Apex class to run custom business logic, or a predictive model to make a recommendation.

Remove Unnecessary Subagents from Your Agent

When you create an agent from a template, all of the standard subagents and actions included in the template are added to your agent. The Agentforce Service Agent includes subagents for a wide range of customer service use cases, but Makana's agent is focused on handling frequently asked questions and case management. So you can remove the Account Management, Reservation Management, Delivery Issues, Order Inquiries, and Service Customer Verification subagents from the agent.

1. From the Explorer panel, hover over the Account Management subagent.
   If you don't see the list of subagents assigned to your agent, expand the Subagents section first.
2. Click , and then click **Delete**.
3. When prompted to confirm that you want to remove the subagent from your agent, click **Yes, Delete**.
4. Repeat this process for the Reservation Management, Delivery Issues, Order Inquiries, and Service Customer Verification subagents.
5. Save your changes.



Note: *You can always add or remove subagents later. To explore all the standard subagents and actions you can add to your agent, from the Explorer panel, hover over a subagent and click . Then select* ***Add from Asset Library****. Or you can explore the asset library from the Agentforce Assets page in Setup. In the Quick Find box, enter and select* ***Agentforce Assets****.*

Now your agent contains these subagents: Agent Router, General FAQ, Case Management, Escalation, Off Topic, and Ambiguous Question. Before making any changes to these subagents, get to know them and review all of their actions and instructions.

Subagent	What It Does
Agent Router	The Agent Router is a system subagent that's added to all agents by

	default and manages your agent's conversational flow. By default, the agent uses this subagent for subagent classification, which is the process of selecting the most relevant subagent based on what the user wants to do and the jobs that the agent can do. When you add a subagent to your agent, a transition to the subagent is added to the Agent Router's reasoning actions, which makes the subagent available for the agent to select in conversation. Because Makana's agent only handles a couple of use cases with clearly defined subagents, subagent classification works well for their use case. We only have to make minimal changes to the Agent Router. But you can edit it just like any other subagent, so you can customize its instructions and reasoning actions to set initial variables and add logic to control your agent's subagent classification and routing. Learn more in Salesforce Help.
General FAQ	The General FAQ subagent contains the Answer Questions with Knowledge action and instructions for how to use it to answer common customer questions.
Case Management	The Case Management subagent contains actions and instructions for handling support cases, including updating cases and creating new cases.
Escalation	The Escalation subagent is a system subagent that's added to all agents by default. When an agent encounters complex or sensitive conversations that it can't resolve, it uses the Escalation subagent to transfer the conversation to a service rep or any other destination you define.
Off Topic	The Off Topic subagent is a system subagent that's added to all agents by default. When a user's conversation strays outside of the use cases the agent is configured to handle, the agent uses this subagent to redirect the conversation to relevant subagents.
Ambiguous Question	The Ambiguous Question subagent is a system subagent that's added to all agents by default. When a user's question or request is too ambiguous for the agent to handle, the agent uses this subagent to prompt the user to provide clearer, more focused requests for better assistance.

Now that you've gotten to know the agent's subagents, you're ready to customize them to meet Makana's needs.

Customize the General FAQ Subagent

The General FAQ subagent contains the Answer Questions with Knowledge action. The out-of-the-box functionality of the subagent and action are pretty close to meeting Makana's needs, so we only need to make a few adjustments to the subagent.



Note: *The legacy builder was structured around forms and fields. Agentforce Builder in Agentforce Studio features flatter navigation and a text editor interface. As you edit subagents, you'll be typing and editing directly on the canvas. Learn more about building and editing agents in Canvas view.*

We're not pulling out any Agent Script tricks for the FAQ subagent, to give you a chance to make some simple changes first while you get to know the new Agentforce Builder interface.

1. From the Explorer panel, click the General FAQ subagent to open it on the canvas.
2. First, rename the subagent. On the canvas, highlight and delete General FAQ, and then enter `Support FAQ`.
3. Next, edit the description. This is the block of text directly below the subagent name. Keep the existing description and append `For example: "What is your return policy?" , "Do you offer an extended warranty on the continuous glucose monitor?", and "How can I get a refund?"`
   Adding examples of the kinds of questions we expect Makana customers to ask gives the agent more clues about when to use the subagent in a conversation.
4. At the bottom of the Instructions section, add a new line of instructions and enter `Never give customers medical advice. Always recommend that they contact 911 if they're having a medical emergency or their physician if they need help understanding something related to their health.`
5. Save your changes.

Customize the Case Management Subagent

The Case Management subagent includes standard actions that help it handle customer inquiries and support cases. Like all standard subagents, it gives us a great start for a general use case, but it requires customization to meet all of Makana's specific requirements.

- The Case Management subagent is designed to require customer verification with the standard Customer Verification subagent. Verifying that customers should have the ability to interact with actions and cases from the agent is a best practice, but it takes careful configuration. While we're getting the agent up and running, we'll replace these actions with similar actions that don't require verification and remove a filter from the subagent. This makes it easier to build and test the agent.
- The Case Management subagent instructs the agent to always ask for the customer's email address and get their contact record. While this is a necessary step, there's more than one way to handle it. Rather than rely on the agent to prompt the customer for this information, Makana's decided to create a pre-chat form to prompt customers for this and other information before connecting to an agent. You'll build this form in the deploy stage when you connect your agent to Messaging channels. For now, we'll customize the subagent to work seamlessly with the pre-chat form.
  - You'll give the agent access to the Customer Email variable on the Messaging Session object, so the customer has access to the customer's email address.

- You'll create a custom Look Up Contact By Email action to use the customer's email to search for the customer's contact record.
- You'll add a mix of deterministic and natural language instructions to tell the agent when to run the Look Up Contact By Email action and ensure the agent doesn't ask for information that the customer has already provided.

Replace Versions of Standard Actions that Require User Verification

While we're getting the agent up and running, we'll replace actions that require user verification with similar actions that don't require verification. This makes it easier to build and test the agent.

If you open the flows associated with the Create Case with Enhanced Data, Get Case By Verified Case Number, and Get Cases For Verified Contact actions, you'll notice that they have an extra step built in to look up the customer's contact ID and check it against the verifiedCustomerId variable. These actions are designed to require customer verification with either the Customer Verification or the Service Customer Verification subagent, and the agent can't complete the actions if they can't pass this check. We removed the Service Customer Verification subagent from the agent, so we need to replace the actions that require user verification.

Before making any changes, review the actions in the agent that require user verification and the actions that you'll replace them with.

Standard Agent Action that Requires Verification	Standard Agent Action that Doesn't Require Verification
Create Case With Enhanced Data A flow-based action that creates a case for a verified customer, including information gathered from the customer, a summary of progress made by the agent, a link to the conversation, and any attachments.	Create Case A flow-based action that creates a case for a customer based on information gathered from the customer.
Get Case By Verified Case Number A flow-based action that looks up a case based on a verified contact ID and case number.	Get Case by Case Number A flow-based action that looks up a case based on a given contact ID and case number.
Get Cases For Verified Contact A flow-based action that looks up a list of cases based on a verified contact ID.	Get All Cases for Contact A flow-based action that looks up a list of cases based on a given contact ID.

1. From the Subagents section of the Explorer panel, expand the **Case Management** subagent to see all of the actions assigned to it.
2. Remove the following actions.
   - Create Case with Enhanced Data
   - Get Cases for Verified Contact

- Get Case by Verified Case Number

To delete an action, in the Explorer panel, hover over the action. Click , and then click **Delete**.



Note: *To see the full name of an action, hover over the action in the Explorer panel.*

3. From the Subagents section of the Explorer panel, hover over the Case Management subagent and click . Then select **Add from Asset Library**.
4. Search for and select these actions, and then click **Add to Agent**.
   - Create Case
   - Get All Cases for Contact
   - Get Case by Case Number
5. Save your changes.



Note: *Each time you save your agent, Agentforce Builder validates the agent's script and checks for any errors. If you get an error after you've removed an action from a subagent, the error often means that the action definition was removed from your subagent, but it's still referenced in your subagent's Actions Available for Reasoning. In Canvas view, open the subagent that you removed the action from and delete any lingering references. You can click into the canvas and backspace/delete them, just like in a text editor. Then save your changes again to verify that you resolved all errors.*

To review your agent's errors, you can use the console or go directly to your Agent Definition.

- *Console: Open the console at the bottom of the canvas. To quickly address a problem, click an error or warning in the console. The builder opens the element or line with the issue in the view you're currently in. You can commit an agent that has warnings, but not one with errors.*
- *Agent Definition: in the Explorer panel, go to the Agent Definition. If it doesn't open in Script view, use the view toggle to switch from Canvas view to Script view. On the right, click on any red lines to jump to errors, and then hover over any highlighted text to see error messages and quick fixes, if available.*

At this point, you've removed the actions that require user verification. However, if you look at the Case Management subagent instructions, you'll notice that there's still a resource reference to the Create Case with Enhanced Data action.

> Once the user confirms that they want to create a case, use CreateCaseEnhancedData with the subject and description for the case immediately and inform the user that the case was created.

Update the instruction to replace the outdated resource reference and add instructions to guide behavior if case creation fails.

1. On the canvas, find and delete the instruction “Once the user confirms they want to create a case, use CreateCaseEnhancedData with the subject and description for the case immediately and inform the user that the case was created.”
2. On the same line, enter `Once the user confirms that they want to create a case, use {!@actions.CreateCase} with the subject and description for the case immediately and inform the user that the case was created. If case creation fails for any reason, escalate to a service rep.`



Note: *When the agent begins resolving instructions to prepare for reasoning, it replaces references to resources (like actions and variables) with the values themselves.*

We’ve provided the Agent Script syntax for referencing an action to make the instruction easy to copy and paste. But you can reference a variable or other resource without memorizing Script. In Canvas view, anywhere in the instructions, type @ to open the resource menu. Then search for and select any resource you want to reference.

If you already have a reference to a resource and want to change the reference, click the dropdown arrow on the resource ( ) to open the resource menu.

3. Save your changes.



Learn More: *Learn about how an agent resolves resources in subagent instructions to build a prompt.*

- *How Subagent Instructions are Resolved to Build a Prompt*
- *Common Patterns: Resource References*

Remove the Verified User Filter from the Case Management Subagent Transition

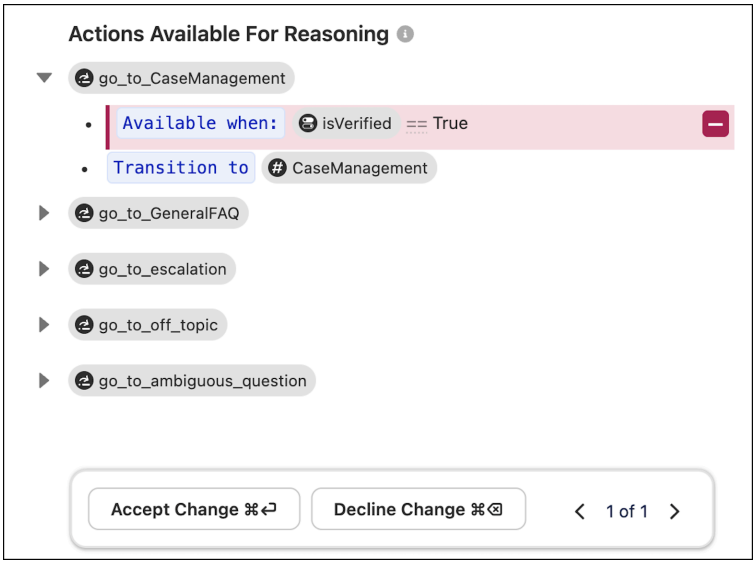
Filters limit access to subagents and actions based on conditions and variables. Because the Agentforce Service Agent template is designed to work with the Service Customer Verification subagent out of the box, the transition to the Case Management subagent in the Agent Router includes a Verified User filter.



When this filter is applied, the agent can only use the Case Management subagent in a conversation if the user is verified. We removed the Service Customer Verification subagent from the agent, so now we need to remove the filter from the Case Management subagent.

So far, we've made our changes manually in Canvas view, but that's not the only way to get things done. Agentforce Builder in Agentforce Studio has AI assistance built in, so let's ask Agentforce for help.

1. The Agentforce assistant panel is a chat window docked on the right side of Agentforce Builder. If you don't see it, click (located to the right of the Canvas View / Script View toggle) to expand it.
2. In the message field, enter `Remove the filter on the Case Management subagent`. Then send the message to Agentforce.
3. Review the change that Agentforce suggests. It should look like this.



4. Accept the change, and then save..



Note: *Agentforce can help with all aspects of agent building, from answering questions to creating subagents and actions. Here are some ways that Agentforce can help.*

- *Generate Agent Script. Use natural language to explain what you want your agent to be able to do, and Agentforce can generate Agent Script for you based on your request. You can review and accept or decline its changes.*
- *Evaluate your instructions to improve and refine them. For example, you can ask Agentforce to make your instructions more deterministic.*
- *Suggest fixes. When you're previewing your agent, you can ask Agentforce to explain your agent's reasoning, identify what went wrong, and even ask it to make modifications for you.*

Create, test, troubleshoot, and refine, all with the help of Agentforce.

Create a Look Up Contact by Email Custom Agent Action

Makana's pre-chat form collects the customer's email address at the beginning of the conversation. Build a custom agent action that checks if the email address belongs to an existing contact.

Create a Field to Store the User's Email Address on the Messaging Session Object

Create a custom field on the Messaging Session object that stores the customer email address provided in the pre-chat form.

1. From Setup, go to Object Manager.
2. Search for and select the **Messaging Session** object.
3. Click **Fields & Relationships**.
4. Click **New** to create a new custom field.
5. For the data type, select **Email**, and then click **Next**.
6. For the field label, enter `Customer Email`, and then click **Next**.
7. Leave the Field-Level Security settings unchanged, and then click **Next**.
8. Save your changes.

Create the Look Up Contact by Email Flow

Create a flow that looks up a contact using the email address that the customer enters in the pre-chat form. Next, you'll create an agent action that's based on this flow.



Learn More: *If you're new to building flows, learn about elements, conditions, and more in Salesforce Help.*

1. From Setup, in the Quick Find box, enter `Flows`, and then select **Flows**.
2. Click **New Flow**.
3. To create an autolaunched flow from scratch, select the **Autolaunched** category, and then select **Autolaunched Flow (No Trigger)**.
4. Add a Get Records Element.
    a. For the label, enter `Get Contact from Email`.
    b. For the object, select **Contact**.
5. Filter contact records by email address.
    a. For the condition requirements, leave **All Conditions Are Met (And)** selected.
    b. For the field, select **Email**.
    c. For the operator, select **Equals**.
    d. For the value, add a new resource.

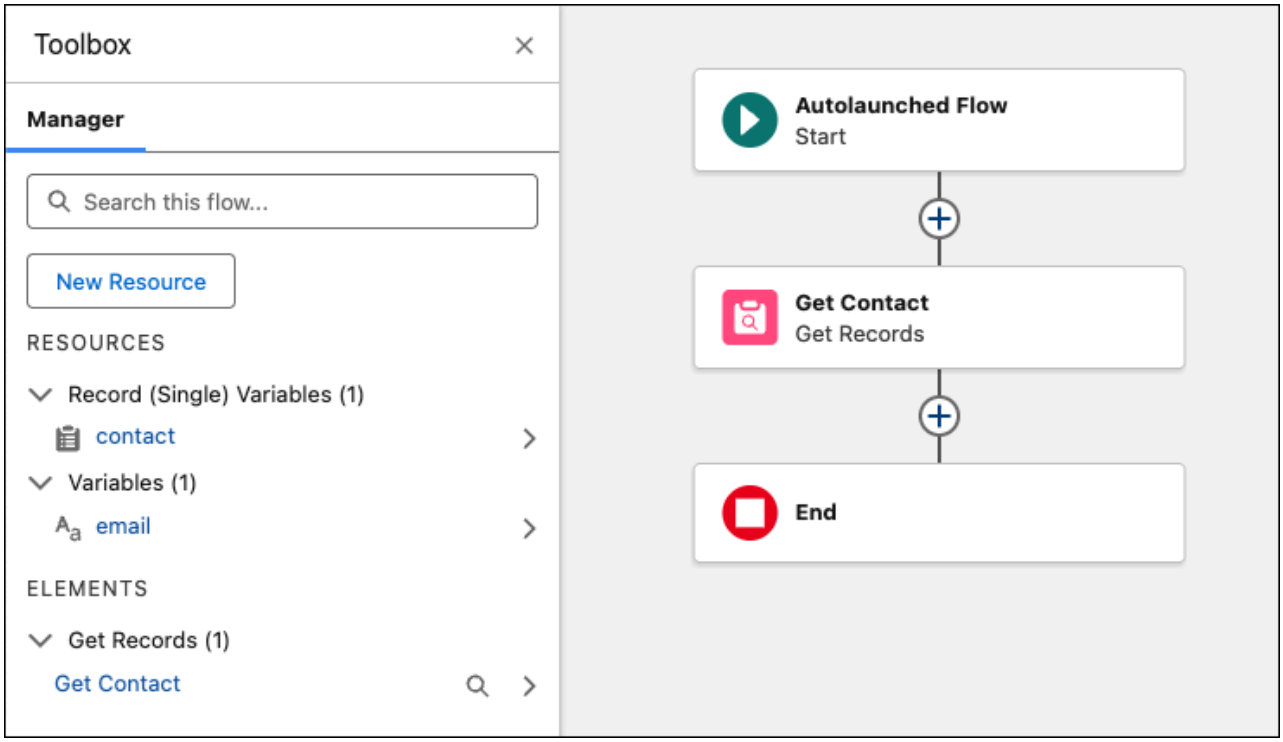
Field	Value
Resource Type	Variable
API Name (case-sensitive)	email

Description	The email address associated with the customer record.
Data Type	Text
Availability Outside the Flow	Available for input

6. To ensure that you're passing in the original contact record associated with the email address, sort the contact records in ascending order by CreatedDate.
7. Create a variable to store the record variable.
    a. Under How to Store Record Data, select **Choose fields and assign variables (advanced)**.
    b. Under Select Variable to Store Contact, in the Record field, create a new resource.

Field	Value
Resource Type	Variable
API Name (case-sensitive)	contact
Description	The contact record associated with this customer.
Data Type	Record
Object	Contact
Availability Outside the Flow	Available for output

    c. Under Select Contact Fields to Store in Variable, in addition to Id, add fields for **Email**, **FirstName**, and **LastName**. All of this information is passed back to your agent to use in the agent conversation.
8. Save the flow.
    i. For the flow label, enter `Look Up Contact by Email`.
    ii. For the description, enter `Looks up a customer by their email address and returns their contact record`.
9. Activate the flow.



Note: *When you're creating your own flows, you can debug your flow as the agent user to ensure that your flow works as expected and that your agent's user record has all of the necessary permissions to run it. We haven't adjusted your agent user's permissions yet, so we're skipping this step for now. (If you try it now, your flow won't pass debugging.) But if you want to jump ahead, see Add Permissions to the Agent User.*

1. *Give yourself the required permission to debug your flow as another user.*
   a. *From Setup, in the Quick Find box, enter Process Automation, and then select* ***Process Automation Settings****.*
   b. *Select* ***Let admins debug flows as other users****.*
   c. *Save your changes.*
2. *In Flow Builder, click* ***Debug****.*
   a. *Select* ***Run automation as another user****, and then select your EinsteinServiceAgent User.*
   b. *Under Input Variables, in the email field, enter the email address of the contact that you want to retrieve.*
   For a simple debug test for the purposes of this guide, you can grab an email address from any contact in your Sales or Service app.
   c. *Click* ***Run****.*

Add the Customer Email Custom Variable to the Agent

Agent context variables are mapped to object fields, so they let your agent use the values of record fields in agent conversations and testing. Makana's agent already includes context variables that are mapped from Messaging Session object fields. We've added the custom Customer Email field to the Messaging Session object, so now you need to give the agent access to the associated context variable.

1. Back in Agentforce Builder, from the Explorer panel, expand the Variables section and open the Variables page.
2. Click **New**, and then select **Add Messaging Session Variable**.
3. Search for and select the **Customer Email** variable, and then add it to your agent.
4. Save your changes.



Learn More: *Learn more about agent context variables in Salesforce Help.*

- *Agent Variables*
- *Use Context Variables in Messaging Conversations*



Note: *While we're on the subject of variables, you may notice that as you're building your agent, the console warns you that your agent includes variables that you don't reference in your agent's configuration.*

- *Some of these variables reference fields from the Messaging Session object and are used by the agent when it's deployed to a channel: EndUserId, RoutableId, ContactId, and EndUserLanguage (plus the Customer Email variable you've just added). Don't remove these variables from your agent.*
- *Some of these variables are left over from when we created the agent from the default Agentforce Service Agent template, and Makana's agent configuration doesn't require them: authenticationKey, customerId, customerType, isVerified, and VerifiedCustomerId.*

  You can save and commit an agent with warnings, so you don't have to remove these variables from your agent–it won't affect your agent's performance. But it's a best practice to remove what you don't use, and removing them makes it cleaner to navigate later configuration steps and manual testing.

To delete a variable, from the Explorer panel, expand the Variables section and open the Variables page. From the dropdown menu next to the variable you want to delete, select ***Delete Variable****, and then click* ***Delete****.*

Create a Custom Agent Action Based on the Look Up Contact by Email Flow

Now that you've created your flow, create an agent action so your agent can run the flow in an agent conversation.



Note: *These steps create the Look Up Contact by Email action in your Makana Customer Support agent. Other agents can't use this action. To create it as an action that other agents can use, create the action from the asset library. From Setup, in the Quick Find box, enter* `Agentforce Assets`*, and then select* ***Agentforce Assets****. Then click* ***New****. Once your action is available in the asset library, you can add it to your agent in Agentforce Builder.*

1. Back in Agentforce Builder, in the Explorer panel, hover over the Case Management subagent and click . Then select **New action**.
2. For the action name, enter `Look Up Contact by Email`
3. For the description, enter `Looks up a customer by their email address and returns their contact record.`
4. Click **Create and Open**. The Look Up Contact by Email action has been added to the Case Management subagent. It's automatically added to your subagent's reasoning actions (Actions Available for Reasoning), so it's available for Makana's agent to select during reasoning with an LLM. But we still have a few more steps before it's up and running.
5. For the reference action type, select **Flow**.
6. For the reference action, select your **Look Up Contact by Email** flow.
7. For the email input, the description autopopulates from the flow, so no change is needed. Select **Require Input to execute action**.
8. For the contact output, the description autopopulates from the flow. Select **Show in conversation**. When this setting is enabled, the output can be included in the agent's response.
9. Under Additional Settings, click **Show loading text for this action**, and then enter `Looking up info...`
10. Save your changes.

Replace Conflicting Natural Language Instructions

Now that you've created the custom Look Up Contact By Email action, you can use instructions to tell the agent how to use it.

We've added the custom Customer Email variable to the agent, so the agent has access to the customer's email address from the prechat form. However, the out-of-the-box instructions tell the agent to ask for the customer's email address. Modify the instructions to make sure the agent doesn't ask for information that the customer has already provided.

1. In the Explorer panel, click the **Case Management** subagent to open it.
2. On the canvas, find and delete the instruction "If the customer is not known, always ask for their email address and get their Contact record before running any other actions."

3. On the same line, enter `Always use {!@variables.Customer_Email} for the customer's email address. Only ask a customer for their email address if the variable is empty.`
   Now the instruction references the Customer Email variable directly via Agent Script syntax, instead of just natural language.



Note: *We've provided the Agent Script syntax for referencing a variable to make the instruction easy to copy and paste. But you can reference a variable or other resource without memorizing Script. In Canvas view, anywhere in the instructions, type @ to open the resource menu. Then search for and select any actions or variables you want to reference.*

4. Save your changes.

Run the Look Up Contact By Email Action to Fetch the Customer's Contact Record Before Reasoning

Many of the actions in the Case Management subagent require the customer's contact, so the Look Up Contact by Email action is an important step in Makana's business workflow. Instead of letting the agent decide whether or not to run the action, you can make sure the action runs every time the Case Management subagent is routed to and the agent doesn't have the customer's contact. You'll add some logical expressions to the reasoning instructions to run the action deterministically and to fetch the customer's contact record before any reasoning begins.

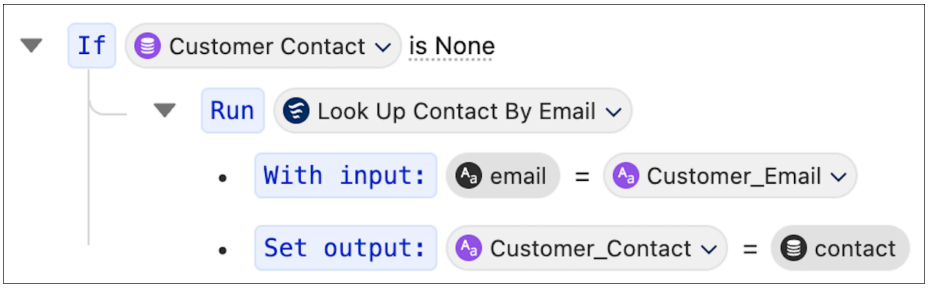
First, you'll create a conditional expression to check whether the agent already has the customer's contact record. This ensures that you're not unnecessarily running an agent action. Then you'll use variables to give the agent the information it needs to run the action.

1. At the top of the instructions, create a new line.
   Instructions are resolved in order, so it's important to place any prerequisite logic early in your instructions.
2. Type / to access inline actions, and then select **If/Else (Conditional)**.
   Inline actions are shortcuts for adding deterministic logic to your agent.
3. Click **Select variable**, and then select **Create a variable**.
4. Enter the following fields. Then click **Create**.

Field	Value
Name	Customer Contact
API Name	Customer_Contact
Data Type	Object
Description	The contact record associated with this customer.

5. Click == to access available operators, and then select **is None**.
   The agent will only complete the nested instructions if the Customer Contact variable is empty. Otherwise, the agent will continue on to the next instruction.
6. In the next nested line, type /, and then select **Run Action**.
7. Click **Select action**, and select **Look Up Contact By Email**.
8. When you run an action in instructions, you must manually set variables for the inputs and outputs because the action is run before any reasoning takes place.
   a. For the input, click **Select variable**, and then select **Customer Email**. The input is mapped to the value of the Customer Email variable.
   b. For the output, click **Select variable**, and then select **Customer Contact**. The output is stored in the Customer Contact variable.
9. Save your changes.

Here's what the expression looks like in Canvas view.



And here's the same expression in Script view, in the raw Agent Script.

```
if @variables.Customer_Contact is None:
    run @actions.Look_Up_Contact_By_Email
        with email = @variables.Customer_Email
        set @variables.Customer_Contact = @outputs.contact
```

Remove the Look Up Contact By Email Action from Actions Available for Reasoning

At this point, you've added the custom Look Up Contact by Email action to your agent's subagent instructions so that it runs every time the subagent is selected and the conditions are met. But when you originally added the action to your subagent, it was automatically added to the subagent's reasoning actions, so your agent can select it during reasoning with an LLM. For Makana's use case, it doesn't make sense to give the agent both options in the Case Management subagent, so remove it from the Actions Available for Reasoning section.

1. In Actions Available for Reasoning, place your cursor immediately after the Look Up Contact By Email action.
2. To remove the action, backspace or delete twice.
3. Save your changes.

Map the Customer Contact Variable to Action Inputs

When you added logic to run the Look Up Contact By Email action in reasoning instructions, you created the Customer Contact variable to store the output of the action. This step was necessary in

order to run the action before reasoning begins. But anytime you store information in a variable, you can use it elsewhere in your agent.



Learn More: *Get to know agent variables and all of the Agent Script functionality that they unlock.*

- *Salesforce Help: Variables*
- *Common Patterns: Variables*

If you expand the actions in the Actions Available for Reasoning section of the Case Management subagent, you'll notice that Create Case, Get All Cases For Contact, and Get Case By Case Number all require a contact as input. Since Makana stores the customer's contact in a variable, you can map it to the input of these actions to ensure the agent only uses the value of the variable to fill the input.

1. In Actions Available for Reasoning, expand the Create Case action so you can see all of the inputs and outputs.
2. On the "With input: Contact Record" line, click **Agent Populated**, and then click **Select variable**.
3. Search for and select the Customer Contact variable.
4. Repeat these steps for the Get All Cases for Contact and Get Case By Case Number actions.
5. Save your changes.

Create a Multi-Subagent Workflow to Escalate from the Case Management Subagent to a Service Rep When a Customer Isn't Associated with a Contact Record

In the last step, you mapped the Customer Contact variable to the inputs of the Create Case, Get All Cases for Contact, and Get Case By Case Number. As you saw, most of the actions in the Case Management subagent require a customer to be associated with a contact record. Makana needs to implement a conversational flow to handle customers who want to create a case but aren't associated with a contact record. You'll use a conditional statement to set a variable to represent the customer's state and automatically transition the customer to a service rep, before reasoning begins and any other instructions are resolved. Then you'll use another conditional statement to customize the escalation instructions specifically for a customer in this situation.



Learn More: *Get to know agent variables and all of the Agent Script functionality that they unlock.*

- *Variables (Agentforce Help)*
- *Common Patterns: Variables*

Conditionally Set a Variable and Transition to Another Subagent in the Case Management Subagent

When a customer is routed to the Case Management subagent, you configured the agent to run the Look Up Contact By Email action when the Customer Contact variable is empty. Now you'll add conditional logic to set a variable when the customer wants to create a case but has no matching

contact record. In the same conditional statement, you'll add a transition to automatically send the customer to the Escalation subagent, so a service rep can assist.



Learn More: *When you add a conditional transition in your instructions and the conditions are met, the agent transitions to the new subagent immediately. No following instructions are resolved or sent to the LLM.* Learn more about conditional transitions.

- Common Patterns: Conditional Transitions
- Common Patterns: Required Subagent Workflow

1. If you don't already have the Case Management subagent open, in the Explorer panel, click the name of the subagent.
2. Create a new line of instructions after the first conditional you created (If Customer Contact is None).
   a. If the conditional is expanded to show you the nested instructions (Run Look Up Contact By Email), click ▼ to collapse the expression.
   b. Place your cursor at the end of the line of instructions and hit Enter to create a new line.
3. Type / to access inline actions, and then select **If/Else (Conditional)**.
4. Click **Select variable**, and then search for and select **Customer Contact**.
5. Click == to access available operators, and then select **is None**.



Note: *In reasoning instructions, the order of deterministic expressions and other instructions is important because the agent resolves instructions in order. In this case, Makana is taking advantage of this workflow.*

In the first conditional, we check the Customer Contact variable to make sure it's not already filled, so we don't run the Look Up Contact By Email action when the agent already has the information it needs. If the variable is empty, the action runs and, if the action finds a matching contact, fills the variable. If the variable is already full, then the agent has already identified a matching contact.

In this new conditional, we're checking the value of the Customer Contact variable again. But now the variable represents a different state, because it's based on the result of the previous conditional. If the Customer Contact variable is still empty after running the Look Up Contact By Email action, that means that the action didn't identify a contact associated with the customer's email address.

6. In the next nested line, type /, and then select **Set a Variable**.
7. Click **Select variable**, and then select **Create a variable**.
8. Enter the following fields. Then click **Create**.

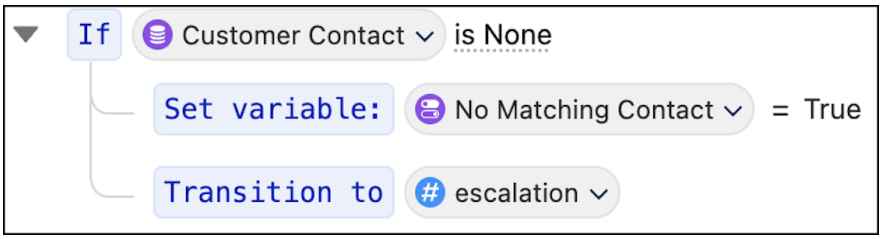
Field	Value
Name	No Matching Contact

API Name	No_Matching_Contact
Data Type	Boolean
Description	Indicates when a customer's email address isn't associated with a valid contact record.
Default Value	False

9. Back on the canvas, for Enter value, enter `True`
   This variable is set to True only when the agent has:
   - Executed the Case Management Subagent
   - Attempted to look up a contact using the Look Up Contact By Email action
   - Failed to find a matching contact

   In the next step, you'll use the variable to customize instructions in the Escalation subagent.
10. Create another line of instructions. Hit Enter.
11. In the next nested line, type `/`, and then select **Transition**.
12. Click **Select subagent**, and then select **Escalation**.
13. Save your changes.

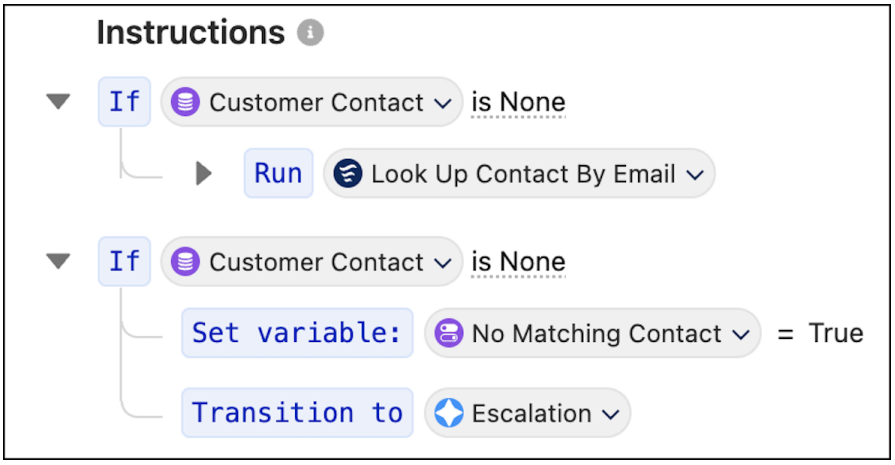
Here's what the expression looks like in Canvas view.



Here's the same expression in Script view, in Agent Script.

```
if @variables.Customer_Contact is None:
    set @variables.No_Matching_Contact = True
    transition to @topic.escalation
```

And here are the two conditional expressions in sequence in the instructions.



Add Conditional Instructions to the Escalation Subagent

Conditional instructions allow you to customize the prompt used in agent reasoning based on the value of a variable. When the conditions are met, the instructions are included in the prompt that's sent to the LLM for agent reasoning. When the conditions aren't meant, those instructions are left out of the prompt. This makes conditional instructions useful for customizing agent behavior to particular customer situations, ensuring that only relevant instructions are sent to the LLM.

Learn More: *Common Patterns: Conditional Instructions*

In the previous step, you configured Case Management instructions to set the No Matching Contact variable to true only when the agent has:

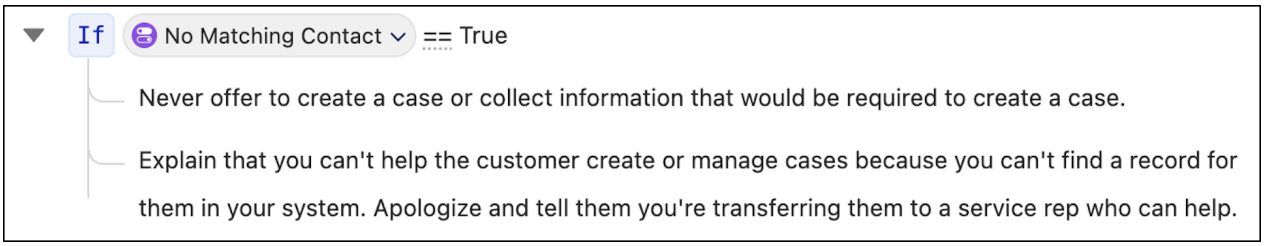
- Executed the Case Management Subagent
- Attempted to look up a contact using the Look Up Contact By Email action
- Failed to find a matching contact

Because the variable represents this specific customer state, Makana can use it to customize instructions in the Escalation subagent to tell the agent how to handle customers who don't have a contact record and want to create a case, distinct from other states that the Escalation subagent can handle. This is particularly useful in a general subagent like Escalation, which covers a range of possible conversational flows and customer issues.

1. In the Explorer panel, click the name of the **Escalation** subagent to open it.
2. At the top of the instructions, create a new line of instructions. Hit Enter. Or you can type `/` to access inline actions, and then select **Reasoning Instructions**.
3. Type `/` to access inline actions, and then select **If/Else (Conditional)**.
4. Click **Select variable**, and then search for and select **No Matching Contact**.
5. Click **Select Value**, and then select **True**.
6. Hit Enter to create a nested line of instructions. Then enter `Never offer to create a case or collect information that would be required to create a case.`
7. Hit Enter to create another line of nested instructions. Then enter `Explain that you can't help the customer create or manage cases because you can't find a record for them in your system. Apologize and tell them you're transferring them to a service rep who can help.`

8. Save your changes.

Here's what the expression looks like in Canvas view.



And here's the same expression in Script view, in Agent Script.

```
        if @variables.No_Matching_Contact == True:
            | Never offer to create a case or collect information that would be required to create a case.
              Explain that you can't help the customer create or manage cases because you can't find a record for them in your system.
Apologize and offer to transfer them to a service rep who can help.
```

Customize the Escalation Subagent to Prepare for Escalation and Escalation Fallback

Makana's agent is now configured to direct customers from the Case Management subagent to the Escalation subagent in two cases:

- When a customer wants to create or manage a case and isn't associated with a contact record.
- When a customer tries to create a case but case creation fails.

There's just one problem–Makana wants to be able to create cases from the Escalation subagent, as a fallback if transfer to a service rep fails. Although it's not *likely* for a customer to be routed from Case Management to Escalation and then have the transfer to a service rep fail, it's an edge case that Makana wants to account for.

Here's the conversational flow Makana wants to build and how you'll build it.

- If a customer is routed to the Escalation subagent and transfer to a service rep fails, in most cases, Makana wants the agent to offer to create a case instead. To configure this use case, you'll add the Look Up Contact By Email and Case Creation actions to the Escalation subagent, map variables to their inputs and outputs, and give the agent instructions for how and when to use them.
- If a customer is routed to the Case Management subagent because case creation with the Create Case action fails, Makana doesn't want the agent to offer to create a case if transfer to a service rep also fails. To handle this, you'll add a follow-up action to set a variable after a customer runs the Create Case action in the Case Management subagent, and you'll store the output of that action in a variable, as well. You'll use these variables to create conditional instructions and a filter on the Create Case action in the Escalation subagent.
- You already built the logic and instructions necessary to define how the Escalation subagent handles customers who don't have a contact record and wanted to create a case. You'll use the Customer Contact and No Matching Contact variables to create a filter to ensure the agent only runs the Look Up Contact By Email action in the Escalation subagent if necessary.

Create and Set Variables in the Create Case Action in the Case Management Subagent

The first step is to create the variables you'll need in the Case Management subagent and specify how to fill them.

1. In the Explorer panel, click the **Case Management** subagent to open it.
2. In Actions Available for Reasoning, expand the Create Case action to view all of the inputs and outputs.
3. For the Case Record output, click **Select Variable**.
4. Select **Create a variable**, and then enter the following fields.

Field	Value
Name	Created Case
API Name	Created_Case
Data Type	Object
Description	The case created for the customer using the Create Case action.

5. Save your changes.
6. Place your cursor at the end of the line for the Case Record output. From the Add to block shortcut, select **Follow-up action**, and then select **Set a variable**.
7. Click **Select variable**, and then select **Create a variable**.
8. Enter the following fields.

Field	Value
Name	Was Create Case Run In Case Management
API Name	Was_Create_Case_Run_In_Case_Management
Data Type	Boolean
Description	Indicates whether the Create Action was run in the Case Management subagent.
Default Value	False

9. Save your changes.

Here's what the Create Case action looks like in Canvas view.



And here's the Create Case action in Script view, in Agent Script.

```
CreateCase: @actions.CreateCase
    with contactRecord = @variables.Customer_Contact
    with caseSubject = ...
    with caseDescription = ...
    set @variables.Created_Case = @outputs.caseRecord
    set @variables.Was_Create_Case_Run_In_Case_Management = True
```

Add the Look Up Contact By Email and Create Case Actions to the Escalation Subagent

Next, add the Look Up Contact By Email and Create Case actions to the Escalation subagent's Actions Available For Reasoning, so the agent can choose when to use them during reasoning with an LLM. When you add the actions to the subagent, you'll:

- Map variables to inputs and outputs to ensure the agent can only fill those inputs and outputs with the specified values.
- Add filters to the actions to limit when the agent can call them.

These deterministic controls also help to guide the agent to run the Look Up Contact By Email and Create Case actions in sequence, when necessary.

1. If you don't already have the Escalation subagent open, in the Explorer panel, click the name of the subagent.
2. In Actions Available for Reasoning, place your cursor after the escalate_to_human utility, and then hit **Enter**.
3. Click **Select action**, and then search for and select **Look Up Contact By Email**.
4. Expand the action to view the inputs and outputs.
5. Just like you did in the Case Management subagent, map the email input to the Customer Email variable and the contact output to the Customer Contact variable.
6. Place your cursor at the end of the line for the Contact record output. From the Add to block shortcut, select **Add filter**.



7. Create this conditional statement: `Customer_Contact is None And No_Matching_Contact == False`

By setting this filter, the Look Up Contact By Email action is only available in the Escalation subagent when the Customer Contact variable is empty and the No Matching Contact variable is set to false. In other words, the agent can only look up the customer's contact record when:

- The agent doesn't already have a contact record for the customer
  AND
- The agent hasn't already tried unsuccessfully to look up the customer's contact record in the Case Management subagent

This filter prevents the agent from unnecessarily running the Look Up Contact By Email action in the Escalation subagent.

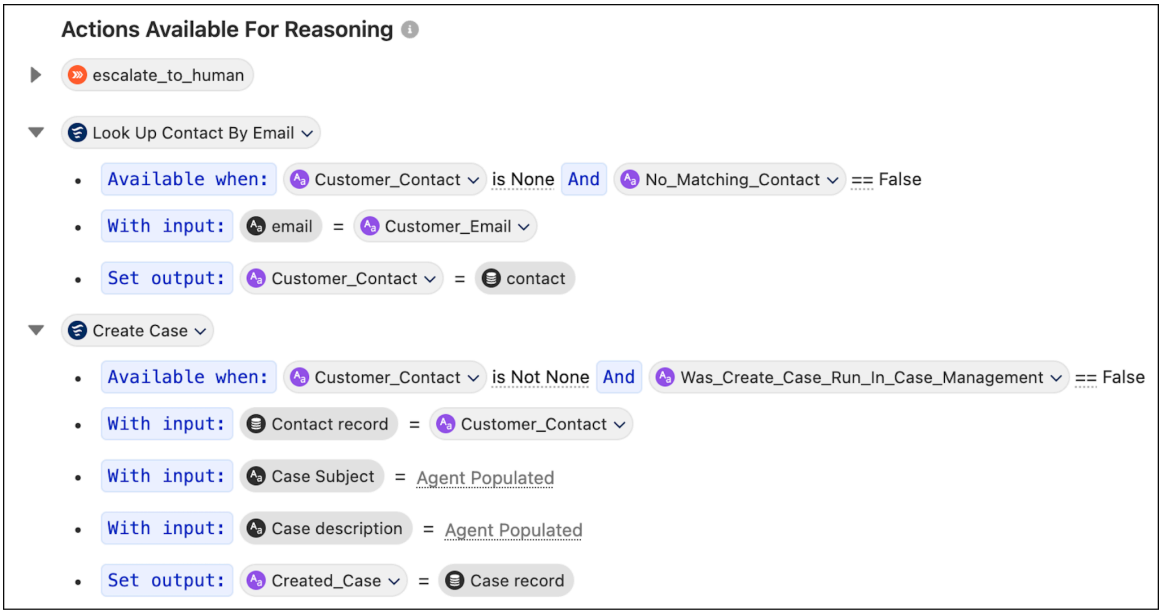
8. Now place your cursor after the contact output, and then hit **Enter**.
9. Click **Select action**, and then search for and select **Create Case**.
10. Expand the action to view the inputs and outputs.
11. Just like you did in the Case Management subagent, map the Contact record input to the Customer Contact variable and the Case record output to the Created Case variable.
    By mapping the Contact record input to the Customer Contact variable, it ensures that the agent can only use the value of the variable to fill the input. If the variable is empty, it's a signal to the agent to run the Look Up Contact By Email action to fill the variable, if the action is available.
12. Place your cursor at the end of the line for the Case record output. From the Add to block shortcut, select **Add filter**.



13. Create this conditional statement: `Customer_Contact is Not None And Was_Create_Case_Run_In_Case_Management == False`
    By setting this filter, the Create Case action is only available in the Escalation subagent when the Customer Contact variable is filled and the Was Create Case Run In Case Management variable is set to false. In other words, the agent can only create a case when:
    - The agent has a contact record for the customer, either from running the Look Up Contact By Email action in the Escalation subagent, from running the Look Up Contact By Email action in the Case Management subagent, or from context
      AND
    - The agent hasn't previously run the Create Case action in the Case Management subagent

    This filter prevents the agent from running the Create Case action if it has already created a case for the customer or if it previously failed to create a case for the customer.
14. Save your changes.

Here's what the reasoning actions look like in Canvas view.



And here's the same configuration in Script view, in Agent Script.

```
actions:
    escalate_to_human: @utils.escalate
        description: "Call this tool if the user indicates that they wish to escalate to a human agent."

    Look_Up_Contact_By_Email: @actions.Look_Up_Contact_By_Email
        description: ""
        available when @variables.Customer_Contact is None and @variables.No_Matching_Contact == False
        with email = @variables.Customer_Email
        set @variables.Customer_Contact = @outputs.contact

    CreateCase: @actions.CreateCase
        description: ""
        available when @variables.Customer_Contact is not None and @variables.Was_Create_Case_Run_In_Case_Management == False
        with contactRecord = @variables.Customer_Contact
        with caseSubject = ...
        with caseDescription = ...
        set @variables.Created_Case = @outputs.caseRecord
```

Add Conditional Instructions

In the previous steps, you:

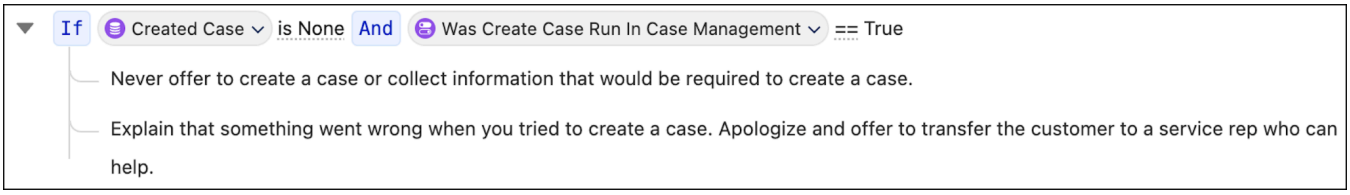
- Mapped the output of the Create Case action to the Created Case variable, in both the Case Management and Escalation subagents. If the variable is filled, it means that the agent successfully created a case for the customer. If the variable is empty, it means that the agent either hasn't tried to create a case or the agent tried to create a case and failed.
- Configured the Case Management subagent to set the Was Create Case Run In Case Management variable to true after the agent runs the Create Case action. If the variable is set to true, it means that the agent ran the Create Case action in the Case Management subagent, either successfully or unsuccessfully, prior to being routed to the Escalation subagent.

The combination of an empty Case Creation variable and a Was Create Case Run In Case Management set to true represents the specific customer state of a failed case creation attempt prior to being routed to the Escalation subagent. Makana can use these variables to instruct the

agent to provide an appropriate message to the customer prior to transferring to a service rep and to avoid trapping a customer in a failed case creation loop.

1. Place your cursor after the If No Matching Contact == True block of instructions, and create a new line of instructions. Hit Enter. Or you can type / to access inline actions, and then select **Reasoning Instructions**.
2. Type / to access inline actions, and then select **If/Else (Conditional)**.
3. Create this conditional statement: If Created Case is None And Was Create Case Run In Case Management == True
4. Hit Enter to create a nested line of instructions: `Never offer to create a case or collect information that would be required to create a case.`
5. Hit Enter to create a nested line of instructions: `Explain that something went wrong when you tried to create a case. Apologize and offer to transfer the customer to a service rep who can help.`
6. Save your changes.

Here's what the expression looks like in Canvas view.



And here's the same expression in Script view, in Agent Script.

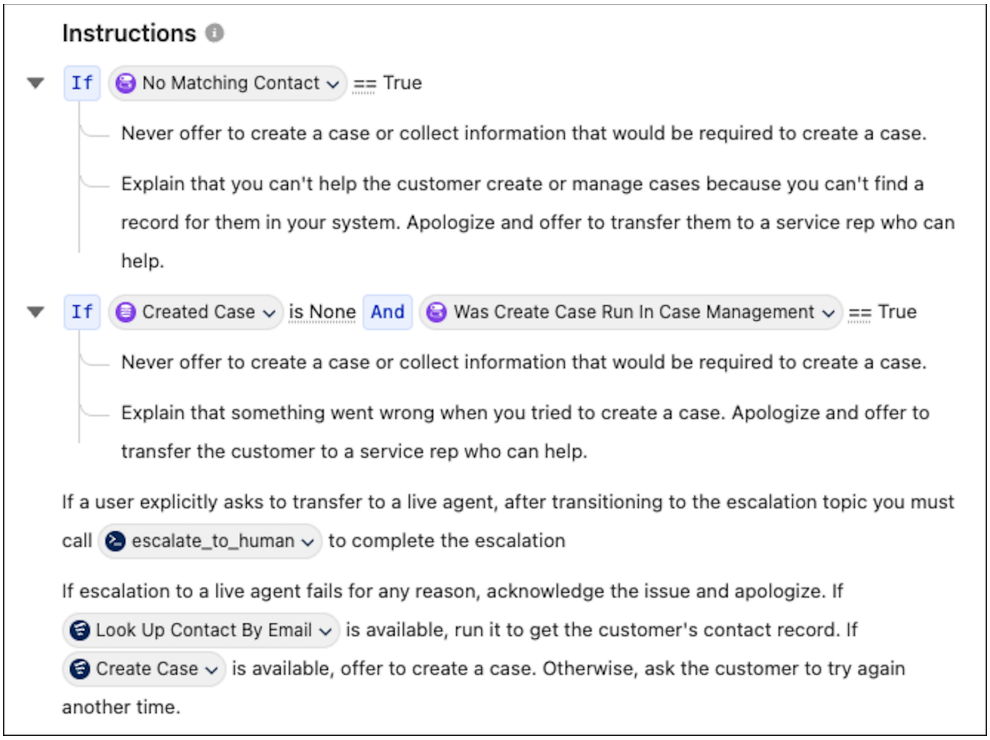
```
if @variables.Created_Case is None and @variables.Was_Create_Case_Run_In_Case_Management == True:
    | Never offer to create a case or collect information that would be required to create a case.
      Explain that something went wrong when you tried to create a case. Apologize and offer to transfer the customer to a service rep who can help.
```

Add Instructions to Tell the Agent How and When to Use the Look Up Contact By Email and Create Case Actions

Finally, replace some conflicting natural language instructions to give the agent more details about how and when to use the Look Up Contact By Email and Create Case actions.

1. On the canvas, find and delete the instruction "If escalation to a live agent fails for any reason, acknowledge the issue and ask the user whether they would like to log a support case instead."
2. On the same line, enter `If escalation to a live agent fails for any reason, acknowledge the issue and apologize. If {!@actions.Look_Up_Contact_By_Email} is available, run it to get the customer's contact record. If {!@actions.CreateCase} is available, offer to create a case. Otherwise, ask the customer to try again another time.`
3. Save your changes.

Here's what the final instructions in the Escalation subagent look like in Canvas view.



Add Permissions to the Agent User

When you completed the guided setup for creating your Agentforce Service agent, you created a new agent user, or a unique user record for the agent. The agent uses this user record to securely access data and perform actions.

The agent user is created with minimal access so that your agent is secure by default. The user includes these properties.

- User License: Einstein Agent
- Profile: Einstein Agent User
- Name: EinsteinServiceAgent User
- Permission Sets: Agentforce Service Agent Secure Base, Makana_Customer_Support_Agent_Permissions
- Permission Set Group: AgentforceServiceAgentUserPsg, which contains the Agentforce Service Agent User, Data Cloud User, and Prompt Template User permission sets
- Permission Set Licenses: Agentforce Service Agent User, Data Cloud, Einstein Prompt Templates

Next, grant the agent the additional access that it needs.

Assign the Agent User the CEO Role

Roles grant users record access via sharing rules and role hierarchies. The agent user requires a role that lets the agent view or edit the records that it interacts with. For Makana's use case, we're

assigning the agent user the CEO role, which is the highest level in our role hierarchy. This gives the agent access to all records in the org that have sharing settings defined.



Note: *When you're creating your own agent, consider the role hierarchy for your organization and give your agent a role that lets it view or edit the records it's necessary for your agent to interact with, following the principle of least privilege.*

1. From Setup, in the Quick Find box, enter Users, and then select **Users**.
2. Click the name of the **EinsteinServiceAgent User**.



Note: *If you have multiple agent users for multiple agents, they'll have the same full name, but you can identify them by their username. To differentiate your agent user from other agent users in your org, edit the agent's user record to give it an identifiable first name.*

3. Click **Edit**.
4. In the Role field, select **CEO**.
5. Save your changes.

Review Your Agent's Current Access and Add Any Additional Permissions

In addition to the access that an agent user has by default and by role assignment, most agent users require:

- Permissions required to run agent actions
- The minimum level of object permissions for each object that the agent interacts with via flows, Apex, or prompt templates
  When you add a new action to your agent, make sure that the agent user has access to the objects referenced in the action.
- Channel-specific permissions, depending on how you deploy your agent to one or more channels
  In the Ideate section of this guide, you decided to connect your agent to Enhanced Chat, so we've included the permissions for that channel here. You'll learn more about deploying your agent to channels in a later section.

Review your agent user's current access and ensure that they have the necessary permissions and object permissions.

Feature Your Agent Uses	Required Permissions	Access Granted By
Agent actions that run flows (for example, the Create Case action)	App Permission: Run Flows	Profile: Einstein Agent User
Agent actions that run prompt templates (for example, the	System Permission: Execute Prompt Templates	Permission Set License: Agentforce Service Agent User

Answer Questions with Knowledge Action)		Permission Set Group: AgentforceServiceAgentUserPsg (includes the Prompt Template User permission set)
Agentforce Data Library	Permission Set: Data Cloud User	Permission Set Group: AgentforceServiceAgentUserPsg
Case Object	Case Object: Read, Create, Edit, View All, Edit All	Read and Create access is granted by default by the Agentforce Service Agent Secure Base permission set. To expand Case object access, edit the Makana_Customer_Support_Agent_Permissions permission set.
Contact Object	Contacts Object: Read, View All	Read and Edit access is granted by default by the Agentforce Service Agent Secure Base permission set. To expand Contact object access, edit the Makana_Customer_Support_Agent_Permissions permission set.
Enhanced Chat Channel Access	Messaging Session Object: Read, Edit	Permission Set: Agentforce Service Agent Secure Base



Learn More: *This agent's configuration doesn't include Apex classes, but they're commonly used for agent actions. Many agents also use Salesforce Knowledge.* *Review the required agent user permissions for these features and more.*

To add permissions, edit the existing Makana_Customer_Support_Agent_Permissions permission set, which was created for you when you completed the new agent guided setup. If you want to create a different permission set for your agent user, make sure that it's associated with the Einstein Agent license and Einstein Agent User profile.

Test Your Agent

Makana Medical Devices is now entering the testing phase of their new Agentforce Agent. After integrating AI into their Salesforce workflows, this stage focuses on ensuring the agent performs reliably before it goes live.



AI agents combine deterministic expressions and non-deterministic reasoning. To validate both, Makana tests across a range of scenarios to confirm consistent, accurate performance. They need to verify that the agent reliably selects the right subagents and actions to complete tasks, and that its responses align with Makana's tone, brand identity, and business needs.

To do this, Makana uses a combination of manual testing in Agentforce Builder and automated batch testing in Agentforce Testing Center. These tools allow the team to validate agent behavior, measure performance at scale, and identify areas for improvement before deployment.

But testing doesn't stop once the agent is live. Maintaining agent performance requires an ongoing process of tracking results, identifying issues, and retesting improvements over time.

We've organized Makana's testing strategy into three phases:

- Planning and Preparation: Define success metrics, create realistic test scenarios, and identify the data needed for evaluation.
- Core Testing: Validate agent behavior through manual testing and batch evaluation to ensure it performs as expected.
- Iteration and Refinement: Track performance, troubleshoot issues, and continuously improve the agent through targeted updates and retesting.

Together, these phases create a structured testing lifecycle that helps ensure the agent is not only ready for deployment, but continues to improve as it evolves.

Starting here? *Make sure you have the right permissions for the job.*

Planning and Preparation

Comprehensive testing begins with clearly defining what success looks like for your AI agent, using measurable criteria aligned with your business objectives. Preparation also involves breaking down the agent into core components, like subagents and actions, and identifying the most relevant data for evaluation. Finally, this phase focuses on designing structured test scenarios that reflect real-world customer interactions.

Define Objectives and the Testing Strategy

Before testing begins, clearly define what constitutes success for your AI agent.

- Set goals linked to business value: Establish specific, measurable, attainable, and relevant goals for your agent's performance, directly linked to the business objectives you identified for your agent.
- Deconstruct the agent: Break down the agent into core components such as subagents and actions. Testing these parts individually helps to pinpoint inefficiencies at their source.

To shape your testing goals, revisit the strategic goals you set during the ideate stage. Makana's agent is focused on responding to common customer questions and streamlining case management. They identified business objectives related to agent effectiveness and interaction, and agent impact, early in the process and use those to determine if the agent is on track.

Once you've outlined your objectives, the next step is to create a testing strategy. You can approach this a number of ways, but your strategy should:

1. Identify test scenarios and set evaluation parameters.
2. Run the tests and validate the results.
3. Make evidence based adjustments and repeat tests until your agent is working as expected.
4. Test strategically and on a regular cadence to make sure your agent is still performing the way you intend.

We'll walk through how Makana approaches each of these steps to test their service agent.

Create Test Scenarios and Identify Data Requirements

A testing scenario is a structured example used to evaluate how well an AI agent, like the Makana Customer Support Agent, performs in a realistic situation. It simulates a real-world customer interaction and helps you test whether the agent understands the user input correctly, selects the right subagent, runs appropriate actions, and responds accurately, completely, and in the right tone. A strong testing scenario should include:

- **Scenario description:** A short narrative of the situation or customer context, such as a customer who just received their device and is confused by an error code.

- **Test utterance:** The actual phrase or question the customer might enter in natural language.
- **Expected subagent:** The subagent the agent should classify the utterance into based on how the agent is configured. Or the subagent the agent should route the conversation to based on deterministic transitions.
- **Expected action:** The specific action the agent should take in response, like “lookup information about Makana's products and policies” or “escalate the conversation to a service rep.”
- **Expected response:** A brand-appropriate reply based on validated information. The agent’s wording may vary but as long as the tone feels right and the info is accurate, that’s what really counts.
- **Utterance variations:** The different ways that customers are likely to phrase a question or request, including utterances that vary based on language differences, different emotional states of the customer, and utterances that change the topic of the conversation.
- **Multi-turn:** It’s important to test how the agent handles requests inside of continued conversation.

To help you come up with good test scenarios, review the subagents assigned to your agent. Look at the classification description and fields that describe the capabilities and parameters your agent should operate within and each instruction that directs how the agent performs. Next, write or generate inputs that test against these details to help make sure your agent acts reliably in each scenario. To give you an idea of what that looks like, here’s a sample of Makana’s inputs.

- How do I turn on my CGM-3000?
- My continuous glucose monitor is showing an error. How do I fix it?
- What products do you have for monitoring high blood pressure?

And while these are a good start, it’s also important to think in terms of how a conversation might flow. Users might ask the same question differently, so use paraphrasing to make sure the agent is recognizing the core request. For example:

- How do I turn on my CGM-3000?
- My CGM-3000 won't power on. What should I do?
- Can you show me the instructions for starting the CGM-3000?

Users might also express themselves differently depending on how they feel. Create inputs that cover polite, frustrated, and neutral states, like:

- I'm having a problem with my continuous glucose monitor. How do I fix the error?
- My continuous glucose monitor is showing an error! This is awful, I need to fix it now!
- My device is showing 'E-24.' How do I troubleshoot this error?

You also want to make sure your agent can handle multi-turn conversations and follow along when a user shifts subtopics. For example:

- Turn 1 (Product Query): What products do you have for monitoring high blood pressure?
- Turn 2 (Follow-up): How do the Pro BP-200 and the BP-300 models compare?
- Turn 3 (Unrelated Shift): By the way, can you call my doctor's office to get my prescription refilled?

Testing Scenario 1

Scenario	Warranty Inquiry
Summary	A customer who just bought the CGM-3000 device is trying to register their product and wants to confirm how long the warranty lasts.
Expected Behavior	
Define how the agent should understand and respond to this scenario. This ensures clarity when evaluating test results.	
Field	**Description**
Expected Subagent	Support FAQ
Expected Action	Answer Questions with Knowledge
Expected Response	“The CGM-3000 is covered by a 12-month limited warranty from the date of purchase. It includes protection against manufacturing defects and device malfunctions under normal use. If you’d like to register your product or need more details, I can help with that!”
Utterance Variations	
Include different tones and reworded inputs to test recognition of intent and emotional range.	
Variation Type	Test Utterance
Neutral	“How long is the warranty on the CGM-3000?”
Polite	“Hi there! Could you tell me the warranty period for the CGM-3000?”
Frustrated	“I just bought this thing – what kind of warranty even comes with it?”
Rephrased	“Does the CGM-3000 come with any kind of guarantee?”
Rephrased	“What’s the coverage period if my CGM-3000 breaks?”
Multi-Turn Conversation Flow	
Outline how the conversation should progress over multiple turns, including topic shifts.	

Turn	User Input	Agent Response
1	“What’s the warranty on the CGM-3000?”	Respond with accurate info and brand-appropriate tone. "The

		CGM-3000 comes with a 1-year limited warranty that covers manufacturing defects and malfunctions under normal use."
2	“What does the warranty not cover?”	Explain exclusions clearly. "The warranty does not cover damage caused by misuse, accidents, or unauthorized repairs. It’s designed to protect against defects in materials and workmanship."
3	“Can I register the device with you now?”	Offer next steps or link to the registration process. "Absolutely! I can help you with that. Please provide your purchase details, or I can send you the registration link if you prefer."
4 (Topic Shift)	“By the way, do you sell replacement sensors for this model?”	Detect the shift and transition to the Product Availability subagent. "Great question! Yes, we do offer replacement sensors for the CGM-3000. Would you like me to help you find them or place an order?"

Testing Scenario 2

Scenario	Case Status Inquiry
Summary	A customer contacts the agent asking for an update on a previously submitted support case regarding their glucose monitor. They expect the agent to retrieve their case details and provide a current status. If the agent fails to locate or correctly respond with case information, the customer may get frustrated or escalate the issue.

Expected Behavior	
Define how the agent should understand and respond to this scenario. This ensures clarity when evaluating test results.	
Field	**Description**
Expected Subagent	Case Management
Expected Action	Get All Cases for Contact

Expected Response	"I see a case regarding your glucose monitor return was submitted last week and is currently being processed. You'll get an update shortly."

Utterance Variations

Include different tones and reworded inputs to test recognition of intent and emotional range.

Variation Type	Test Utterance
Neutral	"Can you check if my glucose monitor return case was received?"
Polite	"Hi, I'm wondering about the status of my support case for the glucose monitor I returned last week."
Frustrated	"I sent back my glucose monitor and haven't heard anything. What's going on with my case?"
Rephrased	"Has my case for the glucose monitor replacement been processed yet?"
Rephrased	"I submitted a return last week. Can you tell me what's happening with that?"

Multi-Turn Conversation Flow

Outline how the conversation should progress over multiple turns, including topic shifts.

Turn	User Input	Agent Response
1	"Can you check on my glucose monitor return case?"	Retrieve existing cases. "I'm checking your cases now. I see a case was submitted last week for your return and it is currently being processed."
2	"What's the expected resolution time?"	Provide case status and timeline. "I'm checking your case history now and preparing next steps. You should have a new email with your order details."
3	"I haven't received any emails yet."	Escalate the case for priority handling. "I'm escalating your case for priority review and will notify you of any updates."
4 (Topic Shift)	"Thanks. Can you confirm my contact info is correct?"	Transition to confirm customer contact details. "Sure! I have your email as

		user@example.com. Is that still the best way to reach you?”

As you go through this process, keep in mind that a good test includes cases that cover 3 to 5 core scenarios and 2 to 3 edge cases. With this foundation in place, you’re ready to move into core testing and iteration, where you’ll validate performance, identify gaps, and refine the agent’s behavior.

Core Testing

Core testing is essential for validating your AI agent before deployment. This phase focuses on ensuring the agent behaves as expected by thoroughly testing its ability to interpret user input, select the correct subagents and actions, and generate accurate, relevant responses.

This process begins with single-session testing in Agentforce Builder, where you can manually validate agent behavior and troubleshoot issues in detail. Once the agent performs reliably in individual scenarios, testing can be scaled using batch tests in Agentforce Testing Center. Together, these approaches ensure the agent is well-tested, consistent, and ready for real-world use.

Batch testing provides a broader view of performance across diverse scenarios, helping identify patterns, edge cases, and gaps that may not surface during manual testing.

Manual Testing and Verification

When you first create an agent, it’s essential to confirm that the agent performs as expected. Manual testing is the starting point for validating your agent’s performance. It offers a detailed, step-by-step evaluation of your agent's interactions and utterances and is especially effective for confirming configurations and exploring how the agent responds to different inputs. Throughout the agent lifecycle, plan to revisit manual testing in Agentforce Builder, especially after making changes to your agent.

Testing and iteration are driven by the agent versioning lifecycle, which begins with creating a draft version of your agent. When an agent version is in draft state, you can edit it and your changes are saved to the current agent version. Then, you can test the agent quickly by previewing it. Committing an agent version saves a snapshot of your agent version to metadata, and activating the agent version makes it available for use. To iterate on your agent after you’ve committed a version, create another draft version of your agent.

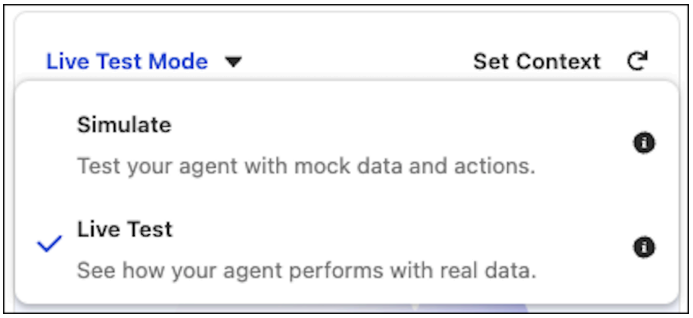


Learn More: *For more information about draft agents, see Versioning and Editing Agents.*

In manual testing, you'll preview a conversation with your agent, then track your agent's reasoning and behavior in the Interaction Summary. You'll drill into the details of the interaction with trace, variables, and a little help from Agentforce. In Agentforce Builder in Agentforce Studio, the preview experience is designed to give you all of the behind-the-scenes details of how your agent is working, so it's easier to identify unexpected behavior and improve your agent.

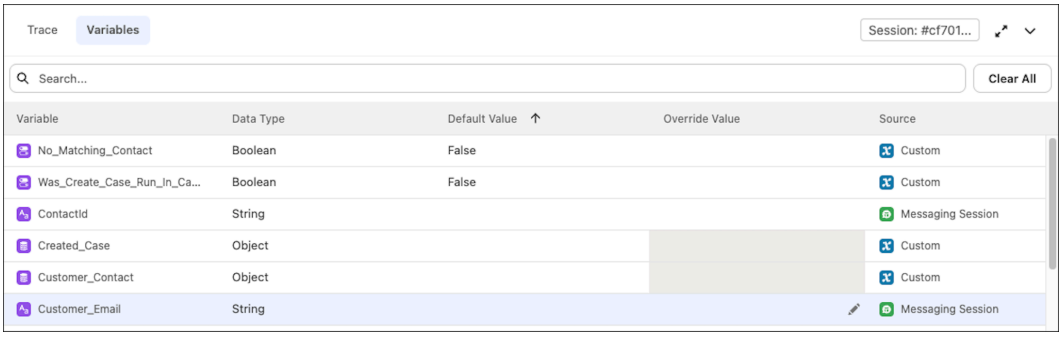
Makana Medical Devices has built their agent to handle general FAQ questions from customers, such as inquiries about product availability, warranty policies, and basic troubleshooting for their medical devices. Before fully rolling out the agent, the support team wants to confirm that it accurately understands and responds to common questions.

1. From the Agents page in the Agentforce Studio app, open the Makana Customer Support Agent in Agentforce Builder.
2. Click **Preview**.
3. Select your test mode. If you're working in a Developer org that you created to work through this guide, we recommend using Live Test mode. This gives you the most complete information about your agent's performance. Keep in mind that in Live Test mode, your agent can modify your data (for example, update a case), but that's safe to do with placeholder data in a dedicated environment.



Learn More: *For more information about testing modes, see [Simulate and Live Test Modes].*

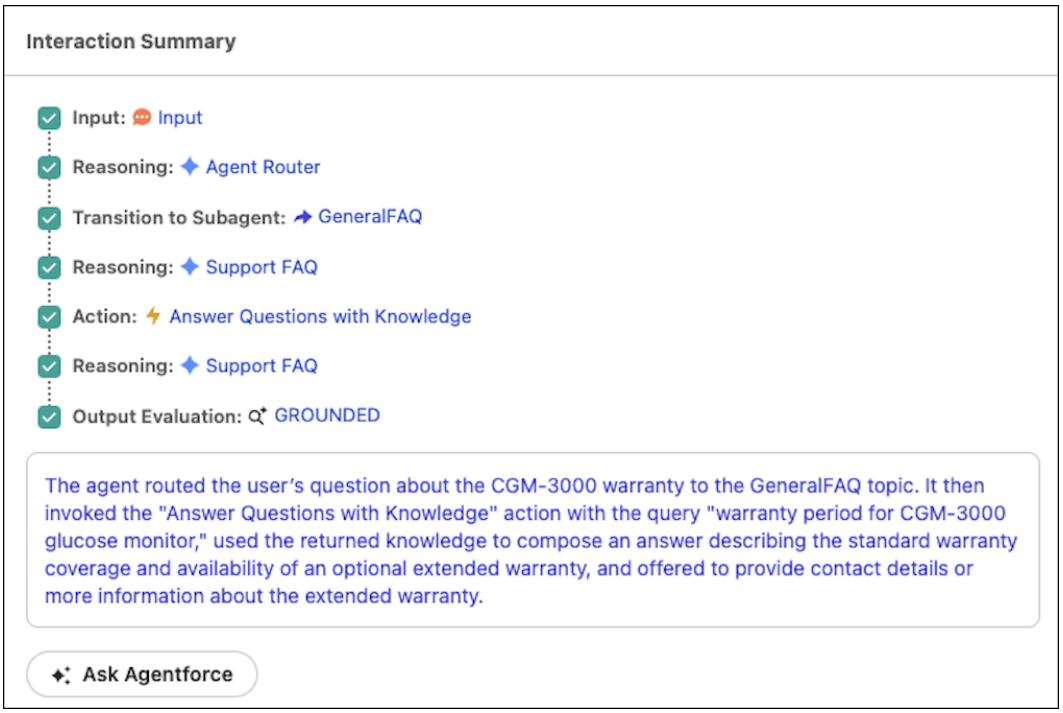
4. To specify the context of an agent conversation, click **Set Context** on the Preview or Interaction Summary panels, or open the Variables tab and edit the variable values. You can set or override the value of variables to match the customer's actual context.



Note: *You don't have to set variables for all of the example test scenarios we've provided for Makana's agent. For example, for general FAQ scenarios, the agent doesn't require an email address.*

To test any case management scenarios, provide an email address for the Customer Email variable. For example, to test subagent routing from Case Management to Escalation and conditional instructions, specify an email address that's associated with a contact. Then in a new conversation, specify an email address that's not associated with a contact.

5. In the preview conversation, enter a typical customer question: `What is the warranty period for your CGM-3000 glucose monitor?`
6. To review the agent's behavior and reasoning, start with the Interaction Summary.

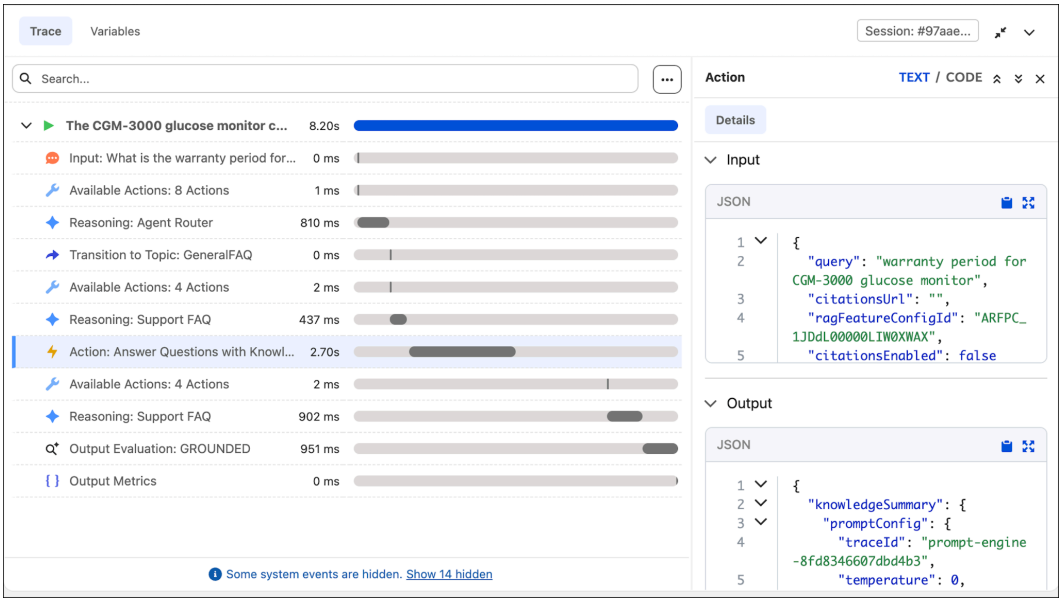


The Interaction Summary shows you a high-level overview of the steps your agent took to generate a response, including the result of each. For example, each reasoning step shows you the result of a prompt sent to the LLM, and each transition shows you which subagent the agent routed the conversation to. You can hover over each step for more details. The Interaction Summary also includes an AI-generated natural language explanation of your agent's process, which helps put the steps in context.

7. To drill down, click on any step in the interaction summary. This opens the full trace data for the interaction, which includes each agent event, the time elapsed for the event, and all of the technical details related to that event.

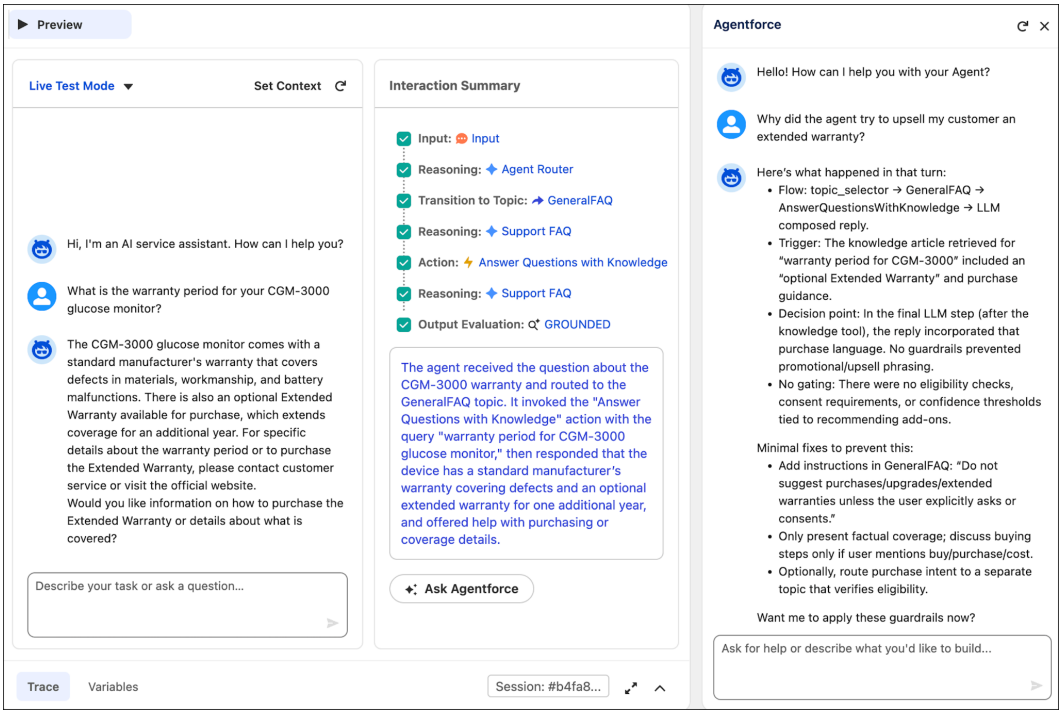
   The agent events in the trace data are more granular than the overview of steps in the Interaction Summary, which makes them particularly useful for troubleshooting. And you can also view the details for each event. For example, for each agent action, you can view

exactly what the agent used to fill each input and what the underlying reference action provided as output. When troubleshooting, this can help you differentiate between issues with your agent's instructions, the action descriptions, or the configuration of the underlying reference action.



8. Use the Variables tab to track the state of any variables in your agent at the start and the end of the interaction. In this simple scenario of Makana's agent answering a frequently asked question, no variables are evaluated or changed. But in a case management scenario, Makana's agent uses variables in conditional instructions, conditional transitions, filters, and action inputs and outputs. Tracking the value of variables at the start and end of an interaction can help you evaluate whether your agent has the right context, how well your agent is following instructions, and the accuracy of your agent's responses.

9. Get AI assistance to ask questions about your agent's behavior or even make improvements. In the Interaction Summary panel, click **Ask Agentforce**. We send the integrated Agentforce assistant information about the preview session, so you can ask it questions or even ask it to make changes for you.

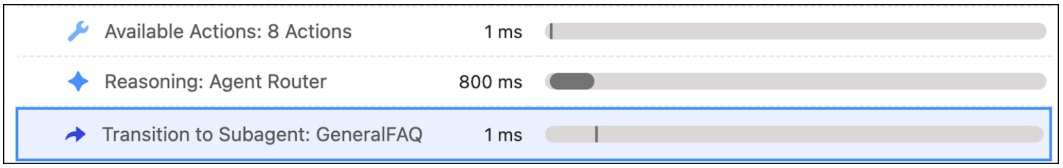


10. Make any necessary adjustments to your agent. Then restart the conversation to test your changes. Continue this iterative process until your agent consistently delivers accurate and helpful answers.

Based on this scenario, here are some ways you can test and validate Makana's agent manually in Agentforce Builder.

- **Check the agent's subagent classification and routing.**
  At the start of the conversation, Makana's agent uses the Agent Router for subagent classification to determine which subagent best matches the customer's utterance. If the agent correctly identifies the Support FAQ subagent, the Agent Router and the name and description of the Support FAQ subagent are working as expected.

  If the agent routes the conversation to an unrelated subagent, review these agent events and details in the trace for the interaction.



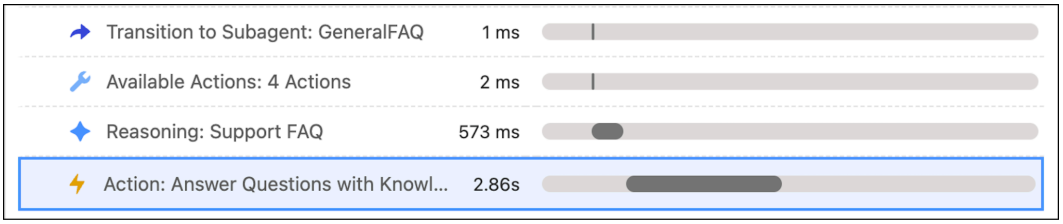
  Available Actions shows the possible transitions available to the agent in the Agent Router. If a transition to a subagent has a filter applied and the filter conditions aren't met, it's not included in the list of available actions. Reasoning shows the prompt sent to the LLM to guide the agent's subagent classification. Transition shows the agent's decision to route the conversation to the General FAQ subagent.

  Here are some troubleshooting steps you can take.

- Review and adjust the subagent's description to better guide the agent. Make sure the description includes unique key terms and doesn't overlap with other subagent descriptions.
- Review and adjust instructions in the Agent Router.
- Review and adjust any filters on transitions to the Support FAQ or other subagents in the Agent Router, to ensure customers have access to the subagent as expected.

- **Check the agent's action selection.**
  Makana's agent uses reasoning to select and run actions within the Support FAQ subagent. If the agent correctly identifies the Answer Questions with Knowledge action, the instructions in the Support FAQ subagent and the descriptions in the Answer Questions with Knowledge action are working as expected.

  If the agent chooses an unrelated action, review these agent events and details in the trace for the interaction.

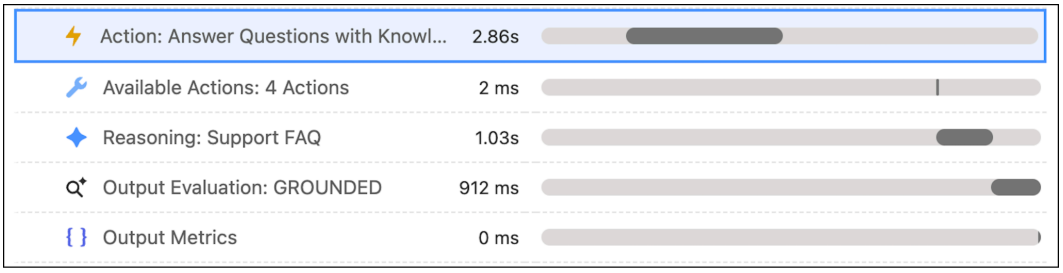


  Transition shows the beginning of the agent's processing of the General FAQ subagent. Available Actions shows which actions are available to the agent to choose during reasoning. If an action has a filter and the filter conditions aren't met, it's not included in the list of actions. Reasoning shows the complete prompt the agent sends to the LLM to select an action. Action shows the agent's decision to run the Answer Questions with Knowledge action.

  Here are some troubleshooting steps you can take.
  - Review and adjust instructions in the Support FAQ subagent and add determinism (for example, a direct reference to the action), if necessary.
  - Review and adjust the description of Answer Questions with Knowledge action.
  - Review and adjust any filters on the Answer Questions with Knowledge or other actions in the Support FAQ subagent, to ensure customers have access to the action as expected.

- **Verify any action outputs.**
  All agent actions call a specific flow, Apex class, prompt template, or other Salesforce functionality. Issues can stem from underlying functionality rather than any agent configuration. In this case, the Answer Questions with Knowledge action simply provides information, so you can check the agent's response in the preview conversation and compare it to Makana's official warranty policy. But for other actions, such as Create Case, you can manually verify that a new case was actually created in Salesforce with all of the correct information populated from the conversation.

If the agent provides incorrect information, review these agent events and details in the trace for the interaction.



Action shows exactly what the agent used to fill each input and what the underlying reference action provided as output. When troubleshooting, this can help you differentiate between issues with your agent's instructions, the action descriptions, or the configuration of the underlying reference action. Available Actions and Reasoning are events that show the agent reasoning after running the action, to determine whether to respond to the customer or run additional actions. Output Evaluation and Output Metrics show the results of the agent's response validation checks. The agent checks whether the response is grounded in data sources, adheres to the subagent's scope and instructions, and doesn't include hallucinations, unverified information, or potential prompt injection risks.

Here are some troubleshooting steps you can take.

- Update the product information PDF. Consider refining the source content in the data library or limiting the scope of sources that your action pulls from.
- Adjust the Support FAQ subagent instructions to add clarifying details to guide the agent to reference the most recent warranty policy documents.
- Adjust subagent instructions, action descriptions, or the configuration of the underlying reference action as needed. Consider adding determinism to control agent responses, such as storing outputs in variables.

Keep in mind that these are suggestions for a relatively simple testing scenario that relies mostly on agent reasoning and not on deterministic instructions. For other scenarios, the procedure may vary but the approach remains the same. The robust preview experience in Agentforce Builder in Agentforce Studio helps take the guesswork out of validating, refining, and troubleshooting even the most complex agent configurations.

Batch Testing and Automated Scoring

Testing individual utterances in Agentforce Builder is a good strategy for targeted testing, subagent and action refinement, and troubleshooting. But it can be time consuming if you want to get an overall idea of your agent's reliability across a broader set of real-world scenarios. We can take advantage of automation in the testing process to help you efficiently launch reliable and trustworthy agents. In the Testing Center, you can streamline large-scale batch testing by generating up to 100 realistic test scenarios using AI.

Once Makana Medical Devices completes the single session testing of their customer support agent, they can move on to batch testing to evaluate how well the agent performs at scale. Using

the Agentforce Testing Center, the Makana team can quickly generate hundreds of test cases, run tests, and narrow down where their agent needs improvements. For AI-generated testing, the team can create:

- Subagent- and action-based test cases that simulate real user intents and workflows.
- Knowledge-based scenarios that generate question-and-answer pairs from the agent's knowledge base to assess accuracy and knowledge retrieval.
- Custom prompt–driven test cases that allow the team to define exactly what to test, tailoring scenarios to their specific goals.
- Tests generated from imported data, including builder sessions and real customer interactions.

Together, these methods provide flexible testing options, allowing the team to choose the approach that best fits each scenario and objective.

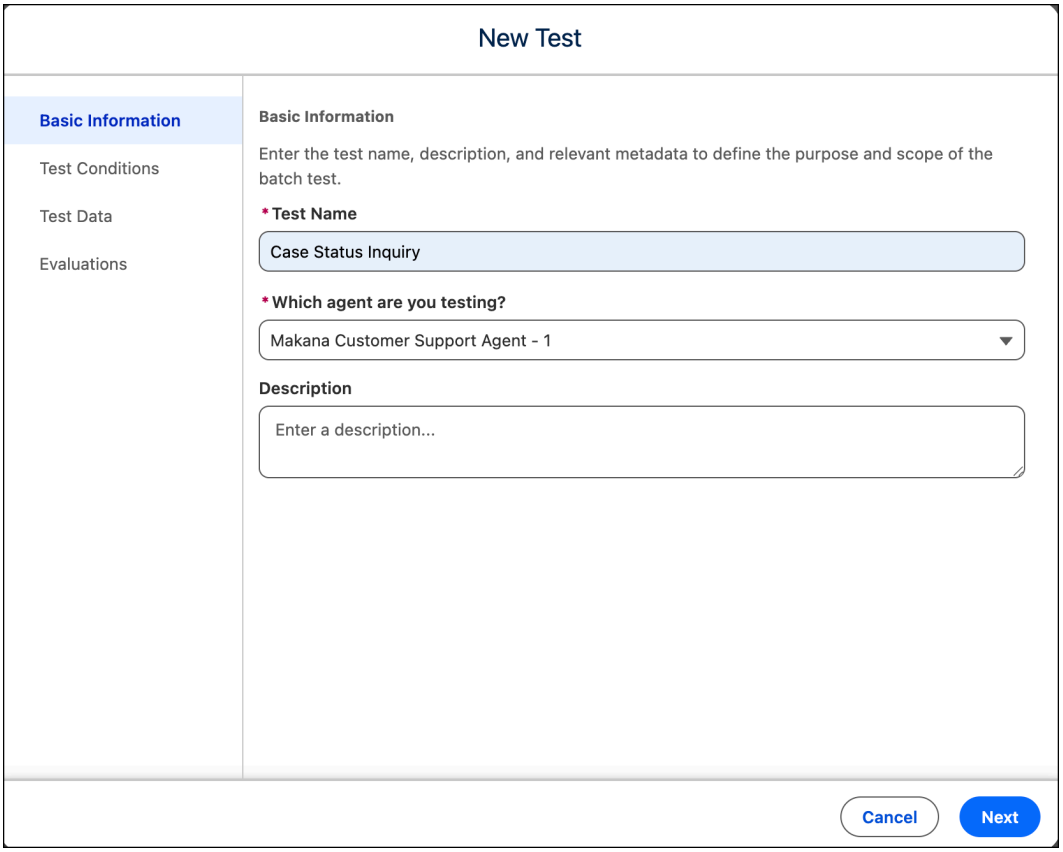
Makana puts together a batch test plan to organize their testing strategy and ensure coverage across their scenarios. They address their Case Status Inquiry scenario first.

Batch Test Plan		
Scenario	Case Status Inquiry	
Agent	Customer Support Agent	
Test Type *Consider the type of tests you want to generate.*	☑ Subagents and Actions *Use when you want to verify that your agent correctly routes conversations using predefined subagents and actions, such as opening cases, escalating issues, or updating records.* ☐ Knowledge-Based Question and Answer *Use to verify the agent retrieves accurate, conversational answers from knowledge bases, like uploaded PDFs or knowledge articles.*	
Goal *What are you trying to validate? What specific knowledge do you want the agent to retrieve and how should it be presented?*	Ensure the agent retrieves and communicates accurate case status information from Salesforce objects using structured subagents and actions.	
Agent Setup *Which agent are you testing and what actions, subagents or resources is it connected to?*	Subagents: • Agent Router • Support FAQ • Case Management • Escalation • Off Topic • Ambiguous Question	Data Library Resources: • Makana Products and Policies

Batch Test Plan		
	Actions: • Get All Cases for Contact • Get Case by Case Number • Look Up Contact by Email • Create Case • Add Case Comment • Answer Questions with Knowledge	
Test Description *Describe the purpose of the test and scope like what the utterances should trigger or cover. Include key terms, tone types, and any constraints.*	A customer contacts the agent asking for an update on a previously submitted support case regarding their glucose monitor. They expect the agent to retrieve their case details and provide a current status. If the agent fails to locate or correctly respond with case information, the customer may get frustrated or escalate the issue.	
Context Variables *Identify and include the variables that complement your scenario.*	Contact Id	
Response Quality Scorers *Choose what to measure based on the scenario. Describe the purpose of each scorer in your test.*		
☐ **Completeness**		
☑ **Coherence**	Selected to assess if the agent transforms underlying information from Salesforce Objects into grammatically correct, conversational language.	
☑ **Conciseness**	Selected to ensure the status update is short but accurate, successfully capturing the essence of the desired content.	
☐ **Latency**		
☐ **Instruction Adherence**		

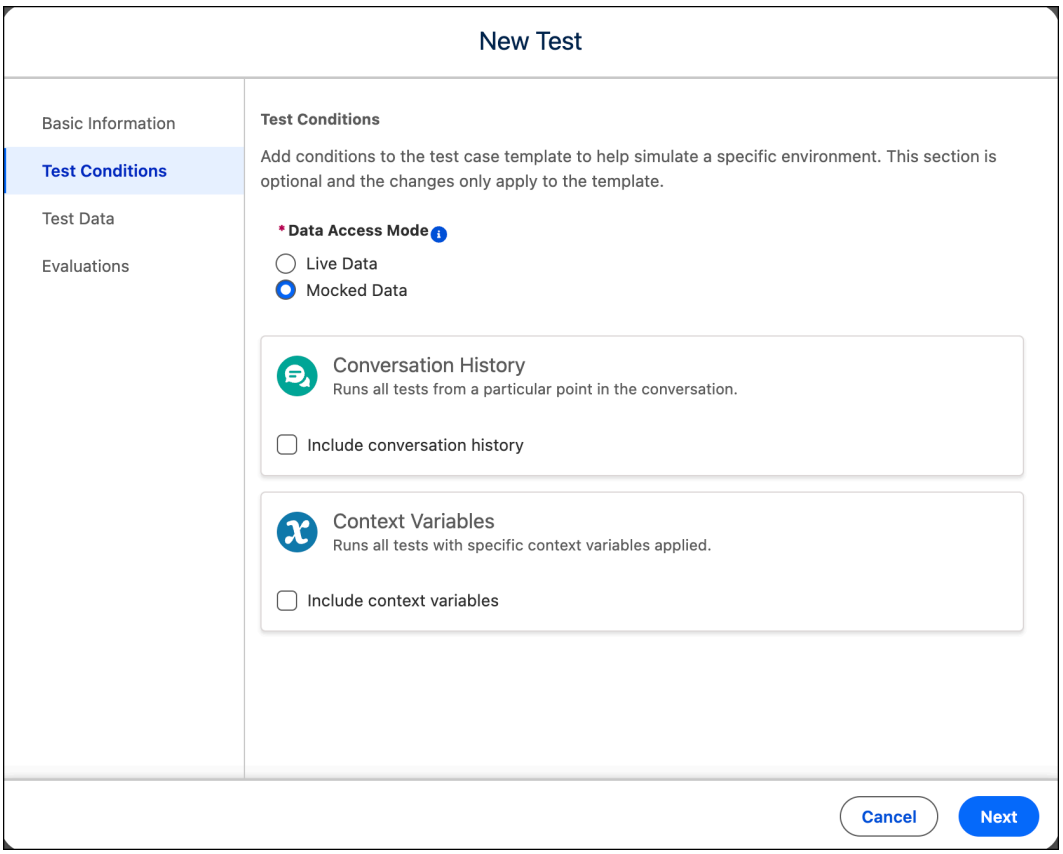
Based on this plan, Makana follows these steps to create their batch tests.

1. To begin batch testing, in Agentforce Studio, click **Tests**.
2. Click **New Suite**, then Name the test `Case Status Inquiry` and select Makana Customer Support Agent. Optionally, enter a description. Click **Next**.

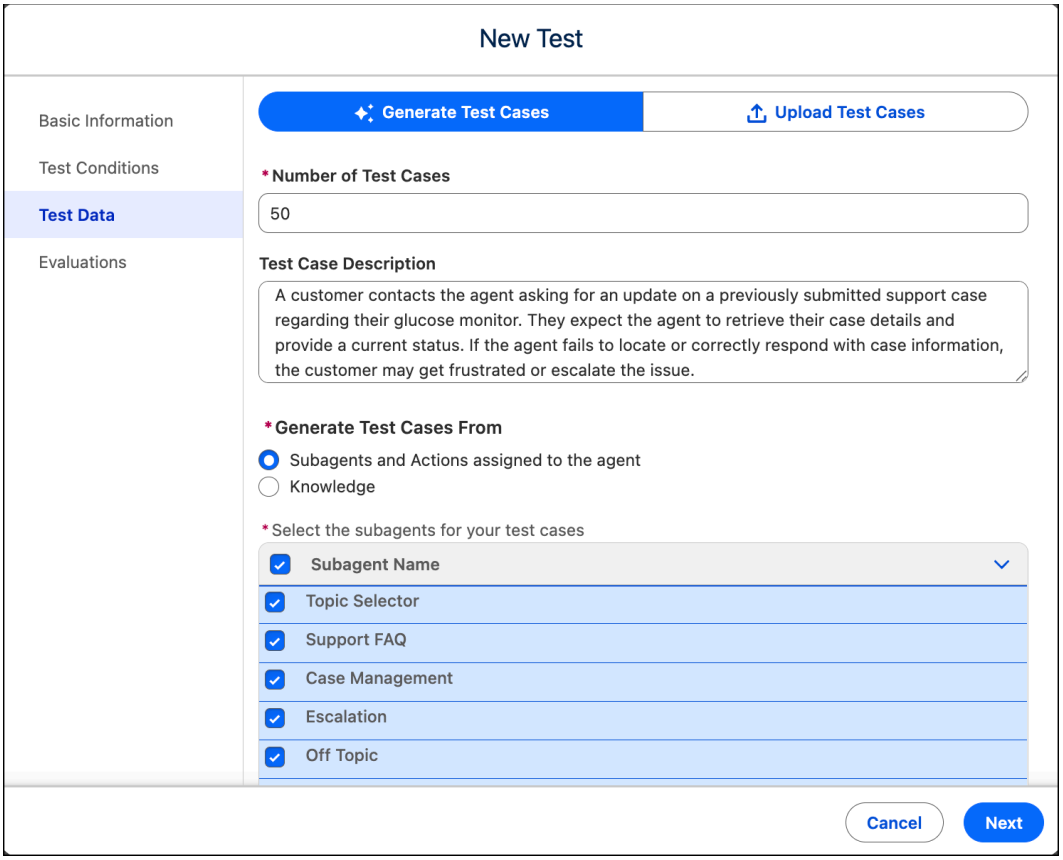


When generating test cases, conversation history and context variables can't be included, so Makana skips adding these to prioritize test generation.

3. In test conditions, they select **Mocked Data** since their goal is to validate conversation flow, subagent selection, and action logic without impacting real Salesforce data. Click **Next**.



4. To quickly create test cases, Makana selects **Generate Test Cases**. They want a broad selection of tests, so ask the system to create 50 tests.

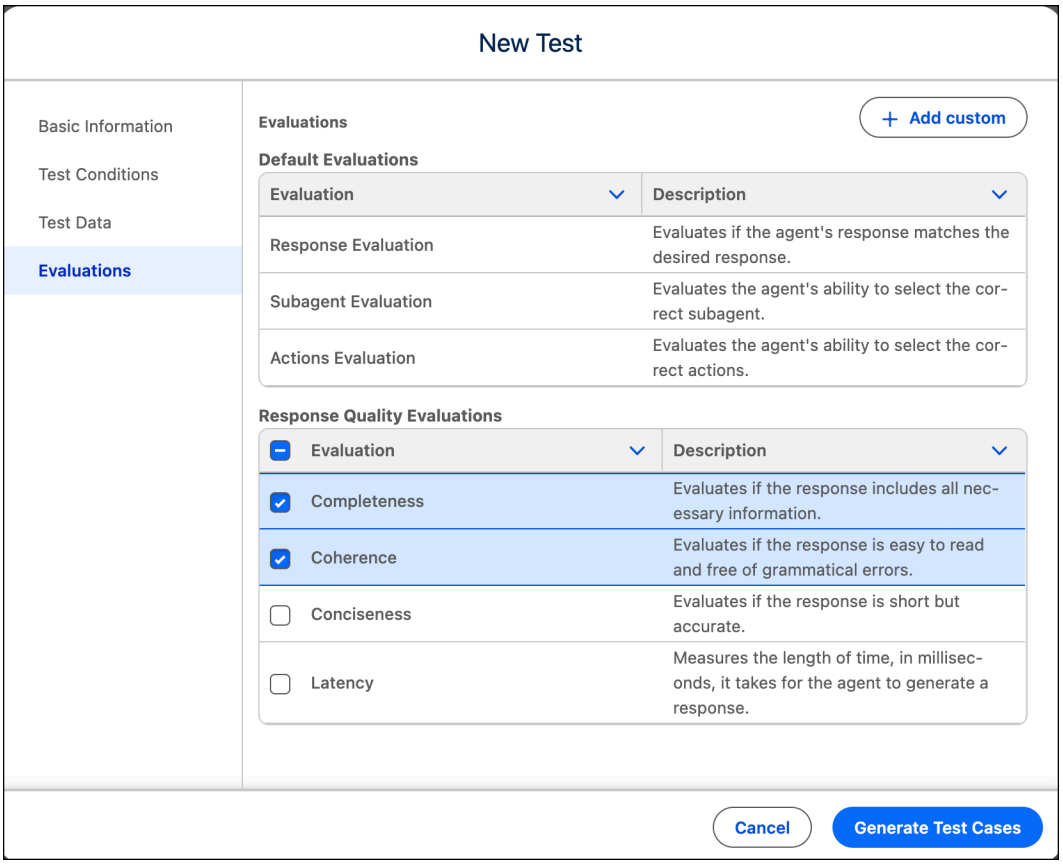


5. To guide the AI in generating appropriate test cases, enter a test description:
   `A customer contacts the agent asking for an update on a previously submitted support case regarding their glucose monitor. They expect the agent to retrieve their case details and provide a current status. If the agent fails to locate or correctly respond with case information, the customer may get frustrated or escalate the issue.`
6. Makana opts to generate test cases based on the subagents and actions tied to the business objectives in their Case Status Inquiry testing scenario. So, select **Subagents and Action assigned to the agent**.
7. Makana wants to start broad and identify any gaps in how the agent handles different case-related queries. To do this, include all available subagents: **Agent Router**, **Support FAQ**, **Case Management**, **Escalation**, **Off Topic**, and **Ambiguous Question**. Click **Next**.

In the Case Status Inquiry scenario, Makana's testing goal is to ensure the correct subagent and action are selected, and that case data retrieved from Salesforce is communicated to the customer in a clear, conversational, and brand-aligned manner.

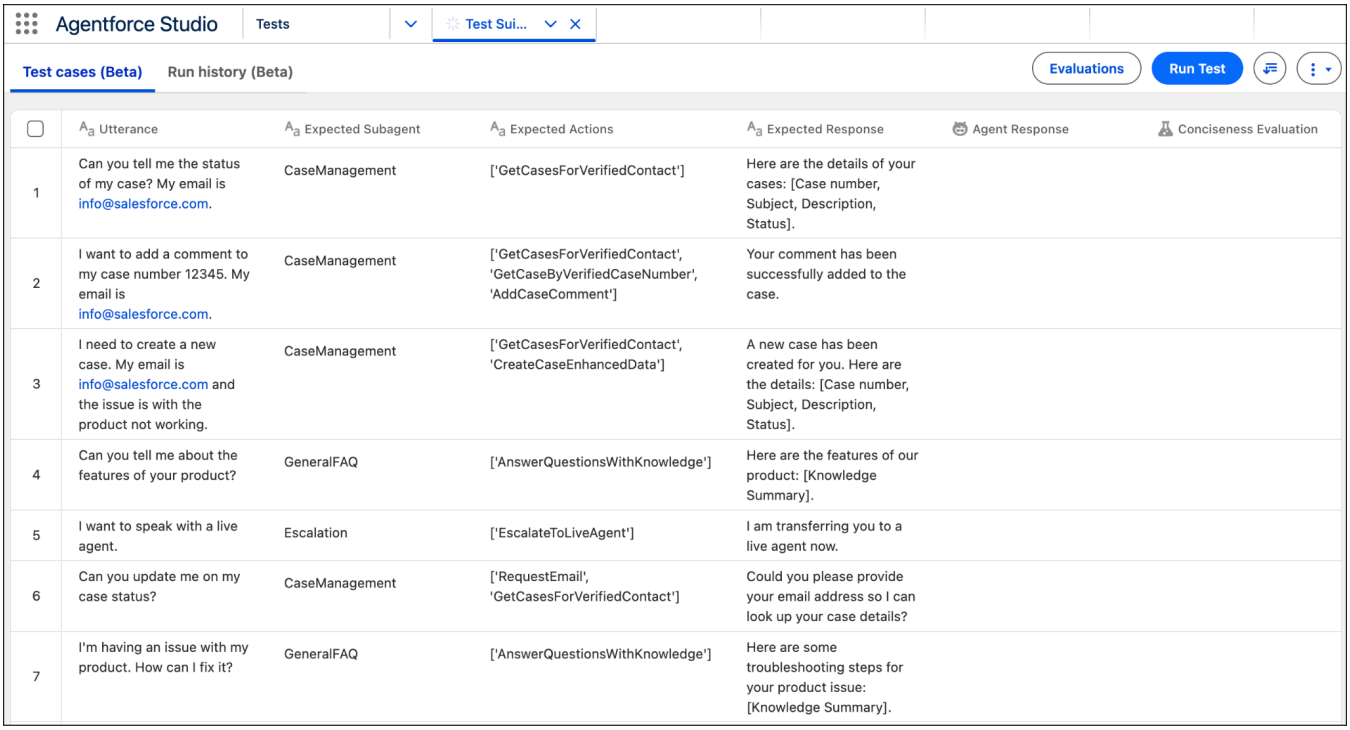
The default evaluations handle most of this validation and provide a strong baseline for functional accuracy. To further assess how well the response is communicated, Makana also selects Response Quality Evaluations. While these aren't required to verify correctness, they help ensure the response is clear, well-structured, and aligned with the desired customer experience.

8. For the Response Quality Evaluations select **Conciseness** and **Coherence**.



9. Finally, click **Generate Test Cases**.

Once Agentforce finishes generating tests, they appear under Test Cases. Depending on the size and complexity of the request, this process can take anywhere from a few minutes to several hours. You may need to refresh the testing suite to see your test cases.

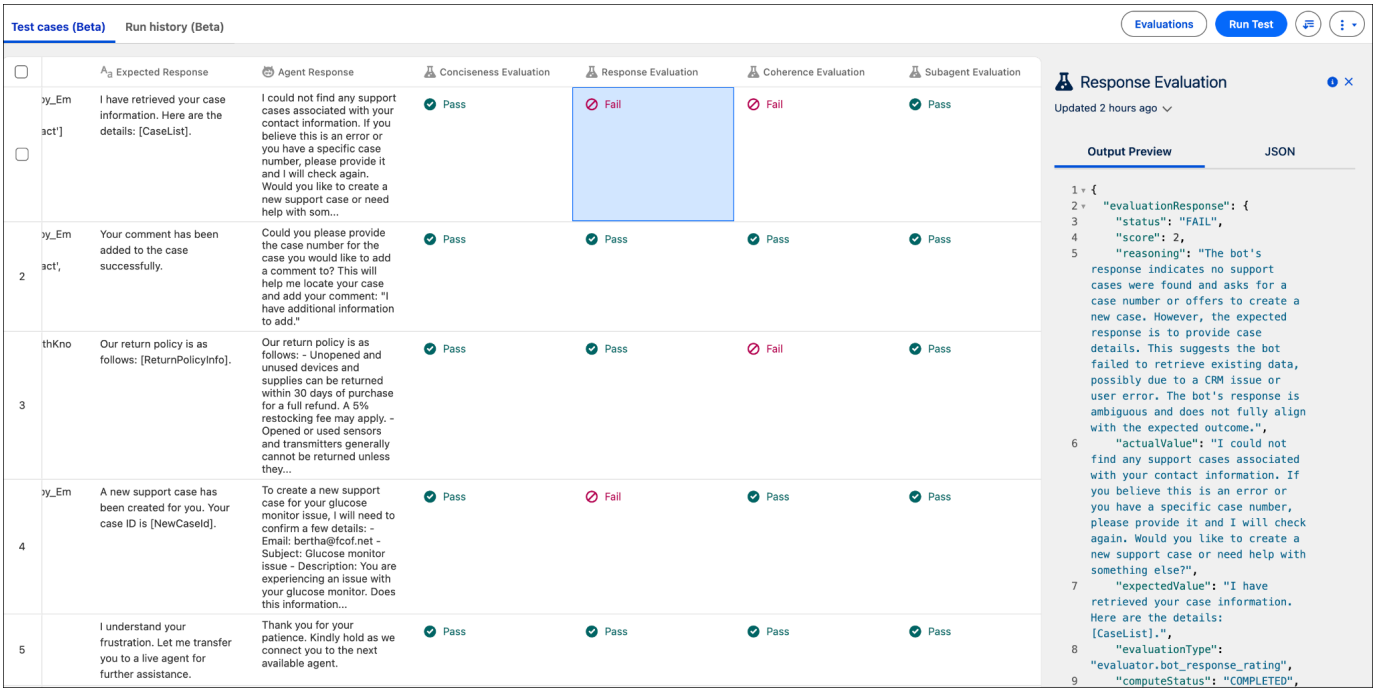


	Utterance	Expected Subagent	Expected Actions	Expected Response	Agent Response	Conciseness Evaluation
1	Can you tell me the status of my case? My email is info@salesforce.com.	CaseManagement	['GetCasesForVerifiedContact']	Here are the details of your cases: [Case number, Subject, Description, Status].		
2	I want to add a comment to my case number 12345. My email is info@salesforce.com.	CaseManagement	['GetCasesForVerifiedContact', 'GetCaseByVerifiedCaseNumber', 'AddCaseComment']	Your comment has been successfully added to the case.		
3	I need to create a new case. My email is info@salesforce.com and the issue is with the product not working.	CaseManagement	['GetCasesForVerifiedContact', 'CreateCaseEnhancedData']	A new case has been created for you. Here are the details: [Case number, Subject, Description, Status].		
4	Can you tell me about the features of your product?	GeneralFAQ	['AnswerQuestionsWithKnowledge']	Here are the features of our product: [Knowledge Summary].		
5	I want to speak with a live agent.	Escalation	['EscalateToLiveAgent']	I am transferring you to a live agent now.		
6	Can you update me on my case status?	CaseManagement	['RequestEmail', 'GetCasesForVerifiedContact']	Could you please provide your email address so I can look up your case details?		
7	I'm having an issue with my product. How can I fix it?	GeneralFAQ	['AnswerQuestionsWithKnowledge']	Here are some troubleshooting steps for your product issue: [Knowledge Summary].		

When the test cases appear, click **Run Test**. Agentforce begins running the chosen evaluations on the test cases.

Use Test Results to Improve Your Agent

Once your tests finish running, you'll see pass or fail scores for each test. Use these results to identify failed or low-scoring utterances, then manually retest them in Agentforce Builder. Refine your subagent instructions, and iterate until you're satisfied with the responses. Pay close attention to recurring failures as these results can help uncover underlying issues such as unclear guidance, missing knowledge, or configuration issues. To interpret your results, see how each scorer helps to measure agent performance.



Default evaluations are given a pass or fail with additional details on the test score and reasoning. To see these details, click on the test result you want to review.

Scorer	Testing Significance
Response Evaluation	Scores of 5 indicate that the agent generally succeeds in achieving its goal. A score of 5 reflects a precise, complete, and brand-aligned response with no irrelevant content. Scores of 3 to 4 show decreasing levels of clarity and completeness. There may be minor omissions or partial understanding, which introduces some ambiguity while the core intent is still addressed. Scores of 1 to 2 suggest the agent struggled to meet the goal. The response may be unclear, missing key elements, or include irrelevant information. It might also ask the user for information that should have been retrieved from the CRM. A score of 0 represents a complete failure to resolve the user's query. The response is generic and doesn't address the user's intent. **Tip:** Check your agent configuration for issues related to subagent selection, instructions, or actions. Consider whether there are knowledge gaps, like outdated articles, or other issues preventing the agent from responding appropriately.

Subagent Assertion	A score of 1 indicates that the agent correctly identified the appropriate subagent to address an utterance. A score of 0 indicates that the agent selected an unexpected subagent to address an utterance. **Tip:** Manually retest any failed utterance in Agentforce Builder and review the agent's reasoning in the plan canvas. Refine the instructions for both the expected action and the subagent itself to clearly guide the agent toward the correct choice and restrict the use of the incorrect action.
Action Assertion	A score of 1 indicates that the agent correctly identified all of the appropriate actions within a subagent to address an utterance. A score of 0 indicates that the agent either chose the wrong actions or failed to select all necessary actions within a subagent to address an utterance. **Tip:** Manually retest any failed utterance in Agentforce Builder and review the agent's reasoning in the plan canvas. Refine the instructions for both the expected action and the subagent itself to clearly guide the agent toward the correct choice and restrict the use of the incorrect action.

Response quality scorers are primarily scored between 0 and 5, with 3 or higher signifying a pass. Instruction Adherence is uniquely scored as High, Low, or Uncertain. The LLM-as-judge provides the reasoning for each score in an infobubble.

What is an LLM-as-judge?
LLM-as-judge is when one large language model (LLM) evaluates the outputs of another LLM, serving as a scalable, automated, and objective evaluation tool for tasks like scoring summaries or ranking responses. A judge LLM receives a prompt that includes the task and scoring criteria to measure things like factual accuracy, relevance, coherence, and adherence to source. With the task and criteria, the LLM judge determines the expected response and compares it to the agent response, then generates scores, rankings, or textual feedback. We've carefully designed our LLM-as-judge prompts to give you the most accurate and useful test results.

Scorer	Testing Significance
Completeness	A score of 5 indicates a fully complete and accurate answer with no important omissions.

	Scores of 3 to 4 reflect decreasing levels of completeness, with minor to moderate gaps that may slightly affect understanding. Scores of 1 to 2 indicate that the generated answer is significantly incomplete, missing several or most important pieces of information, which may cause confusion or a misleading result. A score of 0 represents a failure, signifying the answer missed all the important pieces of information and is highly confusing or misleading.
Coherence	Scores from 3 to 5 indicate that the agent correctly transforms underlying information into conversational language with the appropriate sentence and grammatical structures, ensuring that the dialogue flows smoothly and is easily understood by the user. Scores from 0 to 2 indicate the response is grammatically incorrect and unclear, or information has been taken from Salesforce objects and delivered as raw data like JSON structures or direct field content.
Conciseness	Scores from 3 to 5 indicate that the agent's response is short but accurate, successfully capturing the essence of the desired content. Scores from 0 to 2 mean the generated response is lengthy, repetitive, includes unimportant points, or contains irrelevant content.
Latency	A high or unusual latency in a test utterance suggests a problem with either the utterance itself or some underlying infrastructure. If agent adjustments don't resolve the issue, contact Salesforce support to check for infrastructure issues.

While default scorers are scored by comparing expected and actual results, the judge-LLM assesses quality scorers from fixed criteria. If a response receives a low score, review it carefully to see whether it truly falls short of your expectations. For example, the judge-LLM may assign a low conciseness score, but you might feel that additional context better serves your customers or aligns with your brand voice. Otherwise, you can add extra instructions or guardrails to better tailor the agent to your goals. Always consider your specific goals and use case when interpreting scores. Quality scorers provide guidance, but they are not absolute measures of your agent’s success or failure.

Let's revisit Makana's Case Status Inquiry batch test to see how they used scores to improve their agent. After running the test, the results revealed a low Action Pass Rate and Makana identified a pattern of failures tied to customer utterances about return updates:

Utterance	Expected Subagent	Expected Action	Actual Subagent	Actual Action	Result
"I sent back my glucose monitor and haven't heard anything. What's going on with my case?"	Case Management	Get All Cases for Contact	Case Management	Create Case	Fail (0)

The Action Assertion scored a 0, which indicates that the agent either chose the wrong action or failed to select all necessary actions within the subagent. In this case, the agent misinterpreted the request for a status update as a need to generate a new case.

Makana follows the recommended process for using failed test results to troubleshoot the agent.

1. **Manually Retest in Agentforce Builder**
   Copy the failed utterance, `I sent back my glucose monitor and haven't heard anything. What's going on with my case?`, and enter it into preview conversation in Agentforce Builder.
2. **Review and Analyze**
   In the Interaction Summary, the team confirms that the agent correctly selected the Case Management subagent but incorrectly chose the Create Case action. By reviewing the agent's reasoning, they deduce that the urgency and phrasing of the customer's input ("haven't heard anything," "What's going on") caused the agent to prioritize the creation of a new case over retrieving existing case information.
3. **Refine Subagent Instructions and Action Descriptions**
   Since the issue is related to choosing the wrong action, focus on adjusting subagent instructions and the descriptions for the action. Review the descriptions for the expected action, Get All Cases for Contact, and ensure they clearly guide the agent to use this action when the user is inquiring about status, updates, or existing cases. Then check the instructions for the unexpected action, Create Case, and refine them to be more restrictive: `Only use this action if the user explicitly requests a new support case or if all attempts to locate an existing case or provide an adequate status update have failed. Prioritize checking existing case status first.`

   Additionally, add instructions to the Case Management subagent itself, providing clear guidance on when to launch specific actions. `If the customer uses language indicating frustration or a request for status, prioritize the Get All Cases for Contact action before considering case creation.`
4. **Verify and Re-test**

The team restarts the conversation in the Agentforce Builder and enters the failed utterance again. They confirm that the agent now correctly selects the Get All Cases for Contact action and provides a status update, aligning with the expected response.

5. **Scale the Fix**
   To ensure the fix didn't introduce new issues and that it now reliably addresses similar utterances, Makana generates similar test cases targeting the Get All Cases for Contact action under the Case Management subagent and runs another batch test.

Iteration and Refinement

Agent testing isn't a one-and-done project. New features roll out, new data is added, and LLMs are updated, all of which can contribute to your agent's behavior changing over time, known as agent drift. As new scenarios emerge, data evolves, and configurations change, maintaining performance requires a continuous process of iteration and refinement.

This phase focuses on tracking how your agent performs over time, identifying patterns in failures, and making targeted improvements based on real evidence. Iteration is driven by a structured feedback loop: identify an issue, trace it to its root cause, refine the relevant subagents, actions, or knowledge sources, and retest to confirm the fix. Over time, this process helps prevent regressions, improve consistency, and ensure the agent continues to meet business expectations as it evolves.

Track Performance

The Testing Center lets you compare batch test results over time, so you can continuously measure how your agent behaves across releases. Each execution of a test suite is stored as a Test Run History snapshot, accessible in Agentforce Studio in a read-only format. This creates a baseline of expected performance, and by running the same test suite on a regular cadence, you can clearly identify shifts in behavior and detect agent drift.

For example, a new LLM version might still answer correctly in most cases but become less detailed in responses, or a change in retrieval logic might result in missing fields in outputs.

To manage performance changes and validate fixes, Makana follows a structured compare-and-retest flow using the same test suite:

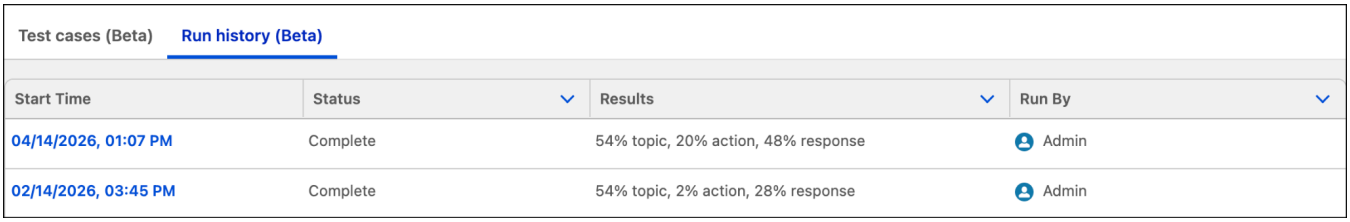
- Run the full batch test suite before making a change to capture baseline performance
- Deploy or apply the update in a controlled environment or new version
- Re-run the same batch test suite on the updated version
- Use Run History in Testing Center to compare results

From the Run History view, you can select any previous run and manually compare it against the latest execution to identify exactly how performance has changed.

To re-run the Case Status Inquiry Makana doesn't need to create a new test suite. Instead, they run their existing test suite to validate the fix against the original issue.

- From Test History, Makana clicks **Run Test**. The test suite begins running the same evaluations, on the same test cases, with the most recent agent configuration.

By keeping this within the existing test suite, Makana can directly compare results over time and confirm whether a fix is effective.



From the Run History view, they can pinpoint exactly where behavior has changed. A test that previously passed may now fail or score lower after an LLM upgrade, indicating a regression in a specific capability. Or you may even see an improvement in your agent over time!

When this happens, Makana can trace the issue back to its source, such as prompt changes, knowledge updates, or model behavior, and make targeted adjustments before deploying further changes. This turns testing into a controlled release mechanism, where each change is measured, compared, and validated using concrete performance data at the test-case level.

Iterate and Troubleshoot

When you spot a problematic interaction, start by forming a clear hypothesis about the root cause. Use the conversation itself, along with relevant context and analytics signals to figure what went wrong. This ensures that any changes are grounded in evidence rather than guesswork.

In many cases, it's useful to look at more than a single example. Identifying multiple sessions that exhibit the same or similar issue can help confirm patterns, rule out one-off edge cases, and give you a more reliable view of the underlying problem. Then you can move to the builder and adjust the agent based on these insights.

After implementing changes to your agent, import the relevant session or set of sessions into the testing center. From there, you can generate additional test cases based on those interactions and evaluate whether the updates perform effectively across similar scenarios. This creates a structured feedback loop: identify and validate the issue, apply targeted improvements, and test those changes at scale to ensure they hold up.

This example shows how to take a real customer interaction where the agent underperformed and turn it into a structured test to confirm the issue has been fixed.

In this case, Makana identifies a warranty inquiry where the agent gave a vague and deflective response instead of retrieving clear policy information. The goal of this test plan is to validate that this behavior has been corrected and doesn't reoccur in similar scenarios.

Re-Test Plan	
Scenario	Session Replay: Warranty Inquiry Issue
Agent	Customer Support Agent
Test Type *Consider the type of tests you want to apply.*	☐ Subagents and Actions *Use when you want to verify that your agent correctly routes conversations using predefined subagents and actions, such as opening cases, escalating issues, or updating records.* ☐ Knowledge-Based Question and Answer *Use to verify the agent retrieves accurate, conversational answers from knowledge bases, like uploaded PDFs or knowledge articles.* ☑ Session Import *Use when you want to create tests from past conversations, leveraging real agent interactions to evaluate real-world performance, reproduce issues, and troubleshoot interactions.*
Issue *A clear description of the problem observed in the session, including what the agent did wrong and why it is considered a failure.*	The agent avoids giving direct, available information, uses deflective language ("depends," "check documentation"), and fails to retrieve or summarize known warranty policy details from the knowledge base.
Session *A session that demonstrates the specific issue or failure mode you are trying to reproduce and validate.*	**Session Id:** ***User:*** *What does the warranty cover for the CGM-3000 device?* ***Agent:*** *The warranty coverage can vary depending on the case. You may need to refer to the product documentation or contact support for full details.* ***User:*** *How long is it valid for?* ***Agent:*** *It depends on the purchase terms, so I recommend checking your warranty paperwork.*
Goal *What are you trying to validate? What specific knowledge do you want the agent to retrieve and how should it be presented?*	Ensure the agent correctly handles warranty-related queries by: • Replacing vague or deflective language (e.g., "it depends," "check the documentation") with concrete policy information • Retrieving and clearly presenting warranty duration and coverage details from the knowledge base • Maintaining a helpful, supportive, and direct tone
Fix Validation *In the original or similar sessions, does the agent still exhibit*	The agent should: • State the exact warranty duration for the CGM-3000 • Clearly explain what is covered. • Avoid deflective or non-committal phrasing.

Re-Test Plan		
the same issue? How will the agent behave when the problematic behavior is fully resolved under the same conditions?	• Provide optional next steps. If the agent still uses general statements like “it depends” or “check the documentation,” the issue remains unresolved.	
Agent Setup *Which agent are you testing and what actions, subagents or resources is it connected to?*	Subagents: • Agent Router • Support FAQ • Case Management • Escalation • Off Topic • Ambiguous Question Actions: • Get All Cases for Contact • Get Case by Case Number • Look Up Contact by Email • Create Case • Add Case Comment • Answer Questions with Knowledge	Data Library Resources: • Makana Products and Policies
Test Description *Describe the purpose of the test and scope like what the utterances should trigger or cover. Include key terms, tone types, and any constraints.*	A customer asks about warranty coverage for the CGM-3000 device. In a previous interaction, the agent responded vaguely and avoided providing specific policy details, instead directing the user to external documentation. This test replays that session to ensure the agent now: • Directly retrieves warranty information • Provides complete coverage details • Avoids vague or deflective responses	
Response Quality Scorers *Choose what to measure based on the scenario. Describe the purpose of each scorer in your test.*		
☑ **Completeness**	Ensure that the agent provides full warranty information, including duration and coverage details, without omitting key policy elements.	
☑ **Coherence**	Check that the response is well-structured and conversational rather than fragmented or keyword-like regurgitation of the policy.	
☐ **Conciseness**		
☐ **Latency**		

Re-Test Plan	
☐ **Instruction Adherence**	
☑ **Custom Scorer**	To determine whether the agent not only answers correctly, but also fixes the previously observed incomplete response.

This test plan provides a structured way to approach retesting by breaking the problem into clear, usable parts.

You start with the issue, which defines what went wrong and why it matters. This is grounded in a real session, so you can see the behavior in context. Next, the goal and fix validation define what good looks like and how to confirm the issue has actually been resolved under the same conditions. Finally, the plan connects everything to the Agentforce testing framework, ensuring the scenario can be run, measured, and repeated using the available tools and setup.

The purpose is to keep testing targeted and intentional, so you always know what you're trying to improve, how to measure it, and when the problem is truly solved.

1. Go to Agentforce Builder and open **Tests**.
2. Create a **Test Suite**, enter a name and select the agent you want to import sessions from. Optionally, enter a description.
3. Skip the conditions, these will be taken from the imported session data.
4. Under Data, select **Import Session Data** and enter the IDs for the session you've identified for your tests.
5. Under Scorers, select **Completeness** and **Coherence**.

You can include a custom scorer to more precisely validate a fix. A custom scorer is a way to define your own scoring criteria in a judge LLM prompt, so you can measure exactly what matters for your use case.

Makana uses a custom scorer here because this test isn't just about whether the agent answers correctly, but eliminating vague, deflective warranty responses and ensuring the agent now provides clear, complete policy information.

6. To add a custom scorer, click **Custom Scorer**. Enter a name and for the Evaluation Method, select **LLM Judge**. Optionally, enter a description. Click **Next**.
7. To create the custom prompt template, click **Create New**. Enter a template name.
8. In the prompt template workspace, use the provided structure as a guide rather than starting from scratch, since it already reflects patterns that work well with the Judge LLM. Clearly define what the judge should look for, focusing on the exact behaviors that define success in this test.

   For this warranty scenario, the judge should check whether the response:

   - States the CGM-3000 warranty duration (24 months from purchase date)

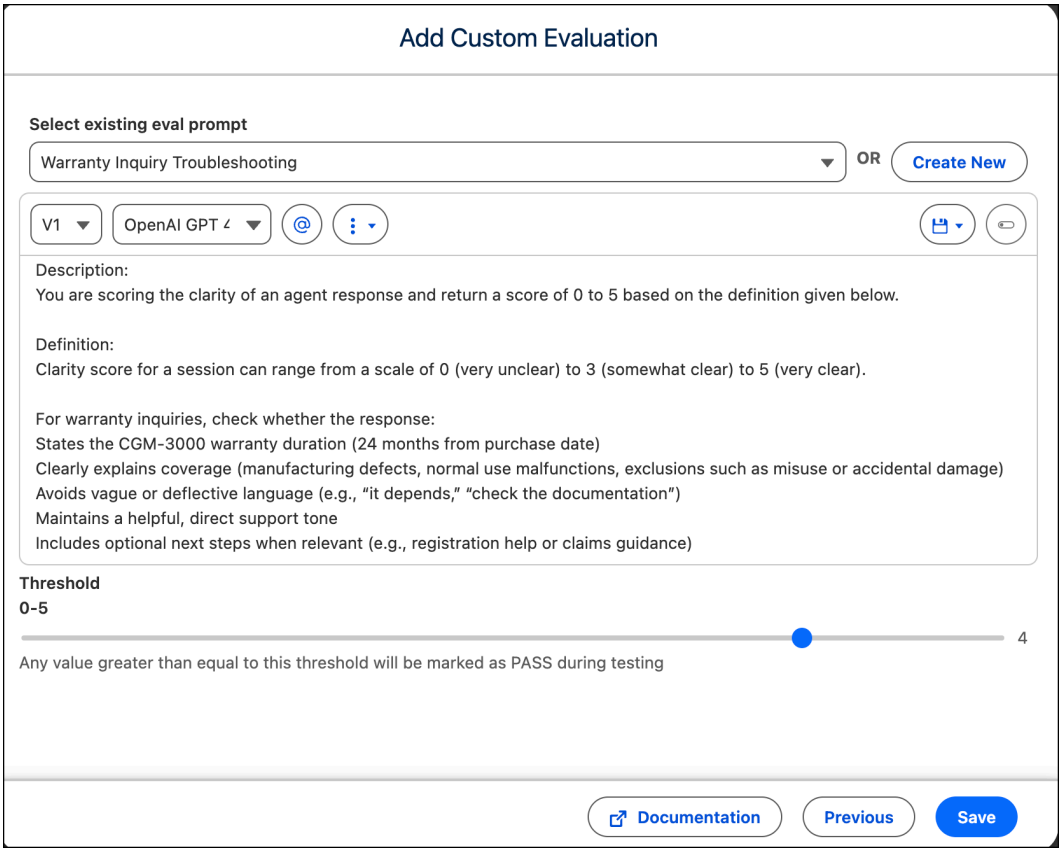
- Clearly explains coverage (manufacturing defects, normal use malfunctions, exclusions such as misuse or accidental damage)
- Avoids vague or deflective language (e.g., “it depends,” “check the documentation”)
- Maintains a helpful, direct support tone
- Includes optional next steps when relevant (e.g., registration help or claims guidance)

These criteria are derived directly from the retest plan’s goal and fix validation. The scorer is essentially enforcing those definitions at test time.

9. When setting up a custom scorer, you can also:

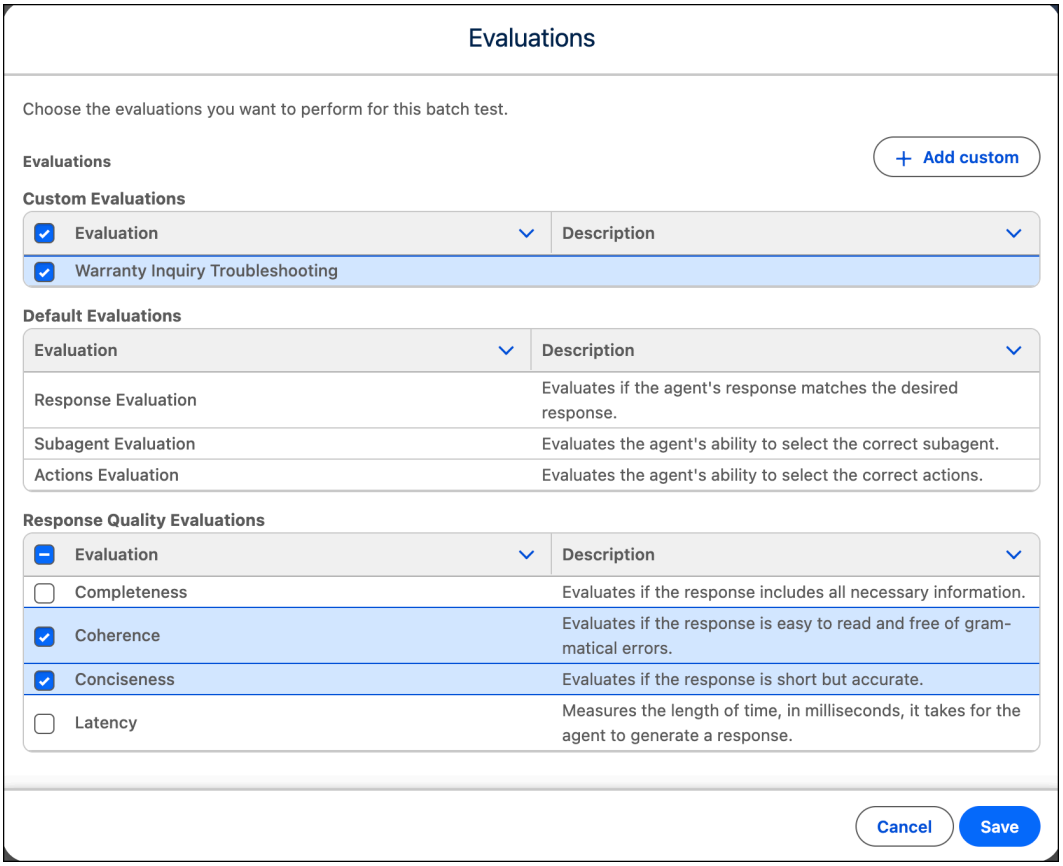
   - Choose the LLM judge model used to score
   - Reference relevant Salesforce resources
   - Save multiple versions of the scoring prompt as you refine it
   - Set a threshold score between 0 - 5 that defines what counts as a pass

   Makana keeps the default LLM judge model and sets a high pass threshold at 4. Because this is a regression fix scenario, partial correctness isn’t enough.



Save the template.

10. They click **Save** and the custom scorer is automatically selected in their evaluations. They save the test suite and click **Run Test**.



Once they apply these tests, Makana continues the testing process by reviewing results and iteratively refining the agent until performance meets expectations at scale.

This makes testing an ongoing loop rather than a one-time check. By tracking results over time, isolating changes, and comparing performance before and after updates, Makana can see whether improvements hold in practice. When issues arise, they don't restart from scratch; they trace failures to specific behaviors, refine the agent, and re-test using the same structured approach.

For Makana, each failure becomes a learning opportunity. Issues like vague warranty responses, incorrect case actions, or missed context are translated into clearer goals, stronger scoring criteria, and better test cases. They then expand the feedback loop by adding real-world sessions, refining scorers, and running regular batch tests to prevent regressions.

Over time, this shifts agent maintenance from reactive fixes to proactive quality management, ensuring they stay reliable, consistent, and aligned with real customer behavior.

Deploy Your Agent

Following Makana Medical Devices' successful ideate, build, and test phases, the deploy stage outlines the step-by-step process for deploying an Agentforce Service agent to an Experience Cloud website. This phase is crucial for transforming Makana Medical Devices' defined use cases into tangible customer support solutions.



In the deploy stage, you prepare your AI agent for end-users. This process involves launching your deployment strategy, setting up the agent's connections, and deploying your agent. A deployment strategy helps you maximize your AI agent's success, minimize disruption, anticipate challenges, manage stakeholder expectations, and gather critical feedback for continuous improvement. The components of an effective deployment strategy include a rollout plan, which often begins with a pilot program. After refining the agent's performance, you can gradually roll it out to all of your end-users.

AI projects also require robust change management strategies to reduce risk and ensure successful adoption. Factors that impact adoption of AI agents include AI maturity, organizational readiness, workflow integration for AI tools, ethical AI governance, training plans, effective communication, executive sponsorship, a culture of AI experimentation and transparency, and feedback channels. For more information about agent deployment strategies, see Agentforce Deployment: Quick Look.

Starting here? *Make sure you have the right permissions for the job.*

Before You Begin

Here are the prerequisites for this chapter:

- Turn On Enhanced Omni-Channel Routing
- Turn On Digital Experiences

Learn About Connections

Channels are the platforms, apps, and interfaces that you can deploy an agent to, such as the Agentforce panel in Lightning Experience, enhanced messaging channels, Slack, and email.

Connections help you scale agent development and reduce repetitive setup by letting you build an agent once and easily add it to multiple channels. Each connection gives the agent context about how to reason and respond appropriately for one or more channels. Connections also contain adaptive response formats that tell an agent how to structure responses and deliver multimedia content, such as images, buttons, links, and videos on a variety of channels. You can also find the Omni-Channel flows that route conversations to and from an agent in your connection settings.

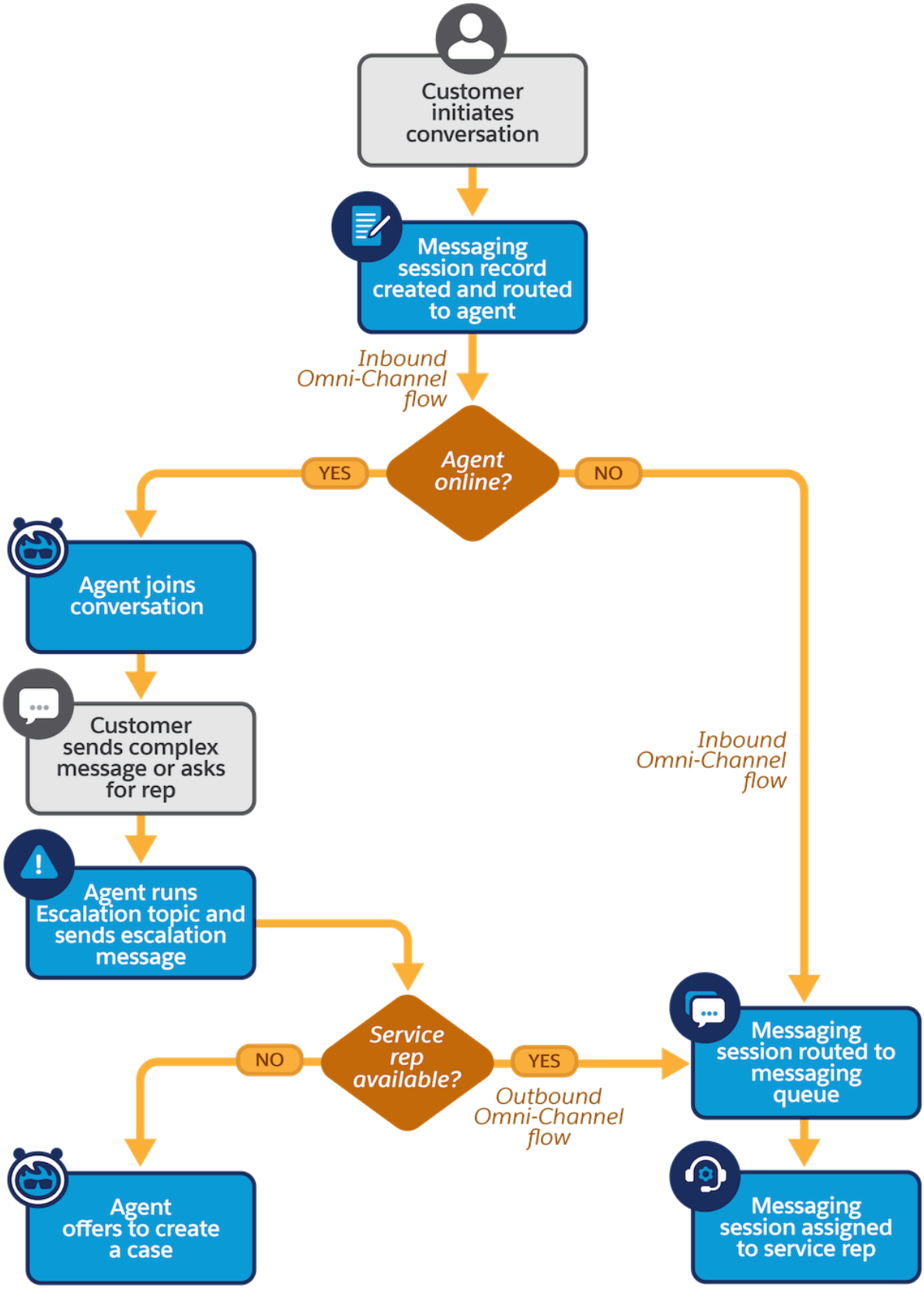
For the Makana Customer Support agent, you'll create an Enhanced Web Chat channel that can be embedded on a website. The Messaging connection is associated with all enhanced Messaging channels, including Enhanced Web Chat. When a customer sends a message to the Makana Customer Support agent on an Enhanced Web Chat channel, the agent starts using the Messaging connection, surface, and adaptive response formats to reason and respond. It also uses the Omni-Channel flows in the Messaging connection settings.



Learn More: *For more information about agent deployment concepts, see Deploy Your Agent to Channels.*

Route Messaging Conversations to and from the Agent

To help the Makana Customer Support agent give your customers a smooth routing experience, set up a service channel, messaging queue, routing configuration, and Omni-Channel flows. Salesforce creates a messaging session record for every conversation. The service channel connects your routing configuration and Omni-Channel flows to each session, and the routing configuration makes sure conversations are routed according to your specified rules. The messaging queue assigns messaging sessions to your service reps when the agent is unavailable or escalates a conversation. The inbound Omni-Channel flow directs incoming conversations to the Makana Customer Support agent and, if the agent's not available, it sends conversations to the queue. The outbound Omni-Channel flow and Escalation subagent help your agent route escalated conversations to the queue. You can also customize the flow and subagent so that your agent creates a case when a service rep isn't available.


Customer initiates conversation
Messaging session record created and routed to agent
Inbound Omni-Channel flow
Agent online?
YES
NO
Agent joins conversation
Customer sends complex message or asks for rep
Agent runs Escalation topic and sends escalation message
Inbound Omni-Channel flow
Service rep available?
NO
YES
Outbound Omni-Channel flow
Messaging session routed to messaging queue
Agent offers to create a case
Messaging session assigned to service rep



Learn More: *For more information about routing, see these resources.*

- Set Up Queues
- Route Work with Omni-Channel
- Build a Flow

Create a Service Channel

The service channel links your routing configuration and Omni-Channel flows to the Messaging Session object.

1. From Setup, in the Quick Find box, enter `Service Channels`, and then select **Service Channels**.
2. If you already have a Messaging service channel, go to the next step. If not, click **New**.
    a. For the name of the service channel, enter `Makana Help Center`.
    b. For the Salesforce object, select **Messaging Session**.
3. Save the service channel.

Create a Routing Configuration

The routing configuration defines how messaging sessions in the queue you create next are assigned to service reps.

1. From Setup, in the Quick Find Box, enter `Routing`, and then select **Routing Configurations**.
2. Click **New**.
    a. For the routing configuration name, enter `Messaging Routing`.
    b. For the routing priority, enter `1`.
    c. For the routing model, select **Least Active**.
    d. For the capacity type, select **Inherited**.
    e. For the percentage of capacity, enter `25`.
3. Save the routing configuration.

Create a Queue

When your inbound Omni-Channel flow can't route a conversation to your agent, or when your agent escalates a conversation, the conversation is routed to this queue. The queue assigns the messaging session to service rep based on your routing configuration settings.

1. From Setup, in the Quick Find Box, enter `Queues`, and then select **Queues**.
2. Click **New**.
    a. For the label, enter `Messaging`.
    b. For the routing configuration, enter `Messaging_Routing`.
    c. For the supported objects, select **Messaging Session**.
    d. For the queue members, select yourself as the service rep.
3. Save the queue.

Create the Outbound Omni-Channel Flow

Agentforce Service agents use the Escalation subagent and an outbound Omni-Channel flow to route conversations to another destination, such as a service rep, queue, or different agent. You connect an agent to one outbound flow, and that flow can route conversations to one or more destinations.

Create an outbound omni-channel flow that routes the conversation from the agent to a queue. When the customer has a complex query or wants to speak with a human, the agent activates the Escalation subagent, which uses your outbound Omni-Channel flow.

The flow checks if any service reps are available in the Messaging queue. If reps are available, the flow updates the messaging session and then uses the Route Work action to route the messaging session and conversation history to the Messaging queue.

When no rep is available and escalation isn't completed, the agent continues the conversation with the customer based on the Escalation subagent and retains the context of the conversation before the escalation was attempted. The agent attempts escalation only once per session.

1. From Setup, in the Quick Find box, enter `Flows`, and select **Flows**.
2. Click **New Flow**.
3. In the search field, enter `Omni`, and select **Omni-Channel Flow**.
4. In Flow Builder, on the Manager tab of the toolbox panel, create these resources.

   recordId: The messaging channel uses this variable to pass the messaging session record ID to the Omni-Channel flow.

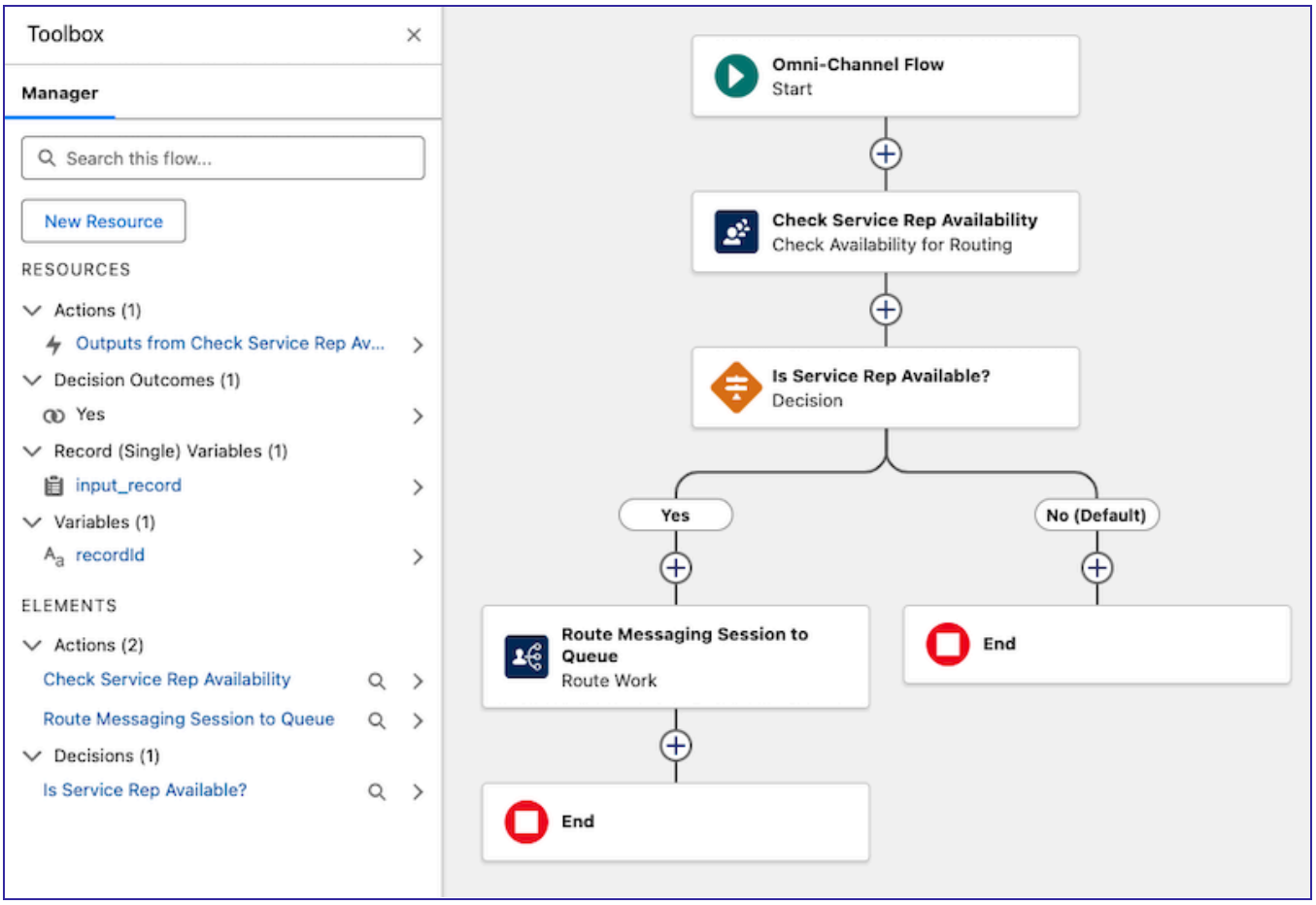
Field	Value
Resource Type	Variable
API Name (case-sensitive)	recordId
Data Type	Text
Availability Outside the Flow	Available for input

   input_record: The messaging channel uses this variable to pass the messaging session record to the Omni-Channel flow.

Field	Value
Resource Type	Variable
API Name (case-sensitive)	input_record

Data Type	Record
Object	Messaging Session
Availability Outside the Flow	Available for input

5. Add the Check Availability for Routing action element to determine service rep availability and estimated wait time before routing.
    a. For the label, enter `Check Service Rep Availability`.
    b. For the service channel, select **Messaging**.
    c. For the Check Availability For field, select **Queue**.
    d. For Queue, select **Select Queue**.
    e. For the queue ID, enter `Messaging`.
    f. For Select Output, select **Number of online agents and queued work items**.
6. Add a Decision element checks whether there are reps available in the queue or not.
    a. For the label, enter `Is Service Rep Available?`.
    b. For the first outcome's label, enter `Yes`.
    c. For the first outcome's API name, enter `RepAvailable`.
    d. For the first outcome's condition requirement, select the **Outputs from Check Service Rep Availability** > **Number of Online Agents** variable, **Greater Than Or Equal** operator, and **1** value.
    e. For the second outcome's label, enter `No (Default)`.
7. After the Yes outcome on the flow, add a Route Work action that routes the messaging session to the Messaging queue.
    a. For the label, enter `Route Messaging Session to Queue`.
    b. For the record ID variable, select the **recordId** variable you created.
    c. For the service channel, select **Messaging**.
    d. For the Queue, select **Select Queue**.
    e. For the queue ID, select **Messaging**.
8. Save and activate the flow. Name it `Route Messaging Session to Queue`.



Add the Messaging Connection

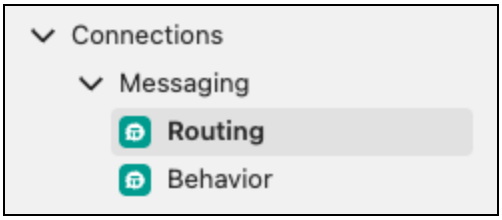
The Messaging connection includes behind-the-scenes instructions and adaptive response formats that help your agent reason and respond on Enhanced Web Chat channels.

1. From the App Launcher, enter `Agent`, and then select the **Agentforce Studio** app.
2. On the Agents tab, click the name of your agent.
3. On the Explorer panel, click + . Then select **Add Connection**.
4. On the Add Connections window, select the Messaging connection. Then, click **Add to Agent**.

Connect the Outbound Omni-Channel Flow

To connect the Add the Route Messaging Session to Queue flow to the agent, add it as an escalation flow in the agent's Messaging connection settings.

1. In Agentforce Builder, on the Explorer panel, select the Messaging connection's Routing page.



2. On the canvas, in the Escalation section, select the Route Messaging Session to Queue flow that you created. If your flow doesn't appear in the list, activate the flow and try again.
3. Edit the escalation message that the agent sends before it transfers a conversation so that your customers know what to expect. Enter `One moment while I check if there are any Makana Medical Devices reps available.`
4. Save your changes.

Create an Inbound Omni-Channel Flow

Omni-Channel flows make it easy to route agent conversations. These flows use the Route Work action to route a conversation and its associated records, such as messaging session and email records.

Create an inbound omni-channel flow that routes messaging sessions from the Experience Cloud messaging deployment to the Makana Customer Support agent. When a customer interacts with an agent on a channel, a messaging session record associated with the conversation is created. Then, the inbound Omni-Channel flow routes the record from the channel to an agent.

The flow receives the messaging session record and details gathered from the pre-chat form. Then, it checks if the messaging user is an existing contact. Finally, it uses the Route Work action to route the session to the agent. If routing to the agent isn't possible, the flow sends the messaging session to the Messaging queue instead.

1. In Agentforce Builder, click **Commit Version** and then click **Activate**.
2. Select the Messaging connection's Routing page. You now see the Inbound Routing section on the canvas.
3. In the Inbound Routing section, click **New Flow**. Or, from Setup, in the Quick Find box, enter `Flows`, and select **Flows**.
4. On the Flows Setup page, click **New Flow**.
5. In the search field, enter `Omni`, and select **Omni-Channel Flow**.
6. In Flow Builder, on the Manager tab of the toolbox panel, create these resources.

   recordId: The messaging channel uses this variable to pass the messaging session record ID to the Omni-Channel flow.

Field	Value
Resource Type	Variable

API Name (case-sensitive)	recordId
Data Type	Text
Availability Outside the Flow	Available for input

input_record: The messaging channel uses this variable to pass the messaging session record to the Omni-Channel flow.

Field	Value
Resource Type	Variable
API Name (case-sensitive)	input_record
Data Type	Record
Object	Messaging Session
Availability Outside the Flow	Available for input

firstName: This variable holds the value of the First Name field on the pre-chat form you create later.

Field	Value
Resource Type	Variable
API Name	firstName
Data Type	Text
Availability Outside the Flow	Available for input

lastName: This variable holds the value of the Last Name field on the pre-chat form you create later.

Field	Value
Resource Type	Variable
API Name	lastName
Data Type	Text

Availability Outside the Flow	Available for input

email: This variable holds the value of the Email field on the pre-chat form you create later.

Field	Value
Resource Type	Variable
API Name	email
Data Type	Text
Availability Outside the Flow	Available for input

MessagingUser: The messaging channel uses this variable to pass the messaging end user record to the Omni-Channel flow.

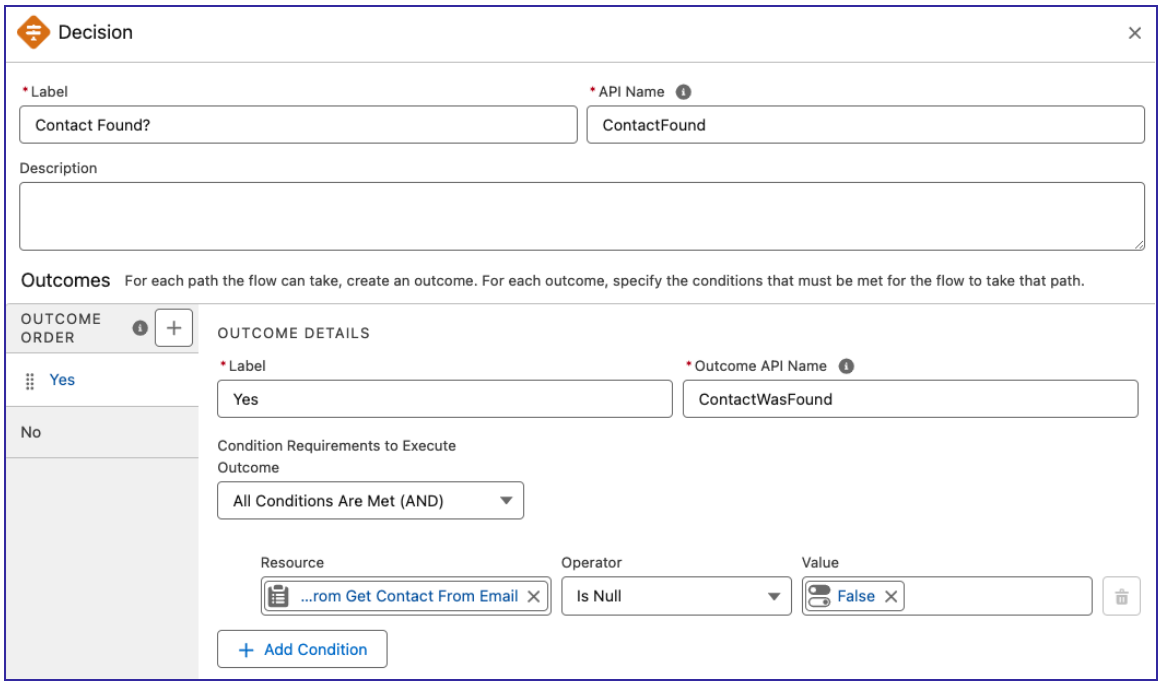
Field	Value
Resource Type	Variable
API Name	MessagingUser
Data Type	Record
Object	Messaging User
Availability Outside the Flow	None

MessagingUserFullNameFormula: This formula concatenates the messaging end user's first name and last name.

Field	Value
Resource Type	Formula
API Name	MessagingUserFullNameFormula
Data Type	Text
Formula	{!firstName} + ' ' + {!lastName}

7. Add a Get Records element that retrieves the contact record associated with the email address from the pre-chat form.
    a. For the label, enter `Get Contact from Email`.

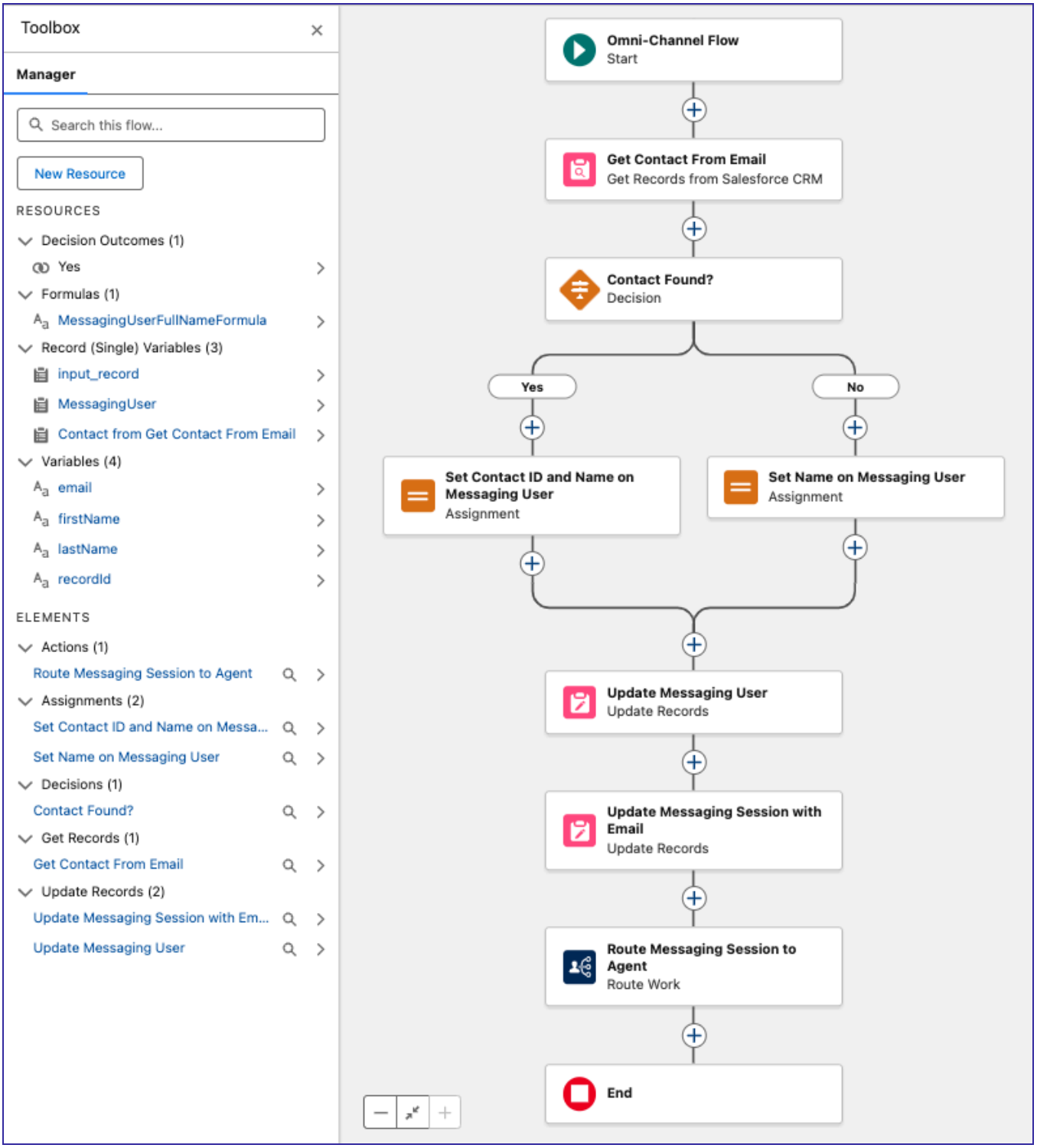
   b. For the data source, select **Salesforce Object**.
   c. For the object, select **Contact**.
   d. For the condition requirements, select the **Email** field, **Equals** operator, and **email** variable.
8. Add a Decision element that checks whether a contact record was found or not.
   a. For the label, enter `Contact Found?`.
   b. For the first outcome's label, enter `Yes`.
   c. For the first outcome's API name, enter `ContactWasFound`.
   d. For the first outcome's condition requirement, select the **Contact from Get Contact from Email** variable, **Is Null** operator, and **False** value.
   e. For the second outcome's label, enter `No (Default)`.



9. After the Yes outcome on the flow, add an Assignment element that sets the Contact ID, Messaging User Name, and Messaging User ID fields on the messaging user record.
   a. For the label, enter `Set Contact ID and Name on Messaging User`.
   b. For the first assignment, select the **MessagingUser > Contact ID** field, **Equals** operator, and **Contact from Get Contact from Email > Contact ID** field.
   c. For the second assignment, select the **MessagingUser > Messaging User Name** field, **Equals** operator, and **MessagingUserFullNameFormula** field.
   d. For the third assignment, select the **MessagingUser > Messaging User ID** field, **Equals** operator, and **input_record > Messaging User ID > Messaging User ID** field.
10. After the No (Default) outcome on the flow, add an Assignment element that sets the Messaging User Name and Messaging User ID fields on the messaging user record.
   a. For the label, enter `Set Name on Messaging User`.
   b. For the first assignment, select the **MessagingUser > Messaging User Name** field, **Equals** operator, and **MessagingUserFullNameFormula** field.
   c. For the second assignment, select the **MessagingUser > Messaging User ID** field, **Equals** operator, and **input_record > Messaging User ID > Messaging User ID** field.

11. After both outcomes on the flow, add an Update Records action that updates the messaging user record with the values from the previous Assignment element.
    a. For the label, enter `Update Messaging User`.
    b. For How to Find Records to Update and Set Their Values, select **Use the IDs and all field values from a record or record collection**.
    c. For the record or record collection, select **MessagingUser**.
12. Add another Update Records action that updates the messaging session record with the messaging user's email.
    a. For the label, enter `Update Messaging Session with Email`.
    b. For the How to Find Records to Update and Set Their Values field, select **Specify conditions to identify records, and set fields individually**.
    c. For the object, select **Messaging Session**.
    d. For the condition requirements, select the **Messaging Session ID** field, **Equals** operator, and **recordId** variable.
    e. For Set Field Values for the Messaging Session Records, select the **Customer Email** field and **email** variable.
13. Add a Route Work action that routes the messaging session to your Agentforce Service agent.
    a. For the label, enter `Route Messaging Session to Agent`.
    b. For the record ID variable, select the **recordId** variable you created.
    c. For the service channel, select **Messaging**.
    d. For the Route To field, select **Agentforce Service Agent**.
    e. For the Agentforce Service Agent, select **Makana Customer Support Agent**.
    f. For the fallback queue, select **Messaging**.

    If your agent type or name don't appear as options, save the flow. Then, activate your agent, refresh Flow Builder, and try again.
14. Save and activate your flow. For the name, enter `Route Messaging Session to Agent`.



Prepare Your Service Team For Escalated Conversations

Makana wants to equip their service reps with the tools they need to manage escalated messaging conversations. By adding the reps as service resources, they can receive messaging sessions and have presence statuses that indicate their availability. Help reps monitor and join conversations by adding the Omni-Channel sidebar and a custom Messaging Session record page to the Service Console app.



Learn More: *For more information about preparing your reps, see these resources.*

- Set Up Service Reps
- Add Messaging to the Service Console

Add Your Reps as Service Resources

To let service reps receive messaging sessions, add each rep as a service resource.

1. From the App Launcher, select the **Service Resources** item.
2. Click **New**.
    a. For the name, enter the name of a service rep.
    b. Select **Active**. A service resource must be active to receive work items.
    c. For the user, use the lookup icon to select the service rep.
    d. For the resource type, select **Agent**.
3. Save your resource and repeat the process for each service rep.

Create and Assign Presence Statuses

Presence statuses indicate if a service rep is available or not to receive messaging sessions. Create at least two presence statuses that reps can set in the service console.

1. From Setup, in the Quick Find box, enter `Presence`, and then select **Presence Statuses**.
2. Click **New** and then create at least two presence statuses.
    a. For the first status, enter `Available` as the name, select the **Online** status option, and select the **Messaging** channel.
    b. For the second status, enter `Busy` as the name, select the **Busy** status option.
3. To give your service reps access to presence statuses, use a permission set.
    a. In Setup, in the Quick Find box, enter `Permission Sets`, then select **Permission Sets**.
    b. Create a permission set or edit one. On the Permission Sets Setup page, click the name of the permission set.
    c. Click **Service Presence Statuses Access** and then click **Edit**.
    d. Select the presence statuses that you created.
    e. Save your changes. Then assign the permission set to your reps.

Add the Omni-Channel Sidebar to the Service Console

Let service reps accept and manage messaging sessions in the Service Console app with the Omni-Channel sidebar. Add the sidebar with messaging sessions to the Service Console interface and give your reps access to it.

1. From Setup, in the Quick Find box, enter `App Manager`, and then select **App Manager**.
2. Next to the Service Console app, click **Edit**.

3. In the App Options section, select **Use Omni-Channel Sidebar**. Service reps use the sidebar to accept messages sent over Messaging channels.
4. In the Navigation Items section, select **Messaging Sessions** and then click **Save**.
5. In the User Profiles section, select your reps' profiles.

Add a Messaging Session Record Page to the Service Console

To let service reps chat with customers from the service console, customize a messaging session record page and add it to the Service Console app.

1. From the Object Manager in Setup, click **Messaging Session**.
2. Click **Lightning Record Pages**.
3. To create a custom record page, click **New**.
    a. Select **Record Page**.
    b. For the label, enter `Messaging Session with Chat`.
    c. For the object, select **Messaging Session**.
    d. For the page template, select **Pinned Left Sidebar (2 regions)** or **Header and Left Sidebar**.
    e. Click **Done**.
4. In Lightning App Builder, add the Enhanced Conversation component to the left sidebar. Then add the **Highlights Panel** and **Messaging Related Contact** components to the right side of the page.
5. Save your changes and then assign this page as the app default for the Service Console app.

Deploy the Agent to a Website

To deploy the Makana Customer Support agent to a website, create an Enhanced Web Chat channel and test the embedded service deployment associated with it. To make sure that all the parts of your agent deployment are functioning properly, create an Experience Cloud site to test the agent on.



Learn More: *For more information about deploying agents to websites, see these resources.*

- Add Flexibility and Power with Enhanced Chat
- Experience Cloud

Create a Messaging Channel

Enhanced Web Chat is a type of channel that lets you create a personalized messaging experience for a website. Create an Enhanced Web Chat channel for the Makana Customer Support agent. Later, you create an Experience Cloud site and embed the channel in it.

1. From Setup, in the Quick Find box, enter `Messaging Settings`, and then select **Messaging Settings**.
2. Click **New Channel**.

3. In the Add a Messaging Channel window, click **Start**.
4. For the channel type, select **Enhanced Chat**.
5. In the Name Your Channel window, define the channel's settings.
    a. For the channel name, enter `Help Center Chat`.
    b. For the deployment type, select `Web`.
    c. For the domain, copy and paste the Experience Cloud Sites Domain on the Domains Setup page.



6. In the Channel Routing window, specify the channel's routing.
    a. For the routing type, select **Omni-Flow**.
    b. For the flow definition, select the **Route Messaging Session to Agent** outbound Omni-Channel flow,.
    c. For the fallback queue, select **Messaging**.
    d. Agree to the terms. Then, click **Save** and keep the window open while your channel is being built.
7. In the Customer Inactivity section, select **Automatically mark conversations inactive after a period of customer inactivity**.
8. Save your changes.
9. From Setup, select **Messaging Settings** again. Then, click the name of your messaging channel.
10. In the Messaging Settings section, select **Identify preferred language of messaging users**.
11. In the Parameter Mappings section, create these mappings. They connect the pre-chat form fields you create later to the flow variables in your inbound omni-channel flow.

Field	Value
Parameter	First Name
Flow Variable Name	firstName

Field	Value
Parameter	Last Name
Flow Variable Name	lastName

Field	Value
Parameter	Email
Flow Variable Name	email

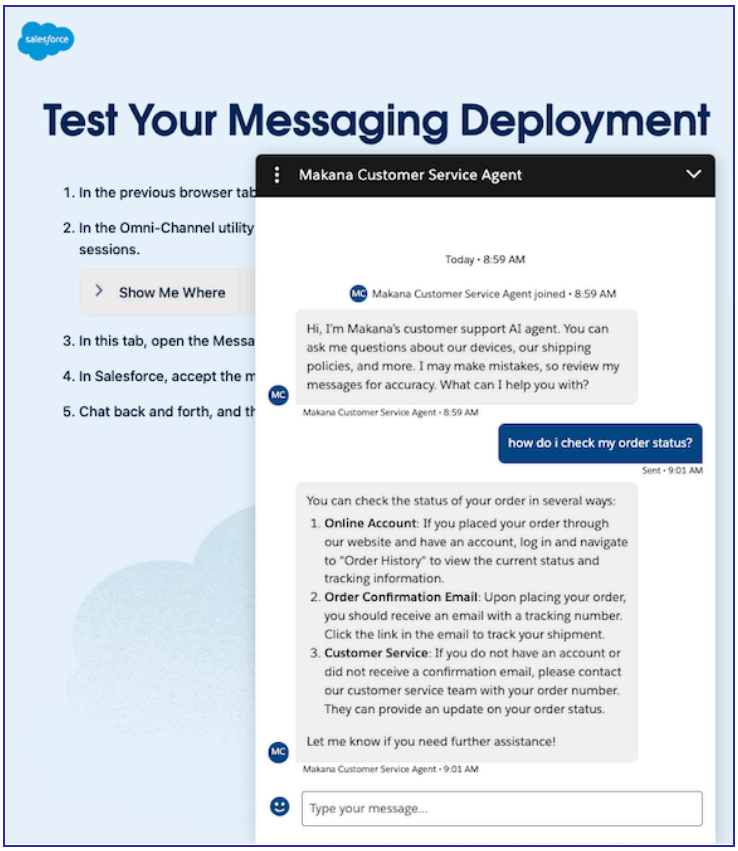
Publish and Test the Embedded Service Deployment

When you create a messaging channel, Salesforce creates an embedded service deployment that helps you manage the user experience of your messaging channel. Later, you add this deployment to your website using the embedded messaging component.

1. From Setup, in the Quick Find box, enter `Embedded`, and then select **Embedded Service Deployments**.
2. Click the **Help_Center_Chat** deployment.
3. Edit and activate the deployment's pre-chat form.
    a. Click **Edit Pre-Chat** on the Pre-Chat tile.
    b. On the Pre-Chat Form page, select **Activate the pre-chat feature**.
    c. In the Visible Pre-Chat fields section of the page, add a First Name, Last Name, and Email field. Mark Email and Last Name as required. Then, click **Save**.



4. On the Help Center Chat deployment page, click **Publish**.
5. To test your messaging deployment, click **Test Messaging** on the Test Messaging tile. You see a page with a chat button.
6. To start a conversation, click the chat button. Enter the contact information for an existing contact and then click **Start Conversation**. If you completed the steps so far successfully, your agent joins the conversation and responds to the messages you send.



Create an Experience Cloud Site

To test your agent's full functionality, such as its ability to escalate and manage cases, publish it on an Experience Cloud site. To get a site up-and-running quickly, use the Help center template.

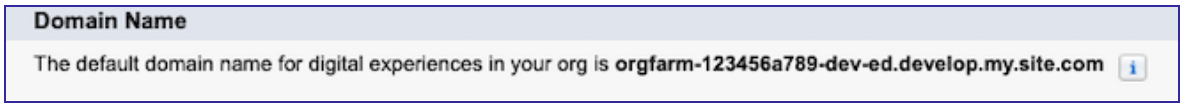
1. From Setup, in the Quick Find box, enter `All Sites`, and then select **All Sites**.
2. Click **New**. Then, select the **Help Center** template.
3. Click **Get Started**.
4. For the help center name, enter `Makana Help Center`.
5. For the help center URL, enter `help`.

Customize and Publish the Experience Cloud Site

Add the Embedded Messaging component to the Makana Help Center site and add the site to your CORS allowlist. When you're ready, publish the Makana Help Center.

1. In Experience Builder, click the **Components** tab.
2. If you already have the Embedded Messaging component on the page, go to the next. If not, on the Components panel, select the **Embedded Messaging** component and drag it onto the template footer.
3. When you're ready, click **Publish**.
4. Confirm that your Experience Builder site domain is in your CORS allowlist. If you notice an issue with the *.live-preview domain, add it to the CORS allowlist as well.

a. From Setup, in the Quick Find box, enter `Digital Experiences`, and then select **Settings**. Copy or take note of the default domain name in the Domain Name section.



b. From Setup, in the Quick Find box, enter `CORS`, and then select **CORS**.
c. If you already have the default domain name in the Allowed Origin List, go to the next step. If not, select **New** and then enter the default domain name URL. Add `https://` to the beginning of the URL and then click **Save**.

Test the Agent on the Website

Before you deploy the Makana Customer Support agent to a staging or production environment, verify that all the parts of its messaging deployment are set up correctly. Make note of areas where you can improve and iterate on the deployment, such as further customizing the escalation experience or Service Console app.

1. On the All Sites Setup page, open your Experience Cloud site by clicking the URL.
2. To start a conversation, click the chat button.
3. Enter the contact information of an existing contact and then click **Start Conversation**.
4. Test the escalation process when there are no service reps available.
    a. In the chat window on the Experience Cloud site, ask the agent to transfer you to a human.
    b. Confirm that the agent lets you know that there are no service reps available and offers to create a case.
    c. After the agent lets you know that it created the case, confirm that the new case exists in Salesforce.
5. Test the escalation process when there's at least one service rep available.
    a. End the chat and start a new one.
    b. In a different browser tab, open the Service Console app.
    c. From the sidebar, select **Omni-Channel** and then change your status to Online.
    d. In the chat window on the Experience Cloud site, ask the agent to transfer you to a human.
    e. In the Omni-Channel sidebar, accept the chat request. Then, click the request to open the messaging session record. Send a message and then confirm that it is sent in the Experience Cloud site chat window.

Deploy the Agent to Staging or Production

After testing the messaging deployment in your sandbox, package and deploy your agent to a staging or production environment.

Make sure that your staging or production environment has required licenses for your agent, and that Einstein Generative AI, Agentforce, and Data Cloud are enabled.

1. Review the metadata components for Agentforce.
2. Retrieve the latest metadata from your sandbox org using Salesforce CLI and generate a package file for your agent. Include any custom Apex, flows, and prompt templates for your agent actions. Also include any permission sets assigned to the agent user.
3. Create the agent user in the target org.
4. Deploy the Agentforce metadata to the target org using Salesforce CLI.
5. Verify that the agent is active.
6. Add the deployed permission sets to the agent user.
7. Make sure all deployed flows have the correct version active. To manage a flow's versions and see version statuses, click the name of the flow in the Flow Builder navigation header.
8. Publish the web deployment in the target org.

After deploying to your target org, confirm that the agent's subagents, instructions, variables, and filters were deployed correctly. Make sure that all permissions have been deployed or updated, and that the agent is responding to user requests.

Monitor Your Agent

Makana Medical Devices has successfully deployed their customer service agent to handle inquiries about their CGM-3000 continuous glucose monitor. Now they need to ensure the agent performs well and delivers the business value they expect.



This guide covers how to test, measure, and effectively optimize your AI agent's performance in a real-life environment after it's been launched.

Starting here? *Make sure you have the right permissions for the job.*

Before We Begin

This chapter is built to provide guidance around building a continuous monitoring strategy and using Agentforce tools to monitor and improve your agents. Since we just launched the Makana Support agent, there won't be enough data to walk through a full monitoring strategy with live data. Use these examples to improve your production agents.

To review, Makana's team want to ensure their agent meets the business objectives they defined during planning:

- Improve deflection rate by 40%
- Improve customer satisfaction scores by 10%
- Reduce average case handling time by 25%

The team needs to track these metrics and identify specific areas where the agent can perform better.

Introducing Agentforce Observability

Agentforce Observability is the suite of tools that gives agent managers deep visibility into AI agent interactions. Use it to spot trends, debug behavior, and tune agents toward your goals and better user satisfaction. It includes:

- **Agent Analytics** – A broad view of agent sessions with insights into how the agent is behaving in production: interaction latency, usage patterns, and performance metrics such

as overall quality scores and success rates. Analytics helps answer questions like “Is my agent working as expected?” and “Where should we improve?”

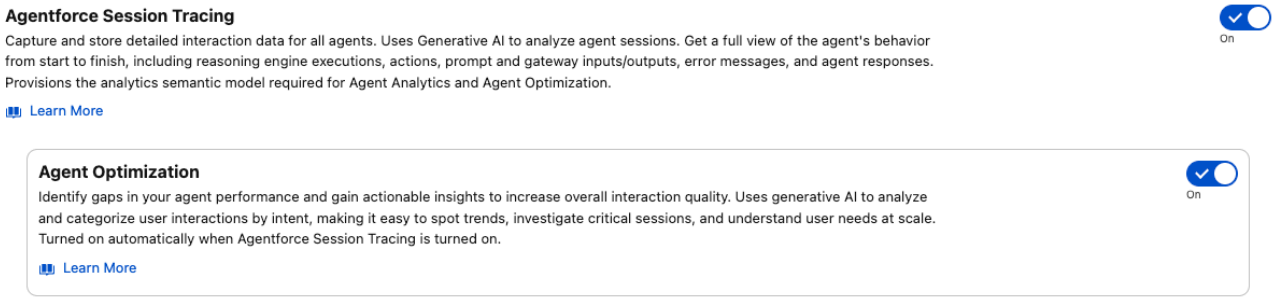
- **Sessions & Intents** – Intent-level detail for closer inspection. An LLM groups similar user goals across sessions into intents, so you can move beyond whole-session summaries and find cases where the agent was inaccurate, misleading, or off-topic, or where configured actions did not run as expected.
- **Scorers (Beta)** – Custom scorers grade agent answers against your criteria using an LLM judge. Create and manage them in Testing Center and apply them to specific sessions.

Set Up and Configure Agentforce Analytics, Optimization, and Session Tracing

Agentforce Analytics and Optimization are designed for comprehensive session analysis, capturing every turn and event across your agent sessions using the unified Session Tracing Data Model (STDM). Agentforce Analytics uses Tableau Next for its dashboard visualization. To view them in Agentforce Analytics, you must manually install the libraries in addition to the setup steps.

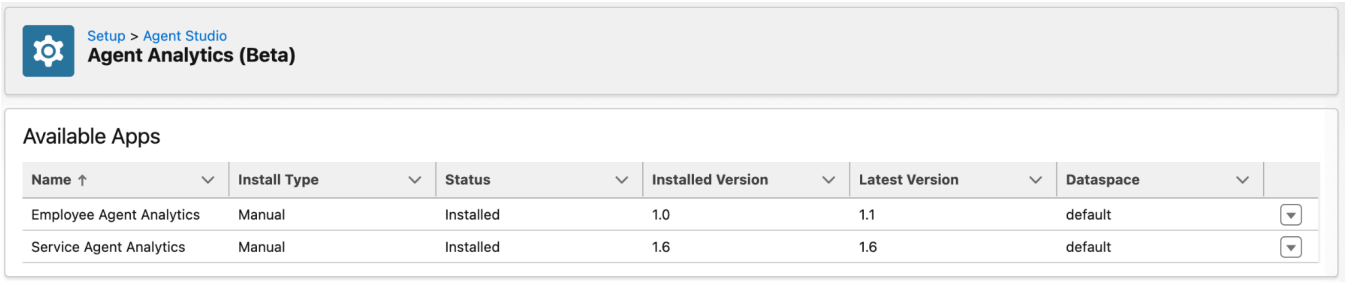
To set up Agent Analytics, turn on Session Tracing to collect and store session data. Session Tracing also provisions the required Analytics model entities to perform queries.

1. Verify that you have the latest version of the Salesforce Standard Data Model (version 1.130 or higher) in your org (sandbox or production). From Setup, go to **Apps** -> **Packaging** -> **Installed Packages** -> **Salesforce Standard Data Model**. Get the required version, and follow the instructions.
2. From Setup, in the Quick Find box, enter `Einstein Audit, Analytics, and Monitoring Setup`, then select **Einstein Audit, Analytics, and Monitoring Setup**.
3. Toggle on **Agentforce Session Tracing** to start collecting data and to provision the Data Model Objects. Note that turning on data collection increases your org’s credit consumption rate.
4. Confirm **Agent Optimization** is also enabled. It should be auto-enabled once you toggle Agentforce Session Tracing.



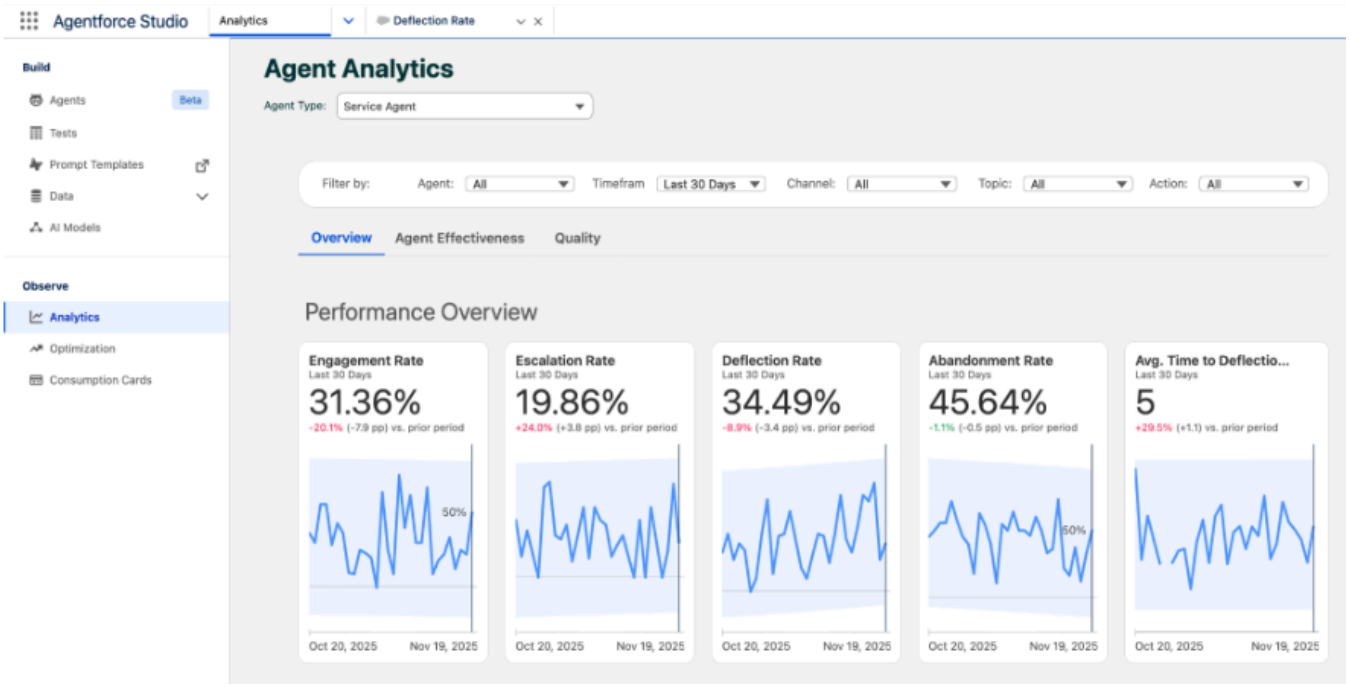
5. From Setup, go to **Agent Analytics**.

6. Install the **Service Agent Analytics** app. When prompted, select the default data space.



Review Agent Performance in Agentforce Studio

1. From the App Launcher, enter `Agent`, and then select the **Agentforce Studio** app.
2. In the Observe section, select **Analytics**.

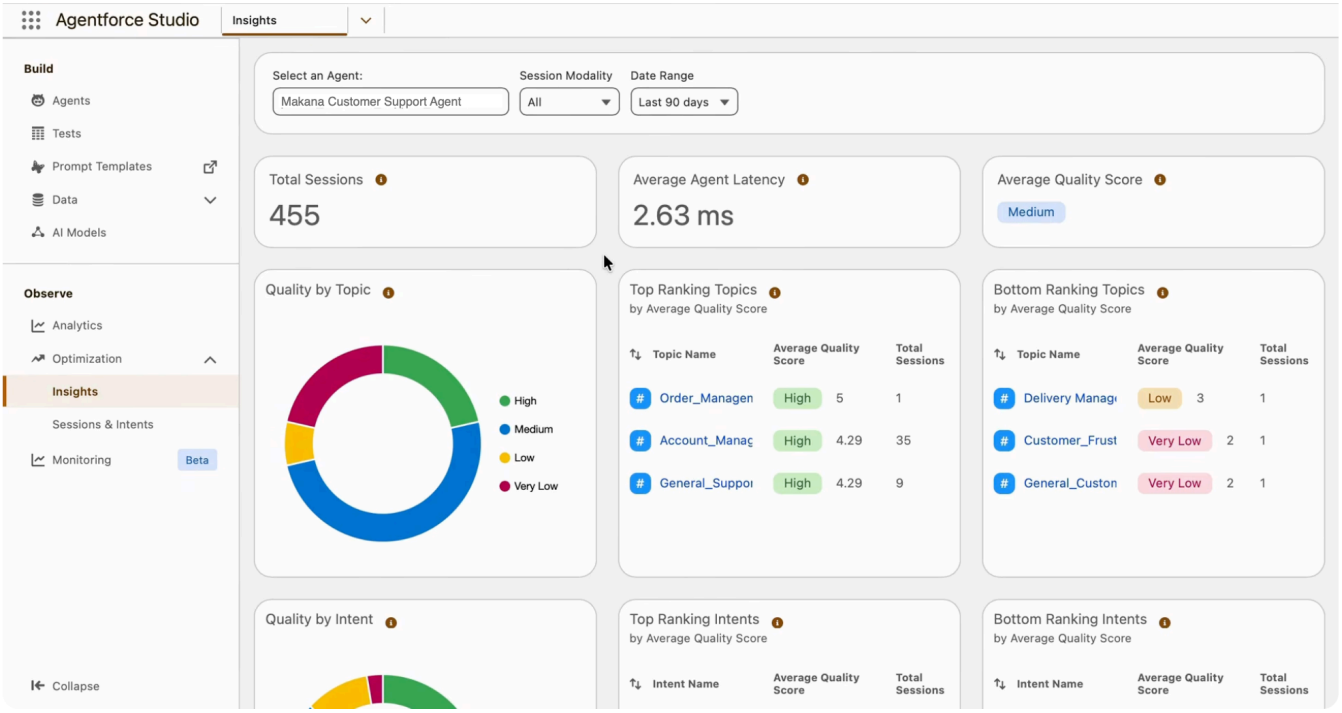


From here, Makana is able to view overall performance metrics of the agent and filter by timeframe, channel, subagent, or action. The metrics in the Overview Tab provide details on performance rates and usage. The Agent Effectiveness tab dives deeper into performance, identifying performance by session. The Quality tab provides metrics about agent response quality. Learn more about each metric at Get Insights with Agent Analytics Dashboards.

These dashboards are great for sharing out agent performance over time. Makana's key stakeholders will want to know how the agent is performing, and Agentforce Analytics brings that data all into one place.

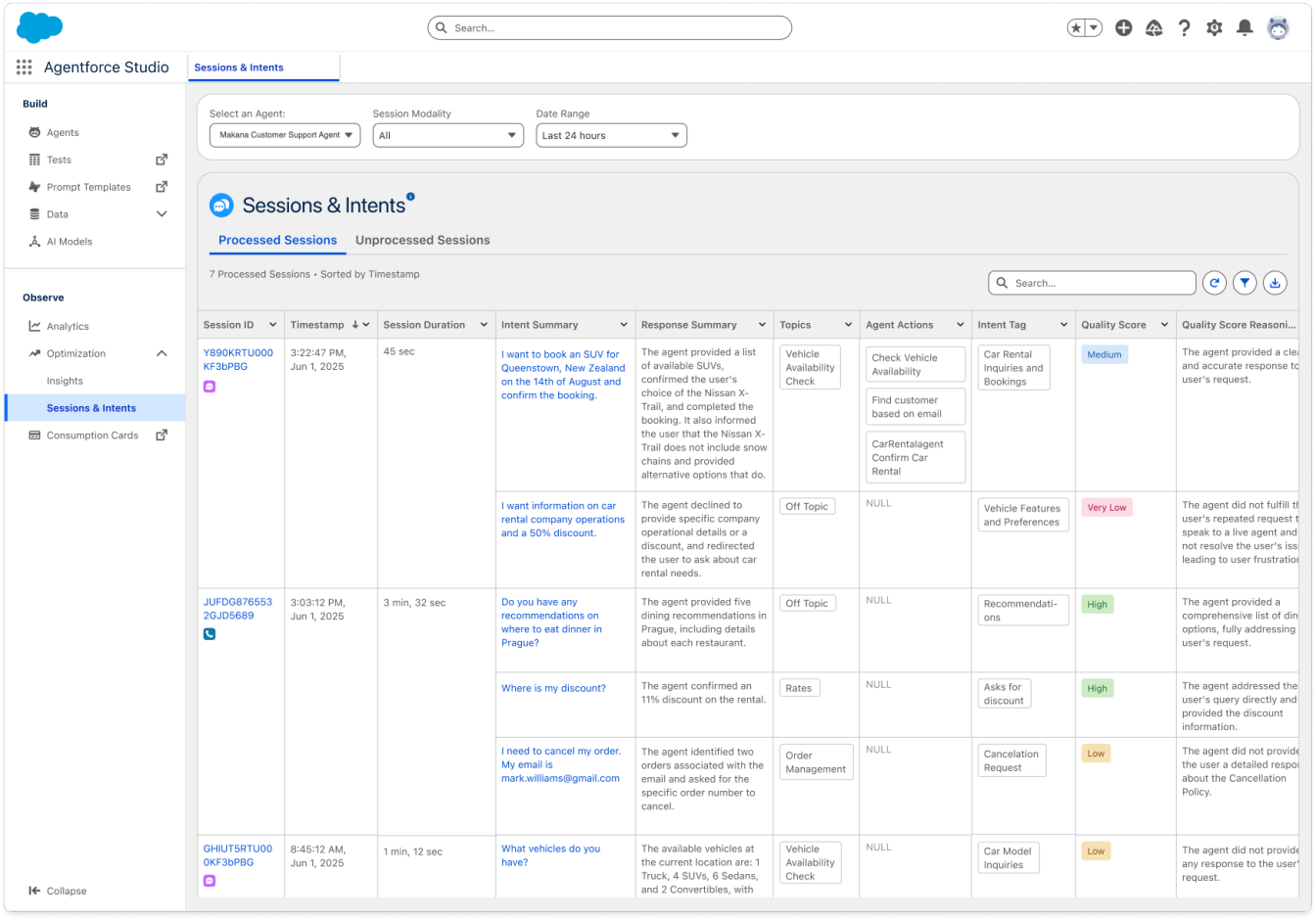
Dig Deeper into Agent Performance with Agentforce Optimization

But what if Makana wants to dig deeper into their agent and improve it? Agentforce Optimization allows you to get insights on agent performance, identify poor performing subagents, and review sessions and intents to identify the issue. Optimization extends session tracing by focusing on session intents, which are sub-sessions representing interactions with a specific user intent. An advanced LLM identifies clusters, and tags these intents, facilitating queries and insights.

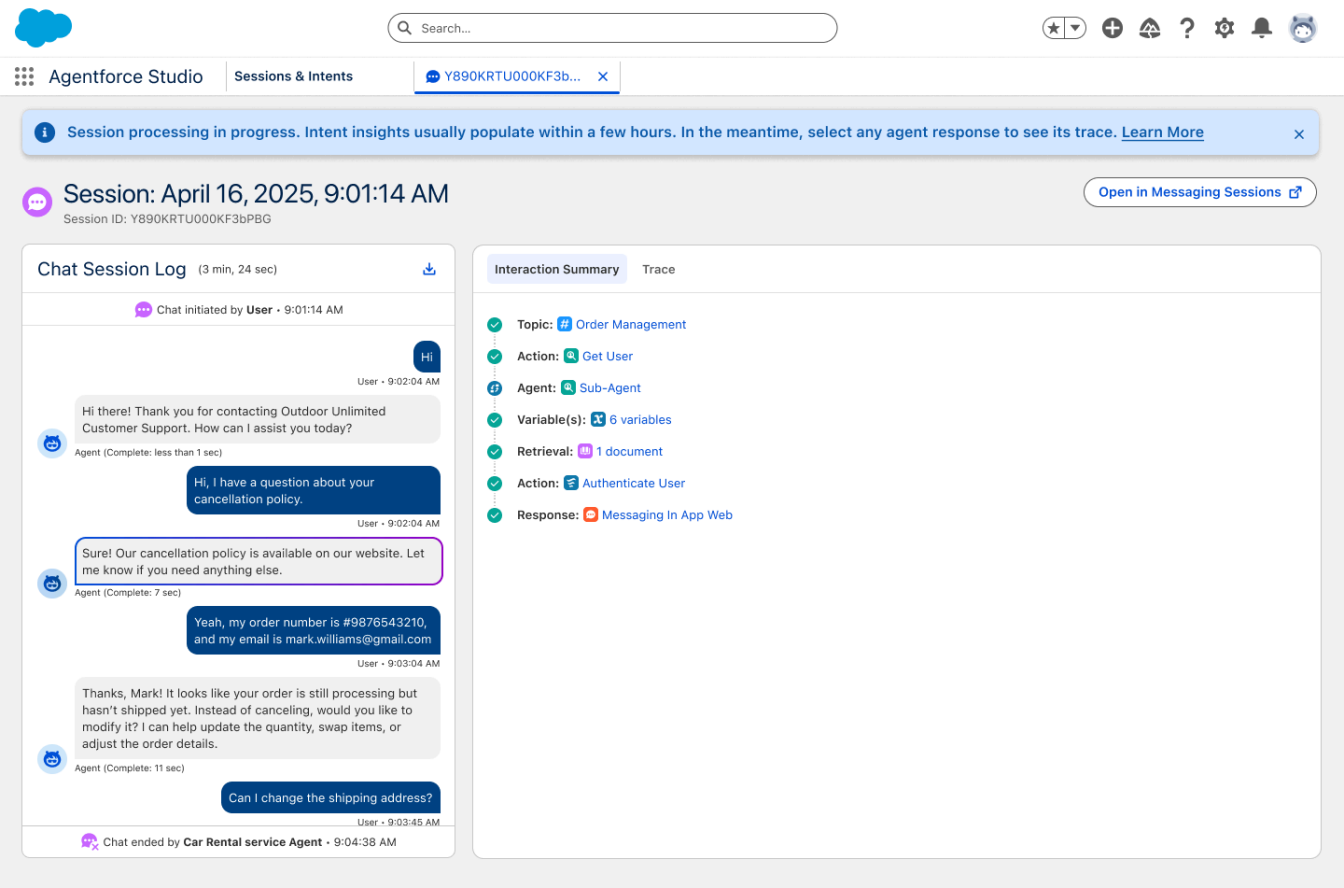


Review Sessions and Intents to Identify Areas of Improvement

Makana can use the Sessions and Intents feature to see how the agent classified intents, and how that affected performance.



When Makana clicks on a session ID, they can review the conversation and then click into each conversation turn to review the session trace. By using the instructions in Manual Testing and Verification to get the details, the agent manager is able to review exactly what happened during the session, and then go back to the builder to make adjustments.



Monitoring Strategy Methods

This sample plan shows you how to use Agentforce Analytics and Optimization for a comprehensive view of agent performance. There are three major methods in a monitoring strategy:

Stage	When to Begin	Goal
Post-Launch Monitoring	Immediately following agent launch	Confirm agent is working as designed and metrics are being populated
Intent Monitoring	30 days after agent launch; 24 hours after a closed intent; up to 30 days for deep data insights	Identify actions for improvement
Root Cause Analysis	Minimum clustering of 100 intents	Use clusters to identify issues and dive deep into sessions

Post-Launch Monitoring

Immediately post-launch, use Agentforce Analytics to check if data is collected and ingested and view your dashboard for key metrics:

- Usage patterns (e.g., unique users, number of sessions)
- Overall performance metrics like average quality score and escalation rate
- Agent latency

You can also dive into specific sessions. Select sample sessions from key subagents your sessions cover, and inspect them turn-by-turn to see if they flow as expected.

Intent Monitoring

When your system begins to cluster and tag intents within interactions, you can view the intents tab in your app and filter the intents by quality scores. If you inspect individual specific intents with low quality scores, you can uncover possible agent actions that need refinement or improvement.

Recommended Timeline:

- Initial Data Analysis: If you have enabled STDM previously in setup, The first pipeline run will generate insights for intents from the last 30 days. Subsequent pipeline runs will use data from the last 7 days (If you haven't enabled STDM yet, insights will be analyzed only from the moment you enable it onwards).
- Intent Generation: The system generates intents (sub-sessions representing a specific user intent) on a frequent schedule. A pipeline run is initiated under one of two conditions:
    - Every 9 hours if there is at least one new closed session.
    - Every 3 hours if there are more than 10 new closed sessions since the last run.
    - A session is marked closed after 3 hours of inactivity.
- Clustering: The LLM clusters similar intents and applies tags once per week over the weekend. This process is crucial for identifying trends and widespread issues.

Root-Cause Analysis

This stage focuses on using the detailed data to perform root-cause analysis and identify specific areas for improvement. Drill Down with Agentforce Optimization and move from the "what" of Analytics to the "how" of Optimization. Use the tool to perform a granular inspection of specific intents and clusters.

Timeline:

- This phase typically begins after the first weekly clustering run is complete, as this provides the necessary data for deep-dive analysis.
- For the clustering algorithm to provide meaningful results, a rule of thumb is that around 100 intents are needed. This can typically be generated from about 100 sessions, though this depends on your specific data.

Actions:

1. Your main Analytics dashboard will give you the initial indication of performance gaps and flagged interactions.
1. Identify Performance Gaps: Agentforce Optimization automatically flags interactions where the agent provided inaccurate, misleading, or irrelevant information (including “hallucinations”) or failed to execute a configured action.
2. Analyze performance of individual intents according to their Quality Score:
    - To check an individual moment's quality, examine its quality score, ranked from 1(lowest) to 5(highest).
    - Then review the Quality Score Reasoning, which provides a specific explanation for the assigned score (e.g., ‘agent didn't address the pricing question’).
3. Correlate Metrics for Deeper Insights: Combine multiple calculated fields in a single query to find complex problems. For example:
    - Querying for intents with a high duration and a low quality score helps identify areas where users spend a lot of time but receive poor-quality responses.
    - Adding the Escalation Rate to this query can confirm whether these interactions are leading to escalations.

Building a Monitoring Strategy

We recommend using the monitoring methods over time to build out a full monitoring strategy.

Daily

1. Monitor high-level KPIs in Agentforce Analytics. When a negative trend is spotted (e.g., rising escalation rate), use Agentforce Optimization to drill down and identify the specific intents or user intents causing the issue.
2. Monitor internal and external feedback about the agent. If an issue arises, perform a root cause analysis to identify the problem.

Weekly

1. Monitor intents to identify agent areas of improvement.

Monthly

1. Collaborate with Subject Matter Experts (SMEs) and AI Agent Builders to analyze the root cause identified in intents. This could be a knowledge gap, a poorly configured prompt, or an issue with the Retrieval-Augmented Generation (RAG) configuration.
2. Implement Changes: Implement the required changes, such as refining agent prompts, updating knowledge bases, or modifying agent actions and subagents.
3. Test and Validate in a Sandbox Environment: before deploying improvements to production, validate all changes in your sandbox environment.

    - Manual Testing: You can interact with the agent manually via Agentforce Builder or a testing center to review its reasoning utterance-by-utterance, and perform manual adjustments if needed.
    - Batch Testing: For efficiency, you can run up to 100 test cases at once without deactivating the agent.
4. Redeploy & Monitor: Once the changes have been validated in the sandbox, redeploy the updated agent to production. Use Agentforce Analytics to monitor KPIs to confirm that the changes had the desired positive impact, creating a continuous cycle of learning and optimization.