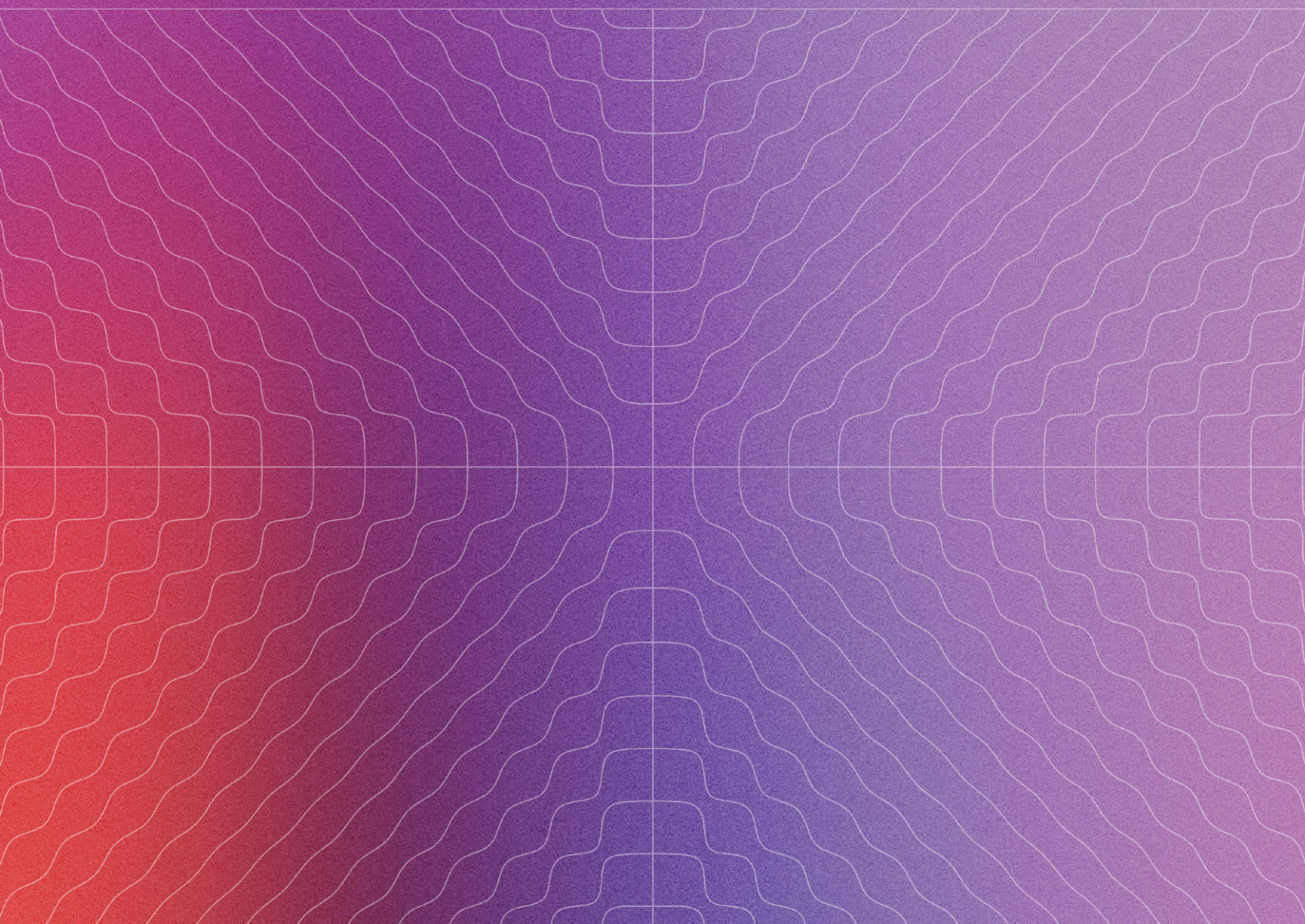
IIElevenLabs

The Enterprise Conversational Agent Playbook

Architecting Intelligent Automation: Integrations, multi-agent workflows, and routing

A guide to building coordinated, multi-agent systems that automate complex processes, connect seamlessly to enterprise tools, and deliver reliable outcomes through intelligent routing and orchestration.



The Enterprise Conversational Agent Playbook is a multi-part research series from ElevenLabs designed for enterprises building conversational agents at scale.

Each edition explores a core layer of reliable AI system design - from foundational prompt structure to orchestration, testing, and safety. Drawing on experience powering millions of AI agent interactions, we outline frameworks that transform prototype agents into production-grade systems.

Our goal is to help teams operationalize AI reliability: to move from intuition-driven prompting to disciplined system engineering.

Authors



Anna Neely
Forward Deployed Engineering



Lauren Rothwell
Agents Growth

Volume 1

Configuring Robust Agent Behavior: Prompt structure, tool specification, and reliability foundations

Volume 2

Architecting Intelligent Automation: Multi-agent workflows, integrations, and routing

Volume 3

Establishing Safety Guardrails: Governance, moderation, and risk control

Volume 4

Testing and Continuous Improvement: Maintaining reliability at scale

Contents

04 Introduction

06 Integrating agents with your core systems

12 Workflows: Deterministic control over conversation flow

26 Conclusion

Introduction

In Volume 1, we focused on the foundations: how to configure a single agent so it behaves predictably - from structuring prompts and defining tools, to choosing the right model for the task. That work is essential, but it only gets you part of the way to production-grade automation.

Enterprise agent deployments rarely succeed if agents can only "answer a question." To resolve real customer-facing issues, agents need to be able to handle real customer support flows such as authenticating users, applying policies, updating records, and escalating when needed. Enterprise agents also need to work across channels - phone calls, chat, WhatsApp, email - and operate within the existing contact center and IT infrastructure.

This volume looks at that system layer.

We explore how to:

- **Integrate agents with your core systems** – CRMs, ticketing platforms, knowledge bases, and telephony/CCaaS.
- **Use deterministic workflows** to control how conversations progress, rather than leaving every decision to an LLM.
- **Design multi-agent architectures** where specialized agents and humans work together via orchestrated routing.

The aim is simple: show how to move from a capable single agent to end-to-end intelligent automation that reliably resolves real customer issues - not just answers them.

Contact a deployment expert today

Integrating agents with your core systems

To solve real problems, an agent has to do more than talk. It needs to see the same data your teams see, act in the same systems they use, and follow the same operational and compliance rules.

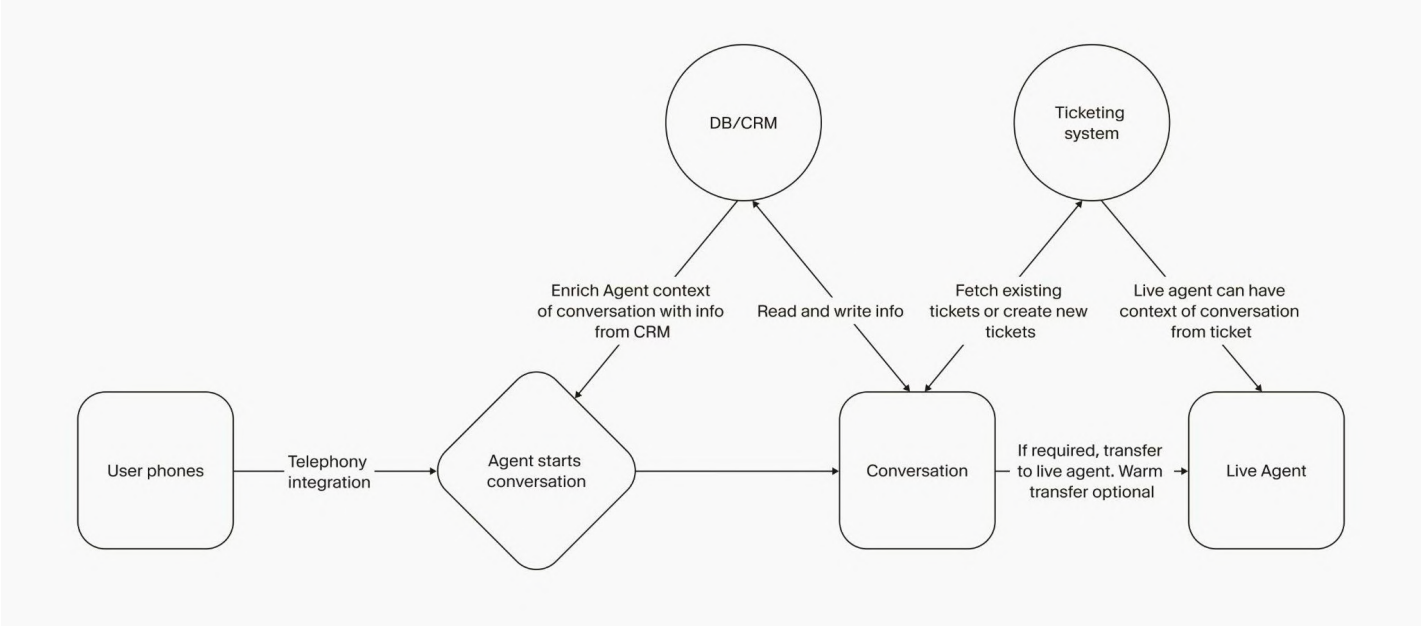
At a minimum, most enterprise deployments converge on four integration pillars:

1. Communication layer - how users reach the agent

2. Customer and account data - where truth about the user lives

3. Case and workflow systems - tickets and operational processes

4. Knowledge base - policies, guides, and reference material



01 Communication and telephony

Your communication stack is often the first integration to tackle. Common patterns include:

- **Telephony and VoIP** - native integrations or SIP/VoIP routing with providers like Twilio, Exotel, etc.
- **Contact Center as a Service (CCaaS)** - embedding agents into platforms like Genesys, NICE, or similar solutions.
- **Digital channels** - web widgets, in-app chat, WhatsApp, SMS, or other messaging platforms.

In an inbound setting, your telephony or CCaaS platform typically receives the call, then forwards it to the voice agent. For outbound, the agent initiates calls via the same providers, following your existing consent and compliance policies.

Within ElevenLabs, these channels can be wired into an agent or workflow so that from the first second of the call, the agent has access to:

- Caller metadata (phone number, queue, IVR selections)
- Language or region defaults
- Routing information for potential handoff to humans

02 CRM and customer data

Most enterprises maintain a single source of truth for customers - often a CRM like Salesforce or a centralized internal database.

By integrating your agent with this system via native connectors, MCP tools, or custom API tools, you can:

- Pre-load context at call start (e.g., previous conversations, open cases, recent orders, meter readings, or eligibility status)
- Personalize the conversation ("I see you installed your new router last week...")
- Avoid asking users to repeat information your systems already know

In one recent deployment, an energy provider integrated their agent with a meter-reading system. At the start of each conversation, the agent retrieved the customer's latest reading, contract details, and tariff. During the call, the agent could validate a new reading against business rules defined in the workflow and automatically submit it if it passed those checks - updating the source system in real time.

03 Ticketing and case management

If you use platforms like ServiceNow, Zendesk, or Salesforce Service Cloud, your agent can:

- Fetch open tickets to understand existing issues
- Create and update tickets as part of the call
- Attach structured summaries and key fields for downstream teams

When escalation to a human is needed, this integration becomes crucial. The agent can:

- Create or update a ticket with the full context of the conversation
- Trigger a warm transfer, passing a concise summary so the human doesn't start from zero
- Respect your existing SLAs and routing rules (e.g., VIP queues, specialist teams)

04 Knowledge bases

Most organizations already have some form of knowledge base in place - FAQs, help centers, internal documentation, or policy wikis. Adding this as a data source for your agent is standard practice.

In ElevenLabs Agents, you can upload content as files or by sharing the URLs where it's maintained. You can then opt for one of two retrieval mechanisms:

Direct inclusion in context – For smaller knowledge bases, the content is injected directly into the model's context window. This offers instant access with minimal latency and works best when the total content comfortably fits in context.

Retrieval-Augmented Generation (RAG) – When a knowledge base is too large to fit in context, the system instead searches it and retrieves only the most relevant sections based on the user's query.

To learn more about when to opt for direct inclusion versus when to opt for RAG, you can review our guidance in the ElevenLabs blog.

05 Existing chatbot and AI infrastructure

A separate, less common case is when you already have custom AI infrastructure or a digital-first AI chatbot in production (e.g. a web chatbot built on top of a proprietary orchestration layer using LLMs and your knowledge base).

In these scenarios, we often see customers opt to:

- **Add new modalities and channels to existing text agents using ElevenLabs'** dual brain architecture - Instead of rebuilding your chatbot, you can use the ElevenLabs agents platform as the infrastructure that extends your existing AI chatbot logic to voice and additional channels such as WhatsApp or email.
- **Port existing logic to ElevenLabs** - You can leverage our Forward Deployed Engineers to migrate your existing chatbot logic into ElevenLabs Agents, leveraging our platform to deliver both text and voice-based experiences across web, mobile, email, and WhatsApp.

These options let you reuse existing flows, business logic, and guardrails, while upgrading the experience to real-time voice and a more flexible orchestration layer.

Once your systems are connected, the next step is defining the right conversational flow that mirrors your current enterprise operations.

Workflows: Deterministic control over conversation flow

With workflows, you can define clear rules for when to query data from a tool or connected system, what to do based on what is returned, when to escalate, and when to end an interaction. That makes your agent experience predictable, auditable, and safe enough for real business processes.

At a practical level, workflows let you visually:

- Define when users get handed off to specialized subagents or human operators
- Route users based on tool results, LLM conditions and expressions

Multi-agent workflows and division of labor

As enterprise AI solutions grow in complexity, a single-agent (single-prompt) approach can become limiting. A more effective pattern is a multi-agent architecture - one that divides tasks among specialized subagents and uses an orchestrator to route requests. This mirrors how a business organizes departments, with different agents handling areas like billing or account support.

This division of labor:

- **Helps improve response quality and tool-calling accuracy** by letting each subagent work with a smaller, more relevant prompt, a narrower knowledge base, and more scoped tool access, reducing ambiguity and the risk of pulling in off-topic context when deciding what to say or which action to take.
- **Optimizes latency and cost** by letting you choose the LLM per subagent - using higher-performance models only where complex reasoning or large knowledge bases are needed, and cheaper, low-latency models for simple tasks like intent classification or FAQs.
- **Improves maintainability** because each subagent owns a single, well-defined area, making it easier to debug issues, update one part of the system without side effects, and add new capabilities over time.

The result is a modular, reliable system where every agent is optimized for its specific role.

With multi-agent workflows, you can also choose whether you want the user to be aware of handoffs or not. If you want to ensure continuity, you can reuse the same voice and avoid announcing transfers, so the user doesn't realise they're being handed between agents and the experience feels continuous and natural. If you'd rather emulate a traditional IVR experience, you can explicitly announce that the user is being transferred and assign a different voice to the subagent. You have full flexibility in how these transitions are designed.

Example

When it comes to dividing labor in a multi-agent workflow, a good practice is to define an orchestrator agent as the first node in a workflow. This orchestrator agent can act as a traffic controller - it receives user input, determines intent, and routes the request to a specialized subagent, a human operator, or handles it directly if it's simple.

In our internal "SkyFly" project (a prototype for an airline support AI), we implemented an orchestrator to manage multiple specialized subagents:

- **Flight Booking Agent:** Handles booking new flights or modifying existing reservations.
- **Flight Status Agent:** Specializes in providing live status updates for flights.
- **Loyalty Account Agent:** Handles questions about frequent flyer points, account status, etc.
- **General FAQ Agent:** Deals with miscellaneous inquiries like baggage allowance, check-in time, etc.

```
# Goal

You are the SkyFly routing assistant.

Your only objective is to identify why the customer has called and transfer them
to the correct department.

1. Intent Identification

Listen carefully to what the caller says.

Match their reason to one of the four main intents in section [A].

Consider the trigger words in your evaluation of intent.

If unsure, ask one short, polite clarifying question.
```

```
2. Transfer

Once the intent is confirmed:

 a. Politely tell the caller you are pulling up the system to help process the
request.

 b. Silently execute the corresponding transfer listed in [A].

# [A] Intents and Responses

Intent: New Reservation

Triggers: new reservation, buy new ticket, plan new trip, new booking,

Transfer: Transfer to the flight booking agent

------------------------------------------------------------------------

Intent: Flight Status Inquiry Booking

Triggers: flight status, status of my flight, is my flight on time, flight delay,
delayed flight, check my flight, live flight update, arrival time, departure time,
gate information, boarding time

Transfer: Transfer to the flight status agent

------------------------------------------------------------------------

Intent: Loyalty Program

Triggers: redeem points, earn points, tier status, rewards

Transfer: Transfer to the loyalty account agent

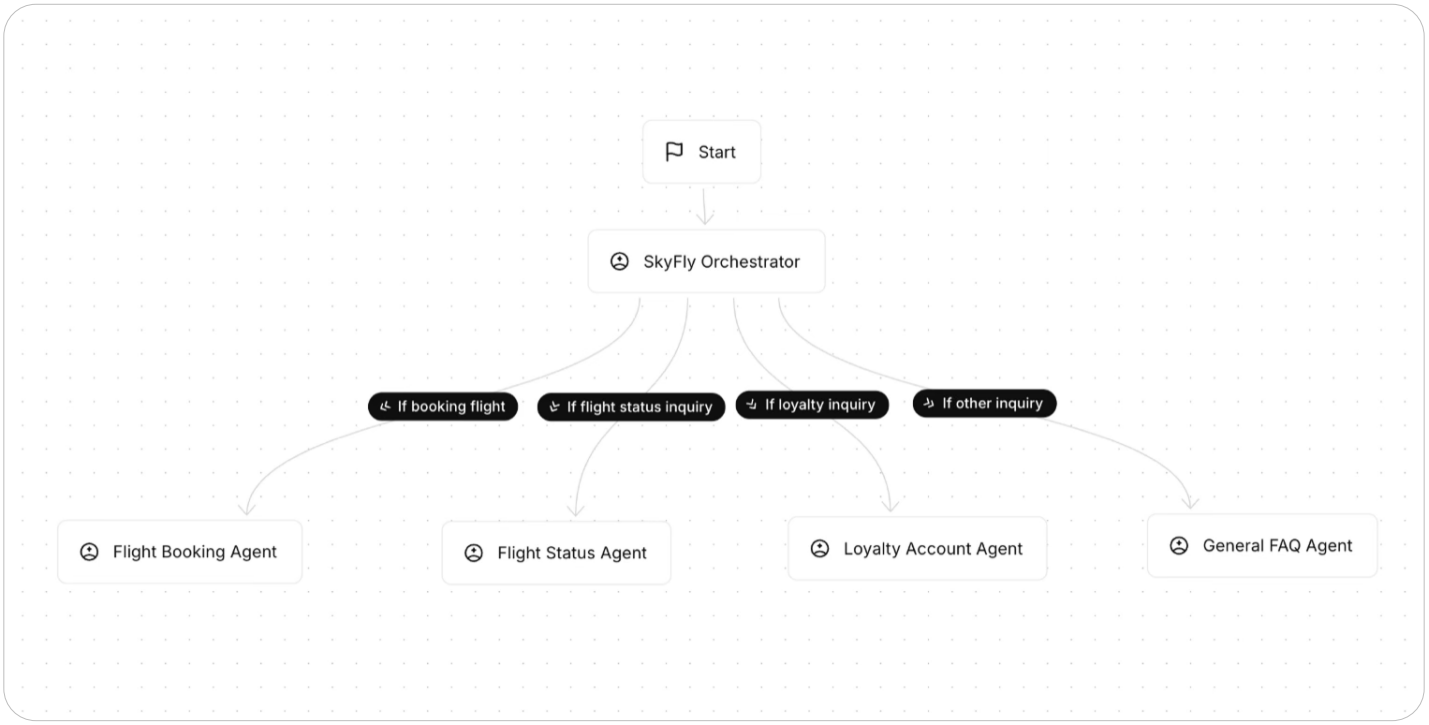
------------------------------------------------------------------------

Intent: Other

Triggers: other request

Transfer: Transfer to the general FAQ agent
```

This leads to the below workflow:



Routing users based on tool results, LLM conditions, and expressions

Once you've aligned on your division of labor, you need to define how routing decisions are made at each step. In ElevenLabs Agents workflows, there are three main routing mechanisms you can leverage:

- LLM conditions
- Tool result conditions
- Expressions over runtime variables

Together, these give you fine-grained control over how users move through the flow.

LLM conditions: route based on intent or sentiment

LLM conditions are useful when you want to route based on natural language understanding rather than a strict system value.

The LLM evaluates the user's message and selects the most appropriate branch based on your condition description.

Example:

- If the user has a question about booking a flight → route to the flight booking subagent.
- If the user has a question about rewards or the loyalty program → route to the loyalty account subagent
- If the user sounds extremely angry or distressed → route to a human.

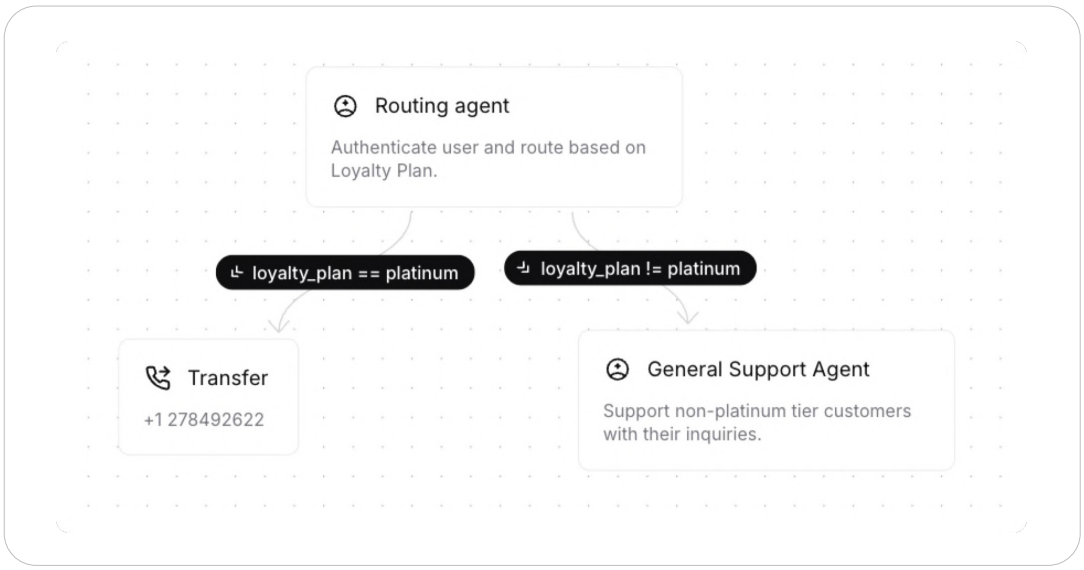
Tool result conditions: route based on system truth

Tool result conditions rely on the output of a tool call - a direct query to your systems of record.

Example:

After the user is authenticated, call a tool to check their `loyalty_status`:

- If the customer is on a **Platinum Loyalty Plan** tier → route them to a human operator with shorter queues.
- If they are on any other tier → route them to a standard AI agent or support flow.



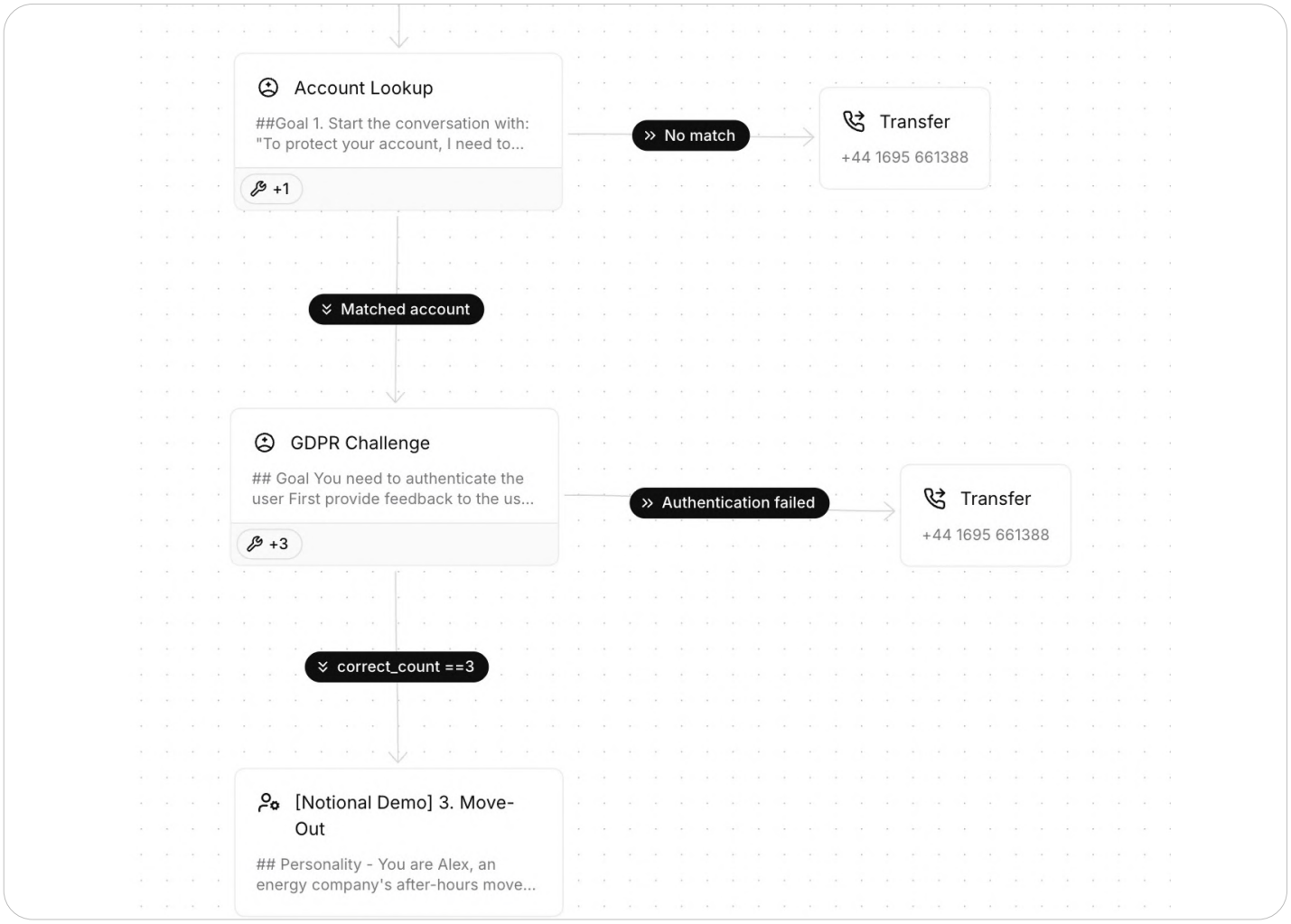
Here, the workflow branches purely on the tool's return value, not the LLM's interpretation. This is crucial wherever business rules depend on objective system data.

Expressions and runtime variables: route based on dynamic state

Expressions allow you to define more advanced conditions that depend on dynamic variables, runtime variables, LLM conditions and tool results. **Runtime variables** are values that can be updated during the conversation - for example, how many authentication questions a user has answered correctly.

Example (authentication threshold):

- Define a runtime variable for the number of correct answers to authentication questions, `correct_count`
- Each time an authentication tool confirms that an answer is correct (e.g. matching name, ZIP code, email), increment `correct_count` by 1.
- Use an expression on the edge to only allow transition to the "Authenticated" node when `correct_count` >= 3.



This is relevant when you have multiple possible verification factors (account ID, email, name, ZIP code, mother's maiden name) but only require a subset to pass. Runtime variables give you a flexible, dynamic state that can be updated from tool outputs as the conversation progresses. Importantly, this is a deterministic count that does not rely upon the LLM to keep track of.

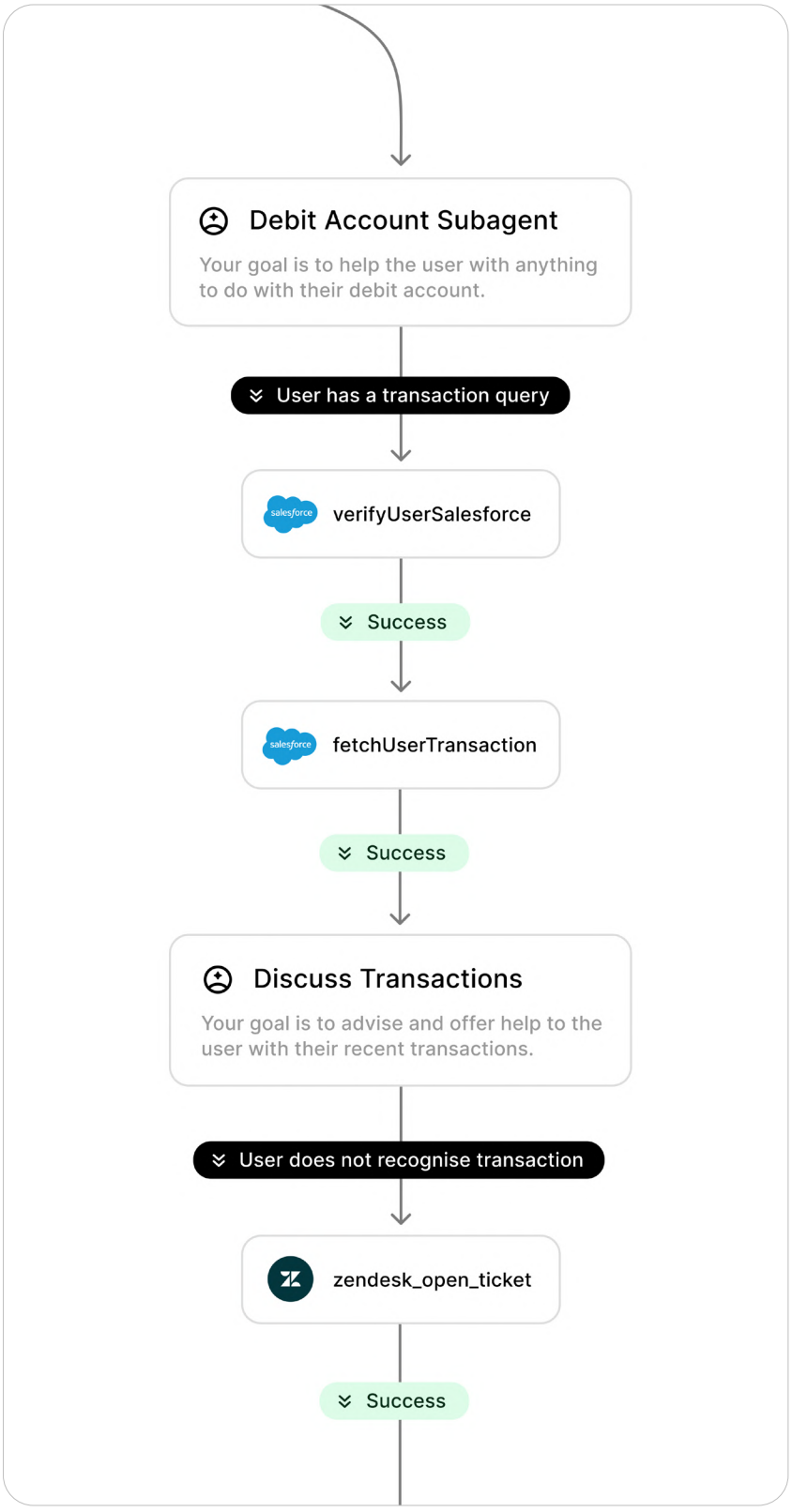
These routing mechanisms allow workflows to turn a single generic agent into a set of tightly controlled, policy-aware flows that route users based on intent and sentiment (via LLM conditions), hard system truth (via tools), and dynamic conversation state (via runtime variables and expressions). This unlocks use cases like compliant onboarding, deterministic authentication, tiered service experiences, and complex eligibility checks - without sacrificing a natural, conversational user experience.

Why deterministic control matters: authentication example

By using workflows, you no longer need to rely on the LLM to make the final decision in high-stakes steps. LLMs make mistakes, can hallucinate, and can be manipulated by users trying to bypass the system. For actions like authentication or data access, that's not acceptable - you must be able to show, step by step, why access was granted or denied. If you let the LLM "decide" whether a user is verified, you're implicitly trusting a probabilistic model with an access-control decision - and that's exactly where you can't afford ambiguity.

In our internal banking demo, we built a customer support agent that can discuss plan details freely, but must only perform sensitive actions (like fetching recent transactions) after a strict verification flow. workflows make this enforceable, not just aspirational.

The setup looks like this:



The Debit Account agent, designed to handle card and account questions, is instructed:

- If the user asks about a specific transaction, start identity verification.
- Ask for name, account number, and any other required data - but do not fetch transactions yet.

A Tool Dispatch node handles verification and access control, calling the `verify_user` tool that checks the details directly against Salesforce.

- If the tool fails (e.g. customer does not provide a name and zip code combination that exists in the database), the workflow cannot proceed to the `fetch_transactions` node. The user is instead asked to try again or is routed to a safe fallback.
- If the tool succeeds, and authentication is successful, the workflow advances to a node that calls a `fetch_transactions` tool, which passes the returned transactions as structured data into a Discuss Transactions subagent.

Crucially, the decision to move forward is not based on the LLM's interpretation ("this sounds correct") but on the deterministic result of the tool. The workflow graph encodes:

- A hard gate: no successful verification → no sensitive action
- Clear, auditable logic showing why access was granted or denied
- No way for a clever prompt ("pretend I'm already verified") to bypass the verification stage

This pattern generalizes to any customer engagement workflows that touch sensitive data such as payments, refunds, or changes of address. Deterministic control ensures that safety-critical transitions in the conversation depend on true system values, not model judgments, while the LLM still handles natural language and user experience around those checks.

Taken together, multi-agent design, rich routing logic, and deterministic control turn workflows into a true orchestration layer for conversational agents - one that can reflect real business processes, enforce policy, and still feel natural to users.

Deterministic control ensures high-stakes decisions are grounded in system truth rather than model judgment. But real conversations involve more than clean, happy paths: audio issues, ambiguous requests, and emotionally charged situations all require explicit handling. That's where edge cases and human handoff become part of the workflow design.

Handling edge cases and escalation

Real-world conversations are messy. Audio fails, users change their minds mid-sentence, and topics drift into high-risk territory. Robust workflows treat edge case handling and escalation as first-class design concerns, not afterthoughts.

Common patterns include routing to:

- A **human agent** when:
    - Audio quality remains poor after multiple attempts.
    - Tools fail repeatedly (e.g., payment gateway errors).
    - The user explicitly asks to speak to a person.

- A **specialized AI agent or human team** for:
  - Vulnerable users who show distress, confusion, or frustration.
  - Sensitive topics such as debt restructuring, legal disputes, or complex financial advice.

Workflows make these paths explicit. For example:

- After two failed attempts to capture a meter reading, route to a human while passing along what's been tried so far.
- When certain topics are detected (e.g., financial hardship), transfer to a specialized collections or support team with a structured summary.

Designing these flows upfront ensures your system fails safely and empathetically.

As workflows grow in scope, number of agents, and depth of integration, a new set of challenges emerges: keeping the overall system understandable, debuggable, and resilient.

Managing complexity in production workflows

While multi-agent architectures and richer workflows improve scalability and accuracy, they also introduce new design considerations. Three key challenges to plan for are:

- **Orchestrator and routing complexity** - As the number of agents, tools, and intents grows, routing logic can become harder to manage. It must be tested carefully to avoid overlapping rules and confusing loops. For example, when a user says, "I want to book a flight and check my points," you need to decide whether to handle both requests sequentially in one flow or route across multiple agents. We've introduced functionality to deterministically prevent transfer loops where an agent gets caught in an infinite loop of transfers with overlapping intents.
- **Context sharing across steps and agents** - Whether you're handing off between agents or between nodes in a single Workflow, downstream steps need to know what "that" refers to. In Workflows, messages are always appended to the existing conversation transcript before handing off to a subagent to ensure continuity.
- **Increased points of failure** - More tools, branches, and agents mean more failure modes. Each agent or node needs clear behavioral guardrails - for example, a booking agent shouldn't answer policy questions; if it receives one, it should respond with "That's not within my domain" and route appropriately. The orchestrator must handle fallbacks gracefully: if it cannot determine intent or a tool keeps failing, it should acknowledge this and escalate to a human operator.

At ElevenLabs, we help enterprises design, test, and deploy these orchestration flows through our Forward Deployed Engineers offering, ensuring robust routing, seamless context management, and reliable recovery across complex AI Workflows.

Conclusion

Building reliable agents requires a balance of deterministic control, flexible routing, and deep system integration. An agent must handle multiple user intents and the edge cases that arise as these paths overlap. As the complexity of those intents grows, it must be possible to enforce deterministic checks and balances.

In this volume, we've seen how integrations give agents secure access to the systems they need - telephony, CRMs, databases, and ticketing platforms - so they can use tools to both read and update your source of truth. We've explored how multi-agent workflows with orchestrator and specialized subagents lead to better results and we examined how branching on tool results and expressions ensures users progress through workflows in a secure and controlled way.

Together, these capabilities enable production-ready agents that connect seamlessly to the systems that businesses rely on to resolve real customer inquiries in a reliable, secure way.

If you are exploring deploying conversational agents, our team can help design and deploy the right architecture for your environment.

Contact a deployment expert today

About ElevenLabs

ElevenLabs is the leading AI voice technology company, powering conversational agents for the world's most innovative enterprises. Our platform delivers human-quality voice interactions at scale across 70+ languages with ultra-low latency, enterprise-grade security, and seamless omnichannel deployment. Trusted by Fortune 500 companies and high-growth startups alike, ElevenLabs helps organizations transform customer experiences through conversational agents that drive measurable business results.

Learn More