Agentic RAG for CRM: From Answers to Actions

Author(s)
Abinaya Mettuaptti Sivagnanam*, Srikanth Singireddy, Vinoth Asoor Palanivel

***Abstract*—Many CRM teams need more than a text answer. They also need the agent to take the next step. We offer a hands on blueprint for an agent that writes a grounded reply and performs a safe action in the CRM. The design is small on purpose: a compact action record with a signature, a policy guard for roles, limits, and required fields, and a simple split between edge and cloud for speed and privacy. An illustrator style diagram (illatrater) follows one case end to end, from a customer request, to retrieved policy, to an approved update with an audit receipt. We close with light weight metrics and a staged roll out plan that a small team can try, measure, and refine.**

***Index Terms*—Retrieval Augmented Generation (RAG), Agentic AI, CRM Automation, Policy Guard, Auditable Actions, Edge and Cloud.**

I. Introduction

Customer service teams often ask large language models for help. The model gives an answer. In many cases that is not enough. Real work needs an action inside the CRM. Someone must update a case, send a reply, escalate a ticket, or issue a credit. We give a practical blueprint that turns answers into trusted actions with a retrieval step and a small set of safety checks [1], [2].

A. Problem and motivation

If an assistant acts without rules, it can break policy, show private data, or leave no audit trail. If it only answers, a human still has to do the work. We need a middle path. The assistant should plan with retrieved facts, follow clear rules, and then perform a safe action that we can audit later [3], [4].

B. What the system offers

This is a hands on guide, not a report of a large experiment. The value is a small pattern that teams can reuse.

1) **Signed action schema.** A short JSON record for each action with a reason and citations. We also add a signature so we can verify who proposed it [4].
2) **Policy guard.** Simple allow and deny rules with checks for role, field level edits, refund caps, required fields, and redaction of non participant personal data [3].
3) **Edge and cloud placement.** Fast and private steps run near the data at the edge. Training and analytics run in the cloud [5].
4) **Audit by design.** Every proposal, decision, and result gets an id and a receipt so we can review it later.

An illustrator style diagram that we call an *illatrater* shows the end to end flow from a user request to a policy checked action and a receipt.

TABLE I: Example CRM action types and impact

Action	Baseline time	With blueprint	Risk
Auto reply from a policy template	6.0 min	1.5 min	Low
Case field update	4.0 min	1.0 min	Low
Escalation with checks	8.0 min	3.0 min	Medium
Refund request with a cap	12.0 min	5.0 min	Medium
Data redaction task	10.0 min	2.5 min	Low

Values are illustrative only.

C. Paper map

Section II shows the architecture with the illatrater. Section III gives a step by step protocol and the signed action schema. Section IV covers security, privacy, and governance. Section V lists simple metrics and a roll out plan. Section VI notes limits. Section VII closes the paper.

II. Architecture

This section shows how the parts fit together. The goal is a small design that real CRM teams can build and maintain. Fig. 1 gives an illatrater of the full flow. Fast and private steps stay near the data at the edge. Training and governance live in the cloud [2], [3], [4].

A. Main parts

1) **User interface.** A chat view or form where the user states the need.
2) **Agent.** Plans the steps, reads retrieved text, drafts a reply, and builds a signed action with a reason and citations.
3) **Retriever.** Finds short snippets from the knowledge base, policy pages, and case history that ground the plan.
4) **Policy guard.** Runs clear allow and deny rules and small checks such as role, field level edit, refund cap, and required fields [3].
5) **CRM hook.** A narrow write API that runs only approved actions and returns a receipt for audit [4].
6) **Audit and observability.** Stores the plan, the decision, the signature, and the result with time and id.
7) **Key management.** Issues and rotates keys that sign and verify actions.

B. Edge and cloud placement

Edge. Retrieval on sensitive data, agent planning, action proposal and signing, policy guard checks, CRM hook writes.
Cloud. Model training, evaluation jobs, analytics, model and policy registry, and governance dashboards [5].
This split lowers data movement and improves time to first reply.

Fig. 1: Illatrater of the path from user request to a checked action and an audit receipt. Edge runs fast and private steps. Cloud holds training and governance.

C. *Data and control flow*

The agent reads evidence, proposes a signed action with citations, the policy guard checks rules, and the CRM hook executes only if allowed. Each step writes to the audit log with a stable id for later review.

D. *Signed action record*

A compact JSON record helps with trace and review. The guard verifies fields and the signature [4].

Listing 1: Signed action record

```
{
  "action_type": "update_case",
  "target": {"case_id": "C-874213"},
  "params": {"field": "Status", "value": "Resolved
    "},
  "justification": "Per KB-REF-14 S3.2; customer
    confirmed fix.",
  "citations": ["KB-REF-14#S3.2", "CASE:C-874213"],
  "risk": "LOW",
  "proposed_by": "agent@edge-node-7",
  "timestamp_utc": "2025-09-06T14:22:11Z",
  "signature": "BASE64_ED25519_SIGNATURE"
}
```

E. *Interfaces at the edge*

Policy guard API

Listing 2: Policy guard API

```
POST /policy/check
Body: { "action_payload": <JSON>, "user_role": "
    CSR_L2" }
Resp: { "decision": "ALLOW"|"DENY",
        "reasons": ["refund_cap_ok", "pii_redacted
    "],
        "obligations": ["supervisor_approval"] }
```

CRM hook API

Listing 3: CRM hook API

```
POST /actions
Body: { "action_payload": <JSON>, "approval_token":
    "..." }
Resp: { "status": "EXECUTED", "action_id": "ACT
    -009812",
        "receipt_hash": "SHA256:..." }
```

F. *What this layout delivers*

Sensitive reads and rule checks stay close to the data. That makes private use safer and the user wait shorter. The signed action and the receipt make later review simple. Teams can add new action types by extending the schema and by adding one small rule to the guard. The overall pattern stays the same [2], [3].

III. Protocol

This is a hands on guide. Follow these steps to build the agent from Fig. 1. Each step is small and easy to test. The goal is a clear path from a question, to a grounded answer, to a safe action that we can audit [2], [1], [3], [4].

A. *P0. Setup*

Make sure four parts are ready. First, a retriever over knowledge pages, policy notes, and case history. Second, a policy guard service that can run allow and deny rules. Third, a narrow write API for safe actions. Fourth, a key store for signing and for verification [4].

B. *P1. Action record*

Use the compact schema from Listing 1. Keep the record small. Include the target, the parameters, a clear reason, and two citations. Sign the record so you can verify who proposed it later.

C. *P2. Prompts at the edge*

Keep prompts short and return simple fields. This makes review and testing easy.

Listing 4: Retrieval prompt

```
Task: Gather 5 to 7 short snippets that answer the
    user need and the policy checks.
Focus: refund caps, entitlement, service level, past
     case notes.
Output: JSON array of {source_id, text, span}. Keep
    spans tight.
```

Listing 5: Action proposal prompt

```
You are an enterprise agent. Propose one action or
    say NONE.
Use the fields from the signed action schema only.
Write a short reason in plain English. Add at least
    two citations.
Set risk to LOW, MEDIUM, or HIGH based on evidence.
If unsure or unsafe, output NONE.
```

Listing 6: User reply prompt

```
Write a short reply in simple and respectful tone.
    Limit to 100 words.
Explain the policy basis in plain English. Do not
    include personal data.
If an action was executed, add the receipt id in the
     last line.
```

D. *P3. Guard rules*

Start with a tiny rule set. Check refund caps. Check the user role and the field that can be edited. Check that required fields are present. Check that redaction passed. Use a two person approval for high risk. Deny by default when a rule is missing or the case is unclear [3].

Listing 7: Policy guard rules

```
rules:
  - id: refund_cap_basic
    when: action_type == "issue_credit"
    allow_if: params.amount <= 50
    else: require: ["supervisor_approval"]

  - id: role_field_check
    when: action_type in ["update_case", "send_reply
    "]
    allow_if: user_role in ["CSR_L2", "Supervisor"]

  - id: redact_non_participant_pii
    precheck: redact(non_participant_pii) == true

  - id: high_risk_two_person
    when: risk == "HIGH"
    require: ["two_person_rule"]
```

E. P4. Orchestration

The control flow on the edge is short and clear.

Listing 8: Edge orchestration

```
def handle_request(user_msg: str, user_role: str):
    need = parse_intent(user_msg)
    evidence = retrieve(need, k=7)              # KB,
    policy, case history
    draft = answer_with_citations(user_msg, evidence
    )
    action = propose_action(need, evidence)    # or
    None
    if action is not None:
        signed = sign(action, key="edge-key-1")
        decision = policy_guard.check(signed,
    user_role)
        if decision["decision"] == "ALLOW":
            receipt = crm_hook.execute(signed,
    decision.get("approval_token"))
            log_audit(user_msg, evidence, draft,
    signed, decision, receipt)
            return draft + f"\n\nAction: executed ({
    receipt['action_id']})"
        else:
            log_audit(user_msg, evidence, draft,
    signed, decision, None)
            return draft + "\n\nAction: not executed
     (" + ", ".join(decision["reasons"]) + ")"
    else:
        log_audit(user_msg, evidence, draft, None,
    None, None)
        return draft
```

F. P5. Placement

Run private work near the data at the edge. That includes retrieval, prompts, action proposal and signing, guard checks, and the write call. Run training, evaluation, and analytics in the cloud [5]. This keeps data movement low and the user wait short.

G. P6. Defaults for a first pilot

Use small and safe starting points. Replace these values after the pilot.

TABLE II: Illustrative defaults for a first pilot

Parameter	Default	Reason
Retrieval k	7	Balance recall and time
Max reply length	100 words	Clear and easy to read
Refund cap without supervisor	$50	Fast and low risk
High risk timeout	2 min	Encourage human review
Action time target p95	2.0 s	Good experience at edge
Audit retention	365 days	Trace and compliance

Values are illustrative only.

H. P7. Small test harness

Create ten to twenty seed cases across update, refund, and escalate. Provide one short page of knowledge and one short policy note for each case. Run the pipeline. Record four things for each run. First, the citations. Second, the guard decision. Third, the action id or the denial reason. Fourth, the total time. Manually grade a small sample for correctness and policy fit.

I. P8. Safety by default

Redact before planning. Deny when a rule is missing or unclear. Use two person approval for high risk or for actions that you can not undo. Sign all action proposals and store the receipts.

J. P9. Why this helps

The pattern is small and clear. A signed action record, explainable guard rules, and edge first placement give speed, privacy, and governance together. The same flow works for many action types in common CRMs [4], [3], [5].

IV. Security, Privacy, and Governance

This section explains how we keep actions safe, private, and easy to review. The aim is simple. Do only what is allowed. Hide what must be hidden. Leave a clear trail that others can trust [3], [4].

A. Threat model in plain words

We assume three kinds of risk. First, honest agents who can make mistakes. Second, insiders with limited roles who may try to see more than they should. Third, outside attackers who try prompt injection, data theft, or forced actions. Our answer is a small stack of controls. Redact first. Apply least privilege. Deny by default when unsure. Ask a human to approve high risk steps [3].

B. Privacy by redaction and by minimization

Redact first. Then plan. Then act. The agent removes non participant personal data from retrieved text before it drafts a reply or proposes any action.

Listing 9: Illustrative redaction code on the edge

```
EMAIL = r"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z
    ]{2,}"
PHONE = r"\+?\d[\d -]{7,}\d"
SSN   = r"\b\d{3}-\d{2}-\d{4}\b"

def redact(text: str) -> str:
    text = re.sub(EMAIL, "[EMAIL]", text)
    text = re.sub(PHONE, "[PHONE]", text)
    text = re.sub(SSN,   "[SSN]",   text)
    return text
```

Note. In real use, call your enterprise data loss prevention library. The code in Listing 9 is only an illatrater.

C. Least privilege and small scopes

Write access is narrow. Only a few fields and objects can change. Tokens are short lived and allow listed by object and by field [3].

Listing 10: Write allow list

```
write_allowlist:
  Case:
    fields: ["Status", "Priority", "Refund_Amount__c
    "]
    actions: ["update_case", "escalate", "
    issue_credit"]
  Message:
    actions: ["send_reply_template"]
```

D. Policy guard with allow and deny and pre checks

The guard returns ALLOW or DENY. It uses clear rules and pre checks such as user role, field limits, refund caps, required fields, and redaction status. It can also add an obligation that a human must approve a step [3]. See Listing 7 for a small rule set.

E. Signing and receipts

Each proposed action is signed on the edge. The CRM verifies the signature and returns a receipt that we store in the audit log [4].

Listing 11: Execution receipt

```
{
  "action_id": "ACT-009812",
  "decision": "EXECUTED",
  "receipt_hash": "SHA256:bf1b...7a",
  "verified": true,
  "ledger_index": 421337
}
```

F. Audit and observability

Every step writes to an append only log with time and id. Store the signed payload, the rules that fired, any approvals, and the CRM response. Keep simple dashboards for teams that review events.

G. Human review when needed

High risk or one way actions need two person approval. The guard can add the obligation `supervisor_approval`. If the system can not satisfy an obligation, it must not execute the action.

H. Edge and cloud governance

Edge holds private work. That includes retrieval, planning, guard checks, and writes. Cloud holds model registry, evaluation jobs, analytics, and policy governance. This split cuts data movement and keeps time to reply low [5].

I. Safety and privacy goals with sample values

Use these numbers only as starting points. Replace them with pilot data later.

TABLE III: Illustrative safety and privacy goals at p95

Metric	**Target**	**Comment**
Redaction latency	$\leq$ 200 ms	One pass before planning
Policy check latency	$\leq$ 150 ms	Rule engine and small lookups
Action time end to end	$\leq$ 2.0 s	Full path on the edge
Redaction recall	$\geq$ 99%	Test corpus for personal data recall
Guard false allow rate	$<$ 0.5%	Manual spot checks
Audit write success	100%	Retries with a queue

Values are illustrative only.

J. Policy tests and simple gates

Test rules often. Add unit tests for the guard and a small set of known risky prompts. Block a release if redaction recall drops, if rule coverage is low, or if any unsigned action appears in smoke tests.

Listing 12: Policy and redaction gates

```
assert redaction_recall(testset) >= 0.99
assert policy_coverage(rule_suite) >= 0.95
assert unsigned_actions_seen == 0
```

K. Incidents and a simple stop control

If metrics drift or a rule fails, auto disable the risky action and send the work to a human only path. A small stop control at the guard makes rollback fast.

Listing 13: Guard stop control

```
POST /policy/disable?action_type=issue_credit
```

L. Why this helps a reviewer

The plan is practical. We redact before we plan. We use clear allow and deny rules with small pre checks. We sign actions and store receipts. We set simple and testable goals. Together these steps give speed, privacy, and trust [3], [4].

V. Evaluation and Rollout

This section shows how to measure the blueprint and how to release it with care. We use a few clear KPIs, a small pilot, and growth in stages. The numbers here are examples. Replace them with your own results [1], [3], [4], [5].

A. What to measure

Track both technical and business outcomes.

a) Technical:

- Grounding quality. Share of answers with correct cites.
- Guard correctness. False allow and false deny rates.
- Latency. p50 and p95 from user request to action receipt.
- Redaction coverage. Recall on a held out test set.
- Audit completeness. Share of actions with a signed record and a receipt hash.

b) Business:

- Handle time in minutes per case.
- First contact resolution in points.
- Denied risk events per one thousand cases.
- User satisfaction score.

B. Scorecard with sample values

Table IV gives an example scorecard for a pilot of four to six weeks. Use it as a guide only.

TABLE IV: Pilot scorecard with targets and observed values

Metric	Target	Observed	Pass
Grounded answers (%)	≥ 95	96.8	Yes
Guard false allow (%)	< 0.5	0.3	Yes
Guard false deny (%)	≤ 3.0	2.6	Yes
Latency p95 (s)	≤ 2.0	1.7	Yes
Redaction recall (%)	≥ 99	99.2	Yes
Audit complete (%)	100	100	Yes
Handle time change	30% lower	27% lower	Near
First contact resolution lift (pts)	+5	+6	Yes

Values are illustrative only.

C. Offline tests before users

Run small tests that are easy to repeat [3].

1) Golden set of twenty to fifty prompts with known good answers and action labels.
2) Policy cases that check caps, roles, and required fields.
3) Red team prompts that try jailbreaks, personal data leaks, or bad escalations.
4) A personal data corpus to measure redaction recall and precision.

Report grounding accuracy, guard error rates, and latency under load.

D. Shadow mode with live data

Run next to real agents but do not execute actions. Log the proposed actions, the guard decisions, and the latency. Compare the suggested outcomes with human outcomes for one to two weeks.

E. Small canary

Turn on execution for a small group such as ten percent of low risk queues. Keep deny by default rules on.

- Guard false allow below 0.5 percent over at least one thousand actions.
- p95 latency at most 2.0 seconds on the edge path.
- Zero unsigned actions in the audit log.

F. Sample calculation for canary size

Let N be the number of observed actions in the canary. To estimate a false allow rate p within plus or minus 0.2 percent at 95 percent confidence, a common rule of thumb is

$$N \approx \frac{1.96^2\, p(1-p)}{(0.002)^2}.$$

With $p = 0.005$ you get $N \approx 4800$. Use this as a guide and adjust to your risk level.

G. Edge and cloud check

Measure the value of the placement choice [5].

- Edge inference on. Expect p95 latency about 1.6 to 2.0 seconds.
- Edge inference off. Cloud only path. Expect p95 latency about 2.8 to 3.5 seconds.
- Privacy gain. Fewer cross boundary reads and simpler sign off by compliance.

H. Rollout plan with stage gates

1) S0. Offline pass. Golden set at least 95 percent grounded. Redaction recall at least 99 percent.
2) S1. Shadow mode for one to two weeks. Tune rules. No writes to production.
3) S2. Canary at ten percent. Execute only low risk. Guard false allow below 0.5 percent.
4) S3. Expand to fifty percent. Add medium risk with two person approval.
5) S4. Full rollout. Keep dashboards, weekly reviews, and the stop control.

I. Dashboards and alerts

Keep one view with the key signals.

- Latency p50 and p95, guard errors, and redaction coverage.
- Executed and denied actions by type.
- Unsigned action detector that must stay at zero.
- Incident log with links to receipts and signatures.

J. Weekly KPI snapshot with sample values

TABLE V: Weekly KPI snapshot with sample values

KPI	Week 1	Week 2	Week 3
Latency p95 (s)	2.1	1.9	1.7
Grounded answers (%)	94.5	95.9	96.6
Guard false allow (%)	0.45	0.32	0.28
Guard false deny (%)	3.4	2.9	2.6
Audit complete (%)	100	100	100
Handle time (min)	5.8	4.6	4.2

Values are illustrative only.

K. Risk drift and stop control

If a KPI drifts, for example false allow at or above 0.5 percent for two days, auto disable the risky action and send the work to a human only path. Use the same stop control from Listing 13. Turn it back on only after a rule fix and a short shadow mode check.

VI. Limitations and Threats to Validity

This is a blueprint. It shows how to build and measure a safe agent that can act in a CRM. It is not a large multi site trial. Results will depend on data quality, policy strength, and how your team wires the system into the stack.

A. Internal validity

- **Retriever drift.** When knowledge pages or policy notes change, grounding quality can drop.
- **Prompt injection.** Clever inputs can push the agent off task if filters are weak [3].
- **Rule gaps.** If a rule is missing, an edge case can slip through.
- **Key practice.** Weak key storage or slow rotation lowers trust in signed actions [4].
- **Race conditions.** A record can change after a check and before a write.
- **Edge limits.** Small edge nodes can run out of memory or time and slow the path.

B. External validity

- **Portability.** Object names and fields differ across CRMs, so some actions may not map one to one.
- **Rules by region.** Privacy and audit law varies by country and by industry.

C. Mitigations in practice

Use small checks early and grow from there. Nightly index rebuilds reduce drift. Deny by default when a rule is unclear. Keep a short red team checklist. Store keys in a vault and rotate on a fixed schedule. Re validate before commit. Set a p95 budget for end to end time and add autoscale if needed.

VII. Conclusion

We shared a clear and small pattern that turns answers into trusted actions in a CRM. The steps are direct. Ground the plan with retrieval. Propose a signed action with a short reason and two cites. Check policy with allow and deny rules plus small pre checks. Execute only if allowed and store a receipt. Keep fast and private work near the data at the edge. Keep training and long running review in the cloud [5].

The aim is to help teams adopt this with low risk. The paper gives prompts, a compact action record, a few rules, and simple goals for time and safety. The illatrater shows the end to end path from a user need to an audited action. Future work can add cross vendor adapters, richer policy checks, and small pilot reports that show change in time to resolve and in user satisfaction.

References

[1] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," 2023. doi: 10.48550/arXiv.2312.10997.

[2] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," NeurIPS, 2020. doi: 10.48550/arXiv.2005.11401.

[3] Z. He *et al.*, "Building Guardrails for Large Language Models," 2024. doi: 10.48550/arXiv.2402.01822.

[4] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," 2017. doi: 10.17487/RFC8032.

[5] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017. doi: 10.1109/MC.2017.9.

[6] C. Ledro, A. Nosella, and A. Vinelli, "Artificial Intelligence in Customer Relationship Management: Literature Review and Future Research Directions," *Journal of Business & Industrial Marketing*, vol. 37, no. 13, pp. 48–63, 2022. doi: 10.1108/JBIM-07-2021-0332.