

**Expiration-Aware Enterprise AI:
Keeping Answers Aligned with Changing Policies**

Sruthi Rachamalla

Southern Illinois University, Carbondale

ORCID iD: 0000-0001-9312-507X

Abstract

Background: An enterprise assistant can retrieve a relevant policy and still provide the wrong answer if the rule no longer applies. Publication dates, effective dates, regional exceptions, and stored summaries can point to different versions of the same instruction.

Methods: This paper specifies a conceptual architecture that treats policy applicability as a condition for releasing an answer. It separates the date governing a business event from the date information entered the system, records dependencies between policies and derived content, and rechecks evidence before delivery.

Results: The proposed workflow combines an authoritative policy register, temporal retrieval, dependency tracking, and a release gate. A fictional reimbursement example illustrates future-effective changes, historical questions, and outdated memories. An evaluation protocol compares these controls with similarity-based retrieval and periodic refresh.

Conclusion: Reliable policy assistance requires more than frequent indexing. Claims need applicable evidence and interpretation checks; reused answers need valid dependencies. The architecture remains unvalidated; its effectiveness depends on accurate metadata, complete change detection, and clear policy ownership.

Keywords: enterprise AI; temporal retrieval; policy governance; cache invalidation; knowledge provenance

1 Introduction

A policy assistant can quote the right paragraph and still give the wrong advice. Consider an employee submitting a March travel claim in April. The newest reimbursement policy may govern April expenses, not the earlier trip. Selecting the latest document would therefore be a mistake.

Temporal retrieval research separates topical relevance from time constraints [1], while recent work examines amendments that change individual clauses without replacing an entire policy [2]. The operational challenge extends to summaries and saved answers that outlive their supporting evidence. This paper proposes a framework connecting policy applicability, dependent knowledge, and answer release.

2 Methods

The method is a conceptual design with a fictional workflow and a proposed evaluation, not a deployment study. The design assumes authenticated and authorized policy owners, versioned source documents, and a managed answer-delivery service.

Each policy clause carries an identifier, owner, scope, effective interval, recorded timestamp, version, and source location. Scope includes jurisdiction, employee group, and relevant exceptions. Effective time describes when the rule governs an event; recorded time describes when the system learned it. Temporal memory research also distinguishes validity from accumulated retrieval history [3]. Corrections preserve both histories rather than silently overwriting earlier records.

A query must identify the governing event, relevant date, and scope. A historical entitlement question uses the rules applicable to that event, including authorized retrospective corrections. An audit question asking what the assistant knew then additionally restricts the recorded-time view. Missing information prompts clarification; neither the current date nor the newest upload is a safe universal substitute.

Eligibility requires all relevant conditions: authorized access, approved source, matching scope, applicable interval, and resolved exceptions. Similarity cannot compensate for a failed condition. Intervals use an inclusive start and exclusive end in the policy’s stated time zone. An unspecified end date means no scheduled expiration, not immunity from withdrawal or later correction.

3 Results

3.1 Register changes and preserve dependencies

Figure 1 separates policy maintenance from answering. Owners approve clause-level changes before publication. The register preserves supersession and exception relationships. A conflicting upload cannot replace an approved rule merely because it arrived later. Unresolved conflicts go to the responsible owner.

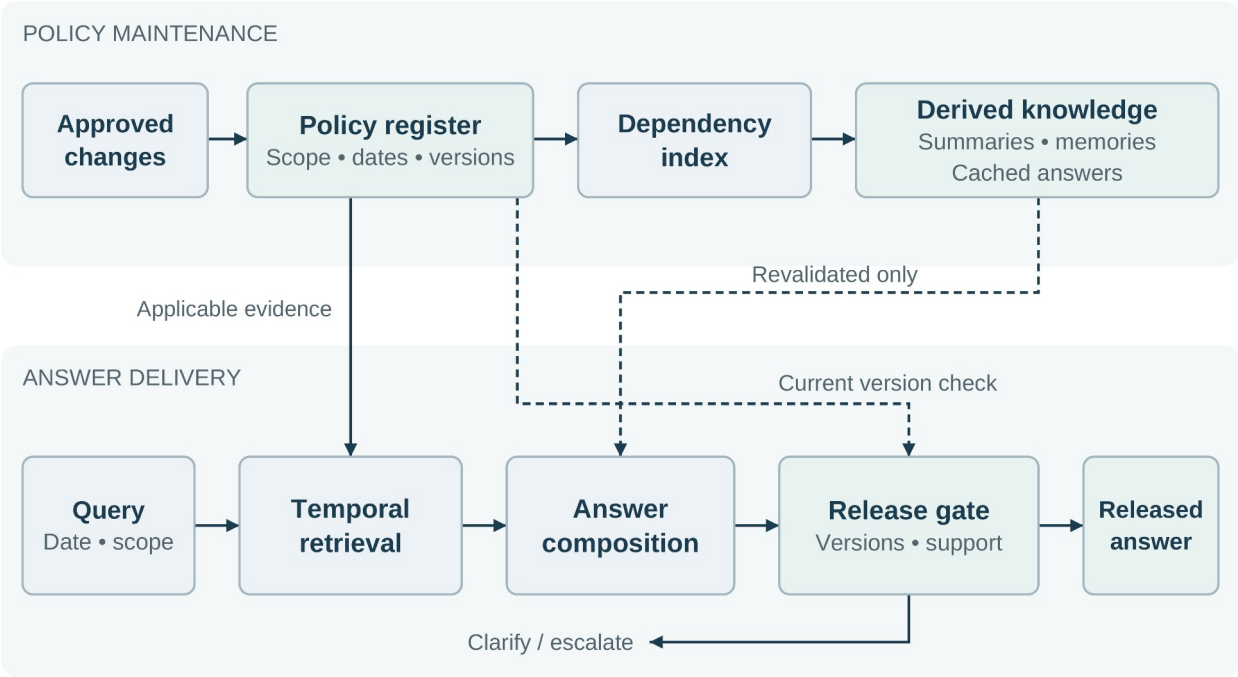


Figure 1. Policy changes trigger revalidation; release checks applicability, versions, and interpretation.

The dependency index links source passages to summaries, memories, and cached answers. Changes trigger revalidation along indirect links, including summaries of summaries. Preserve

historical versions for authorized audits. Where links are incomplete, conservatively invalidate the affected policy collection.

Change notifications carry stable event identifiers and ordered policy versions, allowing duplicate delivery without repeating updates. Periodic reconciliation checks for missed notifications, withdrawn documents, and access changes. Partial amendments invalidate the affected clauses and their dependents; they do not automatically retire unrelated provisions in the same document.

3.2 Retrieve applicable evidence and control reuse

Retrieval first restricts evidence to authorized scope and applicable intervals, then ranks relevance. Approved exceptions follow explicit precedence rules. Answers cite clause, version, and applicable date. Applicability checks do not establish correctness: check material statements against selected clauses and escalate unsupported interpretations. Unresolved contradictions prevent release. Access checks apply equally to original sources and derived content.

Cached answers carry the query scope, temporal mode, evidence versions, and dependency identifiers. HTTP caching distinguishes freshness from validation [4]; policy validity requires additional business rules. A time-to-live limits reuse but cannot prove applicability. Scheduled effective dates trigger revalidation even when no document changes that day.

Before delivery, the gate compares evidence versions with the register. Changes require reevaluation. Consequential answers require current source confirmation or escalation: the register cannot reveal updates it has not received. The verification time records the knowledge boundary, not permanent validity.

For a current-rule answer, the register version check and release authorization should share one consistent transaction where feasible. A bounded retry restarts evaluation after a concurrent change; repeated changes trigger escalation. Historical and current answers use distinct cache entries, even when their question wording is identical.

3.3 Worked reimbursement example

Assume a fictional policy allows USD 60 per day through March 31, 2026. A replacement published March 10 applies USD 75 from April 1, based on expense date. Table 1 shows expected behavior, not measured results. A regional exception remains independently applicable within its approved scope.

Table 1. Expected handling of illustrative policy changes.

Situation	Applicable evidence	Expected response
March expense; April query	Rule effective in March	Apply USD 60.
April expense; March 20 query	Published future rule	Published rule: USD 75 from April 1; recheck before expense.
Expense date missing	Applicability unresolved	Ask for the date.
Covered regional exception	Approved scoped clause	Apply the exception.
Retrospective correction	Owner-approved amendment	Reassess affected claims.

If an answer about an April expense still relies on a USD 60 summary, its dependency check blocks reuse. The assistant retrieves applicable evidence and rebuilds the answer. It does not delete the earlier rule, which still supports March claims.

4 Discussion

Evaluation should replay policy updates in recorded order, withholding future records at each test point. Compare similarity-only retrieval, periodic refresh, temporal filtering, and the full dependency-and-release design using identical questions, source histories, and generation

settings. Policy specialists should independently label governing clauses, effective dates, and expected exceptions.

Measure wrong-version answers among answered questions alongside answer coverage, historical accuracy, unnecessary abstention, update-to-invalidation delay, and latency. Include changes during generation, missed notifications, partial amendments, revoked access, and retrospective corrections. Report error counts and uncertainty; lower error rates achieved by refusing nearly every question do not establish improvement.

Policy ownership and continuing monitoring need explicit responsibilities [5]. A practical rollout begins with one policy family and an accountable owner. The main limitations are incomplete metadata, missed dependency links, and delayed source publication. No retrieval architecture can resolve an organizational disagreement about which rule is authoritative. Such cases require a documented decision, not a more confident answer.

The operational record should retain the question's temporal interpretation, source identifiers, dependency versions, release decision, and subsequent corrections. These records support targeted notification when an earlier answer is affected. Retention and access limits must still apply; auditability does not justify keeping sensitive employee information indefinitely.

5 Conclusion

Expiration-aware assistance treats every answer as a time-bound interpretation of authoritative evidence. Keeping policies, memories, and cached responses aligned requires coordinated change handling and explicit temporal context. The proposed design makes those obligations inspectable; controlled evaluation is needed before claiming improved reliability.

References

- [1] S. Zhang, Y. Xue, Y. Zhang, X. Wu, A. T. Luu, and C. Zhao, “MRAG: A Modular Retrieval Framework for Time-Sensitive Question Answering,” in Findings of the Association for Computational Linguistics: EMNLP 2025, pp. 3080–3118. doi: 10.18653/v1/2025.findings-emnlp.167.
- [2] Y. Nam, J. Kim, H. Song, S. Yoon, J.-G. Lee, and B. S. Lee, “TimelyRAG: Semantic-Temporal Hybrid Retrieval for Time-Critical Question Answering in Overlapping-Evolving Documents,” in Findings of the Association for Computational Linguistics: EMNLP 2026, pp. 8414–8430. [Online]. Available: <https://aclanthology.org/2026.findings-emnlp.398/>
- [3] N. Yadav, “Temporal Validity in Retrieval Memory: Eliminating Stale-Fact Errors for AI Agents over Evolving Knowledge,” arXiv preprint arXiv:2606.26511, 2026. doi: 10.48550/arXiv.2606.26511.
- [4] R. Fielding, M. Nottingham, and J. Reschke, Eds., “HTTP Caching,” RFC 9111, Jun. 2022. doi: 10.17487/RFC9111.
- [5] E. Tabassi, “Artificial Intelligence Risk Management Framework (AI RMF 1.0),” NIST AI 100-1, Jan. 2023. doi: 10.6028/NIST.AI.100-1.