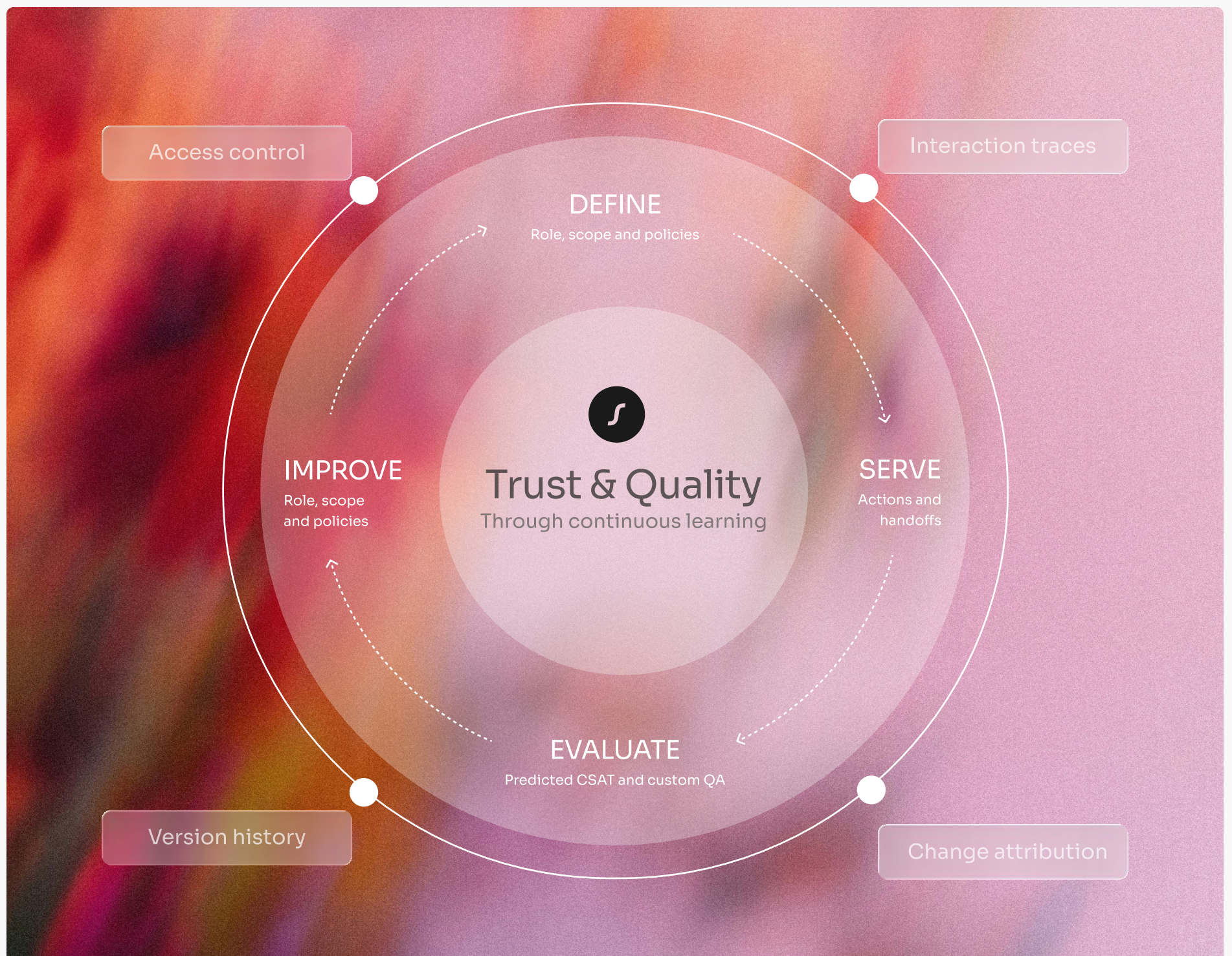


C R E S C E N D O

# Guardrails for Agentic Customer Experience

How Crescendo provides enterprise control and continuous improvement



By Tod Famous, Co-Founder and Chief Product Officer, Crescendo



# Guardrails begin before action

Before an enterprise delegates customer service to AI, it needs a practical answer to a practical question: what keeps the system within its authority when a conversation takes an unexpected turn? Detecting a mistake afterward is valuable. Preventing an unauthorized action requires controls that operate while the work is being done.

Crescendo's approach connects three responsibilities: govern the interaction, evaluate the outcome, and govern the changes that follow. Each addresses a different failure mode.

Instructions establish the job. Technical permissions constrain access and actions. Handoffs provide a path when the system reaches its limits. Evaluation and testing then supply evidence for improving service without casually moving those limits.

Concierge, Crescendo's front-door AI resolution agent, interprets customer requests and chooses how to respond within its assigned scope. It can retrieve knowledge, invoke tools, and transfer work to a person. That flexibility makes clear authority more important: understanding what a customer wants does not establish permission to do it.

Consider an illustrative request to change a delivery address. Recognizing the request is only the beginning. The deployment must establish what verification is required, whether the change is permitted, and which operation may perform it. Focused instructions organize the interaction; the integration must enforce the corresponding access requirements. If evidence or authority is insufficient, the appropriate outcome is a handoff rather than an improvised exception.

The same discipline must extend to improvements. A weak interaction may justify clearer instructions, better knowledge, or a revised integration. It does not, by itself, justify broader permissions. Changes need testing, accountable ownership, and subsequent evaluation.

This is why we regard guardrails as the foundation of the learning loop. They define the standard, make failures diagnosable, and establish how improvements may enter service. Enterprise confidence depends on both the boundaries maintained today and the discipline applied to tomorrow's changes.

Govern the interaction. Evaluate the outcome. Govern the change.



# An agentic core with explicit authority

A predefined workflow directs work through prescribed steps and branches. An agentic system gives the model responsibility for choosing an appropriate path within the authority granted to it. Neither arrangement removes the need for controls; the distinction is where decisions are made.

Crescendo is agentic, with procedural delegation available when a particular operation requires a fixed sequence. Through Integration Agent, Concierge can invoke a workflow and supply its inputs. The workflow executes the required steps and returns a result; Concierge uses that result to continue the conversation.

For example, an order-status operation may first call an order management API to retrieve item identifiers, then use an identifier in a shipping-status API call. The second call depends on the first. A procedural workflow preserves that order while Concierge decides when the operation is useful. A fixed sequence may also be appropriate where a business mandate requires it.

Authority starts with a positive definition of the job: which products Concierge supports, which policies apply, what information it may use, and which actions it may perform. Restrictions and handoff conditions clarify that assignment. Listing everything the system must never do has no natural stopping point; an explicit service responsibility gives unfamiliar requests a useful frame.

The controls work together, but they are not all consecutive gates. Scope and instructions shape the interaction throughout. Technical permissions constrain access and execution. Handoff is an alternative when proceeding is unsupported. An agentic conversation can remain flexible while a sensitive operation remains tightly constrained.

## SHAPE THE INTERACTION

Scope · policies ·  
focused skills

## CONSTRAIN EXECUTION

Access · permissions ·  
ordered workflows

## TRANSFER RESPONSIBILITY

Handoff when evidence or  
authority is insufficient

Understanding the request does not grant  
authority to execute it.





# Layered instructions, enforced permissions

A customer sees a short exchange, while the model may receive a substantial instruction package. In a typical Crescendo enterprise deployment, the behavioral prompt alone contained approximately 41,000 characters—roughly 10,000 tokens by estimate. Clear layers make that context easier to inspect, test, and maintain.

Model guardrails provide the foundation. Crescendo uses frontier models whose safety training and built-in protections influence behavior. The core system prompt adds Crescendo’s instructions for safety, security, unsupported answers, and resistance to prompt injection. These reduce risk; they are not substitutes for access enforcement.

The customer-specific behavioral prompt defines Concierge’s role, scope, policies, and handoff conditions. Enterprise customers can review this layer in the admin interface. Focused skills supply instructions for particular tasks in a subsequent model interaction, compartmentalizing sensitive or specialized behavior for more precise testing and improvement.

Authentication illustrates the distinction. An email address may locate a record without establishing permission to disclose it or modify an account. A skill can express the verification requirements for the requested action and guide the exchange. The integration must enforce the corresponding permissions even when conversational judgment is imperfect.

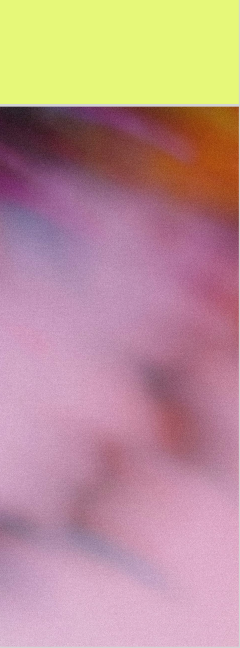
A handoff is also a guardrail. Concierge can transfer work when evidence is insufficient, a request exceeds scope, or a person must exercise retained authority. The objective is a sound customer outcome, not containment at any cost. Some transfers reveal fixable gaps; others are correct boundaries that should remain.

|                    |                                             |
|--------------------|---------------------------------------------|
| Focused skills     | Task-specific instructions                  |
| Behavioral prompt  | Customer role, scope and policies           |
| Core system prompt | Crescendo safety and security instructions  |
| Model guardrails   | Protections built into the foundation model |

Instructions describe the responsibility.  
Technical permissions enforce access and actions.







# See the interaction. Evaluate the outcome.

Once Concierge responds or acts, the enterprise needs evidence of what happened. A transcript records the conversation. Sources, recorded explanations, knowledge queries, tool requests, and returned results make it possible to investigate the path that produced it.

Crescendo exposes the retrieval question Concierge formulates for the knowledge base, not just the customer's original words. That distinction matters: a weak answer may begin with a poorly framed search. Action traces can reveal a different cause, such as incorrect parameters, an integration failure, or missing backend data. Recorded explanations add context without claiming complete access to the model's internal reasoning.

Quality Agent supplies a separate evaluative perspective. Crescendo's opposing-model approach means that the interaction is assessed against defined criteria rather than accepted as successful because Concierge produced it. Crescendo uses a different model to evaluate Concierge's interactions, adding another layer of safety through a separate assessment the response and its outcome.

Predictive CSAT assesses satisfaction at the message-response turn and conversation levels. Turn-level assessment helps locate deterioration; conversation scores and summaries describe the overall experience. Auto QA applies a deployment's custom rubric, including policy adherence, required verification, usefulness, and brand requirements, with scores and written explanations for individual criteria.

These signals answer different questions. A satisfied customer may have received an unauthorized concession. A policy-compliant response may still create unnecessary effort. Evaluation identifies those problems; turn-level scoring does not mean every assessment blocks a response before it reaches the customer.



Coverage matters because explicit feedback is sparse. Across four production chat deployments in Consumer Ecommerce, B2B SaaS, and Consumer Software, customers rated about 0.43 percent of Concierge responses—4.25 per 1,000. Crescendo’s predictive evaluation design targets 100 percent of turns and conversations within the configured scope, with Auto QA adding rubric-based assessment. Evaluation coverage is a standard configuration capability.

Coverage does not make every judgment correct. Human review and explicit feedback help check the evaluators, and missing or failed evaluations must remain visible. A consistent CSAT approach provides a shared signal; custom rubrics preserve business-specific standards. Comparisons still need context, including customer populations, request complexity, and changes to the evaluator.

Evaluation shows where service falls short. It does not replace controls on what the system may do.





# Testing, testing, testing

A deployment needs evidence before customers depend on it. It also needs a repeatable way to check established behavior whenever instructions, knowledge, capabilities, or models change.

Our early Crescendo deployments included roughly a month of user acceptance testing. People exhaustively exercised customer situations, investigated failures, and refined the deployment. We have since implemented automated testing in which AI creatively generates cases, runs them, and identifies behavior that needs improvement. The cases are stored as Evals.

Failures drive a recursive process of diagnosis, correction, and retesting. A test may reveal an instruction conflict, a missing capability, or a response that fails the business's requirements. Test cases themselves may need refinement when they express a requirement inaccurately. Human supervision keeps both the deployment and the test suite anchored to the intended standard.

The lasting value is persistence. From deployment forward, additions and modifications trigger a rerun of the full Evals suite. A revised instruction is checked alongside behaviors that already worked. A new capability is tested for unintended effects elsewhere. The organization retains a permanent set of regression tests—Evals, in the language of AI testing—maintained by AI with human supervision.

That supervision has a specific purpose: changing a test to reflect a corrected requirement is different from weakening the requirement to make a failing deployment pass. Expected behavior remains a business decision. The test suite should preserve that decision as the implementation evolves.

Evals and production evaluation complement each other. Evals exercise behavior before launch and after changes. Predictive CSAT and Auto QA assess real interactions, including situations the suite may not yet cover. Recurring failures and handoffs can inform additional tests, so operational experience becomes a more durable quality guardrail.

As we have automated more deployment and optimization work, we have seen quality improve. AI can examine interacting instructions and test alternatives with a thoroughness that is difficult to sustain manually. That is our operating observation, not a guarantee that every AI-generated change is better. Testing is how a proposed improvement earns its place in the deployment.

Every deployment builds a lasting test suite. Every change must respect what the business has already learned.



# Govern the agents that change the system

Governance cannot stop with Concierge. Quality Agent evaluates interactions; Analyst identifies patterns; Knowledge Agent, Integration Agent, and Optimization Agent contribute improvements. The agents changing the operating conditions need defined responsibilities just as the customer-facing agent does.

Optimization Agent can reorganize the behavioral prompt to improve quality. Its ability to prepare a better configuration does not confer authority to change business policy or expand access. People establish intent, resolve policy questions, and retain responsibility for consequential changes. Proposed improvements belong within those decision rights and must be tested against the deployment's requirements.

The same principle applies to knowledge. An expert resolution or customer transcript can suggest useful guidance, but it is evidence to validate rather than automatic authority to publish. Knowledge Agent can surface conflicts and help prepare revisions. The business must determine which policy applies; customer-specific details and one-time concessions should not silently become general instructions.

Crescendo has an established point of view on continuous knowledge improvement, a topic we explore separately. It is closely connected to guardrails and quality: existing materials, customer conversations, expert escalation, and validated updates can strengthen the knowledge that supports each interaction. The governing connection is simple. Better knowledge can expand what Concierge can answer reliably; it does not automatically expand what Concierge is permitted to do.



Crescendo versions the assistant and tracks behavioral-prompt changes. Updates generate release notes. Role-based access control determines who may make changes, while attribution identifies the users responsible. Together, these provide a reviewable history of how the deployment evolved.

That history supports investigation, but a timestamp alone does not prove that an update improved performance. Evals test the proposed change, and subsequent customer interactions provide evidence of its effects. Governance has to connect the recommendation, the authorized change, and the observed result.

The practical review questions extend across the agent suite: what may each agent change, what requires a person, what standard is tested, and who is accountable for release? Answering those questions makes human supervision an operating responsibility rather than a reassuring phrase.

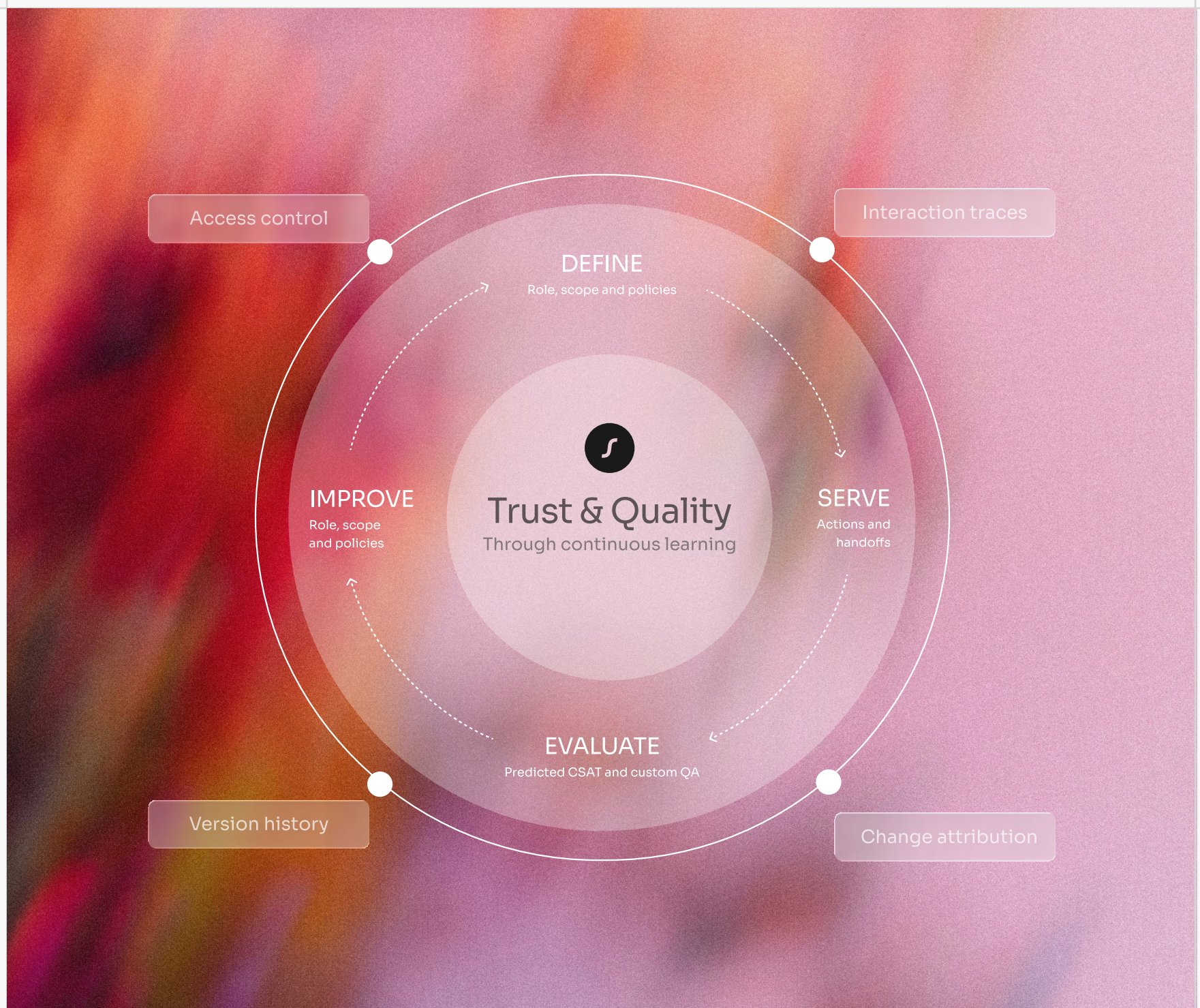
An agent’s ability to propose an improvement is separate from the authority to put it into service.





# A learning loop with accountable change

Consider an illustrative retailer whose Concierge repeatedly hands off partial-shipment questions. It can retrieve the order, but lacks reliable guidance for explaining item-level delivery status. The handoff protects the current customer from an unsupported explanation and produces a signal worth investigating.





Categories bring the conversations together. Turn-level evaluations help identify the point of failure. Summaries describe the unresolved need. Retrieval queries and tool traces show which information was sought and what the backend returned. The team can distinguish missing guidance from missing data or a decision deliberately reserved for a person.

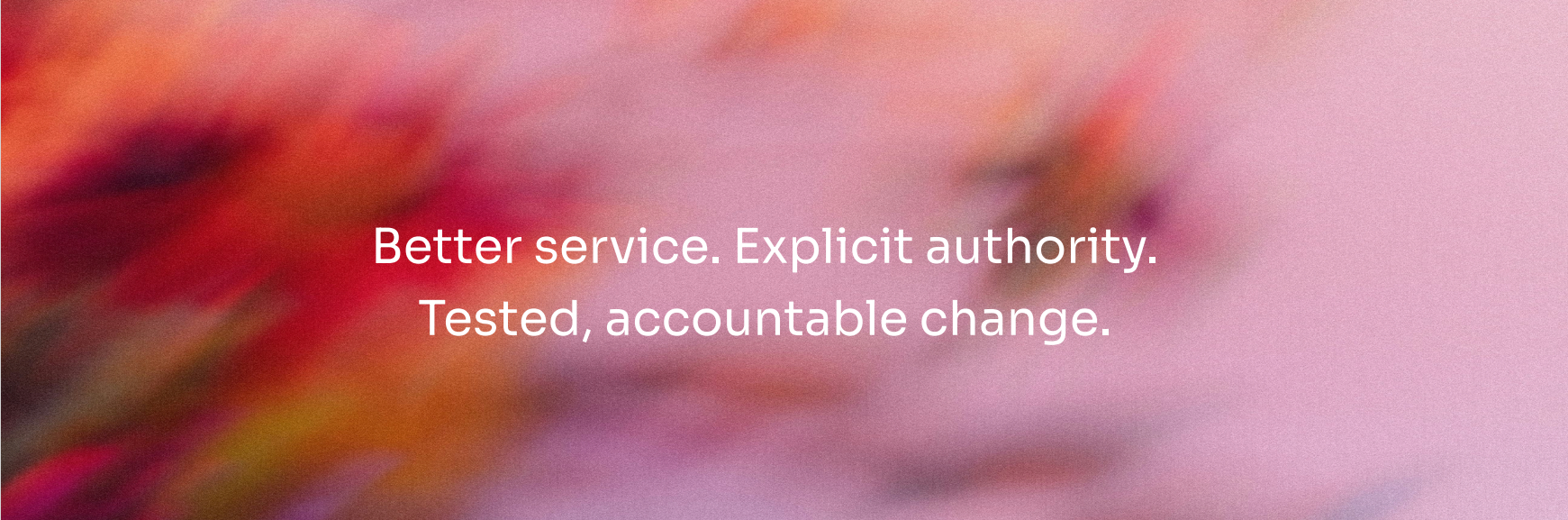
Each cause implies a different response. Knowledge Agent may help prepare clearer guidance. Integration Agent may be needed to expose missing information. Optimization Agent may improve conflicting instructions. A policy question belongs with the business. Some handoffs should remain; improvement does not mean removing every boundary.

Once the appropriate change is authorized, the persisted Evals suite checks the correction and established behavior. The assistant's version record and generated release notes document the update. Later interactions show whether resolution, satisfaction, and rubric adherence improved. A successful correction becomes part of the conditions for the next interaction.

That is the connection between runtime guardrails and continuous improvement. Boundaries protect the interaction. Traces and evaluations expose what happened. Tests and decision rights govern what changes. Learning occurs through better instructions, knowledge, skills, and integrations; the underlying model does not need to retrain after every conversation.

The enterprise should expect stronger AI to help maintain these controls as well as serve customers. But capability does not replace accountability. People define business intent and consequential authority; technical controls constrain execution; evidence determines whether proposed improvements work.

Trust grows when a business can explain what the system may do, see what it actually did, and improve it without losing control of either. That is the standard for a governed learning loop—and the foundation for customer experience that becomes more capable over time.



Better service. Explicit authority.  
Tested, accountable change.



# Schedule a demo

A handoff isn't a failure — it's a signal. Crescendo turns every unresolved conversation into evidence for what to fix next, then tests every proposed change against a permanent suite before it touches production. The result is a system that gets more capable over time, without ever drifting outside its authority.

Set up a personalized demo with us. We'll show you how your support data can power a governed learning loop — better service, tested and accountable at every step.

## Book a demo today.

[crescendo.ai/demo](https://crescendo.ai/demo)

C R E S C E N D O