CHANDRA

THE CONTEXT UNIT PROTOCOL

Perpetual organizational memory for every human decision and every agent action.

MIT LICENSE · v17.0 · GENERAL REASONING, INC.

Auditable Authorization:

RBAC2 on an Append-Only Chain

The structural audit gap in conventional access control, and its closure.

This paper makes a precise claim: the standard RBAC2 rights model, when its authoritative record is an append-only hash-chained log, closes a structural audit gap that conventional RBAC implementations accept as unavoidable. The closure is not procedural. It is architectural.

General Reasoning, Inc. · May 2026

Contents

1. The Honest Framing
2. What NIST RBAC2 Actually Specifies
3. The Structural Audit Gap
4. Closure by Construction
5. The Objects
5.1 The Mutable Rights Store Gap
5.2 A Grant Sequence as Chain Units
5.3 The Authorization Decision as a Chain Unit
5.4 Separation of Duty -- The Denial is an Event
5.5 State Reconstruction at Arbitrary Time
5.6 Provisioning Evidence for CC6.3
6. Design Principles Named
7. Honest Limitations
8. Conclusion

1. The Honest Framing

This paper does not claim that RBAC2 is new, controversial, or underappreciated. NIST ANSI/INCITS 359-2004 is the canonical access control standard. It is what SOC 2 auditors expect to see documented. It is the model underlying AWS IAM, GCP IAM, and Azure RBAC. Choosing RBAC2 is not a design decision worth a paper.

What is worth a paper is a structural defect in every conventional RBAC2 implementation that is accepted without comment, papered over with compensating controls, and treated as unavoidable. It is not unavoidable. The defect is this: the rights record is mutable, and the audit log is separate.

In a conventional implementation, an access control decision at time T is made against a rights store that reflects the current state of assignments. If assignments change after T -- a role is granted, revoked, or modified -- the rights store no longer contains the state that existed at T. The log may record the change. The log may have been rotated, truncated, or stored in a system with different retention policies than the rights store itself. At audit time, reconstructing what a principal was authorized to do at an arbitrary past point requires correlating a mutable store against an external log -- if the log exists, if it is complete, and if the correlation is trustworthy.

SOC 2 auditors accept this situation because there is no structural alternative in conventional implementations. Compensating controls -- periodic access reviews, log retention policies, ticketing system integration -- reduce the risk but do not close the gap. The gap is architectural. The closure must therefore be architectural.

Core Claim

When the authoritative record for an RBAC2 implementation is an append-only hash-chained log -- a Chandra chain -- the structural audit gap is closed by construction. Every grant, every revocation, every authorization decision, and every denial is a permanent, hash-linked, replay-capable unit in the chain. The rights store is not a mutable table that reflects current state. It is a derived projection of an immutable event sequence. Current state can always be reconstructed. State at any past point can also be reconstructed. The audit record is not separate from the rights record. They are the same record.

This paper describes how that closure works, demonstrates it with concrete objects, names the design principles it depends on, and states its honest limitations.

2. What NIST RBAC2 Actually Specifies

The RBAC standard defines four model levels. Core RBAC (RBAC0) establishes users, roles, permissions, and sessions. Hierarchical RBAC (RBAC1) adds role inheritance. Constrained RBAC (RBAC2) adds separation of duty. Symmetric RBAC (RBAC3) combines the prior two. This architecture implements RBAC2: hierarchical roles with separation of duty constraints.

The elements relevant to this paper are precise.

- **Roles** are named collections of permissions. They are defined, documented, and stable. They are not ad hoc groupings of capabilities. The four roles in this architecture -- reader, contributor, approver, owner -- are a complete set for the resource types they govern. No custom roles are defined in version one.
- **Role hierarchy** is strict and transitive. Assignment to approver implicitly carries contributor and reader. This is explicit in the standard, not a convention of this implementation. A principal holding the approver role on a resource requires no additional assignment to exercise reader or contributor permissions on that resource.
- **Role assignments are scoped to a resource.** A principal does not hold a role globally. A principal holds a role on a specific spoke, a specific value stream, or a specific chain. The same principal may hold different roles on different resources. This is the mechanism that prevents access accumulation across product boundaries.
- **Separation of duty constraints** are the defining feature of RBAC2. A SOD constraint states that no single principal may hold two designated roles on the same resource simultaneously. The canonical enterprise example: the person who submits a transaction cannot be the person who approves it. In this architecture, SOD constraints are declared on the assignment record at the data model level, not in application logic.

The choice of RBAC2 as the rights model is not novel. What is novel is the choice of an append-only chain as the authoritative record for that model. The standard specifies the rights model. This architecture specifies how that model's history is stored and how authorization state is reconstructed from that history.

3. The Structural Audit Gap

The structural audit gap in conventional RBAC is best understood by examining what an auditor actually needs to answer during a SOC 2 Type II review period. The auditor's question is not 'what are the current access rights?' The auditor's question is 'what were the access rights at this specific moment, and were those rights properly authorized?'

A conventional RBAC implementation cannot answer this question from its rights store alone. The rights store reflects the current state of assignments. Any revocation since the moment in question has already modified or deleted the record. The auditor must consult the log. The log may answer the question -- if it was not rotated, if it is stored with sufficient fidelity, if the correlation between the log event and the rights record state at that moment is recoverable. These are operational properties, not structural ones. They require compensating controls, policy assertions, and auditor trust.

The gap has three components, each of which requires a separate compensating control in a conventional implementation.

- **State-at-time reconstruction.** The rights store holds current state. Historical state requires log replay. Log and store use different retention policies and are maintained by different systems. Compensating control: log retention policy, periodic backup, access review documentation.
- **Denial recordkeeping.** Most conventional implementations log successful authorizations reliably. Denials are frequently not logged, or logged at lower fidelity, or logged to a separate system. An unauthorized access attempt that was blocked leaves no trace in the rights store. It may leave a trace in an application log. Compensating control: application-level denial logging, SIEM integration.
- **Provenance of the authorization record itself.** The rights store records current assignments. It may record who granted an assignment and when. It does not, in most implementations, record the chain of authorization events that led to the current state in a form that is independently verifiable. Compensating control: ticketing system integration, periodic access review documentation.

Each compensating control introduces an additional system, an additional retention policy, and an additional point of potential failure. The SOC 2 auditor accepts this structure because it is standard practice. Standard practice is not the same as structural soundness. The gap is accepted, not closed.

4. Closure by Construction

An append-only hash-chained log -- a Chandra chain -- closes all three components of the structural audit gap without compensating controls. The mechanism is not procedural. It follows from the structural properties of the chain itself.

A Chandra chain is a sequence of chain units (CUs). Each CU contains a payload, a timestamp, and a cryptographic hash of the prior CU. The chain is append-only: units are added but never modified or deleted. The hash linkage makes any retroactive modification of a prior unit detectable -- the chain breaks at the point of modification. These properties are not claims. They are verifiable structural consequences of the data model.

When gr-identity -- the identity and authorization substrate -- uses a Chandra chain as its authoritative record, the three gap components close as follows.

- **State-at-time reconstruction.** Every grant and revocation is a CU in the chain. The chain is never modified. To reconstruct authorization state at time T, replay the chain to T and project the active assignments. No log correlation required. No external system required. The chain is the rights store and the audit log simultaneously. They cannot diverge because they are the same artifact.
- **Denial recordkeeping.** Every authorization decision -- grant and denial -- is a CU written at the time of the decision. A denial is structurally identical to a grant event: same chain, same hash linkage, same replay guarantee. The denied attempt is permanently in the chain with the same tamper-evidence properties as every other event. There is no separate logging system for denials because there is no separate logging system at all.
- **Provenance of the authorization record.** Every grant CU records the granting principal, the timestamp, and an optional reference to an external authorization event such as a ticket number. Every revocation CU records the revoking principal, the reason, and a reference to the grant CU being revoked. The chain is its own provenance record. No ticketing system integration is required to establish the chain of authorization, though the chain can carry references to external systems when they exist.

The gr-identity chain is separate from the domain chains of its consumer products. IAM events are not mixed with DXMachine workflow events or Chandra protocol events. The identity chain is independently auditable. Its integrity does not depend on the integrity of any consumer product's data.

5. The Objects

The following examples are concrete representations of chain units and query results from the gr-identity implementation. They are included because the architectural argument above is abstract. These objects make the argument falsifiable: either the implementation produces objects with these properties, or it does not.

The implementation is in Common Lisp using AllegroCache for persistent storage and the Chandra Protocol for chain management. Readers unfamiliar with Common Lisp syntax can read the objects as structured records. The field names are self-documenting.

5.1 The Mutable Rights Store Gap

The following SQL represents the rights store of a conventional RBAC implementation after a revocation has been processed. The principal previously had contributor access to the finance value stream. That access has been revoked.

```
-- Conventional RBAC: query the rights store after revocation
SELECT * FROM assignments
  WHERE principal_id = 'hp-007'
    AND resource_id  = 'vs-finance-close';

-- Result: 0 rows.

-- She had access. Now she does not.
-- When was it granted? The store does not know.
-- When was it revoked? The store does not know.
-- Who authorized the revocation? The store does not know.
-- Was she authorized at 2025-11-01T14:32:00Z? The store cannot answer.
-- The log might answer -- if it was not rotated.
```

Figure 1. A conventional rights store after revocation. The authorization history is gone. The log is a separate system.

5.2 A Grant Sequence as Chain Units

The same authorization event on an append-only chain. Three chain units are shown: the role grant, an authorization decision made while the role was active, and the subsequent revocation. All three are permanent. None can be modified without breaking the hash chain.

```
;; CU 1 -- role granted
(:cu-id       "cu-0041"
 :prior-hash  "a3f9c2..." ; hash of cu-0040
 :timestamp   1746000000
 :payload (:event          :role-granted
           :assignee-id    "hp-007"
           :assignee-type  :human
           :role           :contributor
           :resource-type  "value-stream"
           :resource-id    "vs-finance-close"
           :granted-by     "hp-001"
           :review-ticket  "TKT-4102"))

;; CU 2 -- authorization decision (gate called and passed)
(:cu-id       "cu-0044"
 :prior-hash  "b12c8f..." ; hash of cu-0043
 :timestamp   1746003600
 :payload (:event          :authorized
           :principal-id   "hp-007"
           :action         :checkout
           :resource-type  "spoke"
           :resource-id    "spk-finance-q2"
           :satisfied-by   :contributor
           :assignment-ref "cu-0041"))

;; CU 3 -- role revoked
(:cu-id       "cu-0051"
 :prior-hash  "d88a1b..." ; hash of cu-0050
 :timestamp   1746007200
 :payload (:event          :role-revoked
           :assignment-ref "cu-0041"
           :revoked-by     "hp-001"
           :reason         "contractor offboarded"
           :review-ticket  "TKT-4471"))
```

Figure 2. Grant, authorization decision, and revocation as three permanent chain units. The authorization at cu-0044 is recorded even after the role is revoked at cu-0051. The chain answers every question the SQL store could not.

5.3 The Authorization Decision as a Chain Unit

Every call to gri:authorized-p writes a chain unit regardless of outcome. The decision is part of the permanent record, not an ephemeral runtime event. The :assignment-ref field links the decision to the specific grant CU that satisfied it, making the authorization chain fully traceable without external systems.

```
;; In gr-identity -- every gate call produces a CU
(defun gri:authorized-p (principal-id action resource-type resource-id)
  (let* ((assignments (collect-active-assignments
                       principal-id resource-type resource-id))
         (min-role    (action->minimum-role action))
         (satisfying  (find-if (lambda (a)
                                 (role>= (gra-role a) min-role))
                               assignments))
         (result      (not (null satisfying))))
    ;; Write the decision to the chain unconditionally
    (write-auth-cu principal-id action resource-type resource-id
                   result satisfying)
    result))
```

Figure 3. The authorization gate function. The chain write is unconditional -- denials produce chain units with the same properties as grants.

5.4 Separation of Duty -- The Denial is an Event

RBAC2 separation of duty constraints are enforced at the authorization gate. When a constraint is violated, the denial is a chain unit with the SOD class and constraint reference recorded in the payload. In a conventional implementation, a SOD denial either goes unlogged or appears in an application log with no connection to the rights record. Here, the denial is structurally identical to every other chain unit.

```
;; hp-009 submitted the work item. hp-009 attempts to approve it.
;; The SOD constraint :submit-approve-separation is active on
;; vs-finance-close. The gate denies and writes the CU.

(:cu-id       "cu-0089"
 :prior-hash  "f44b3c..." ; hash of cu-0088
 :timestamp   1746010800
 :payload (:event          :authorization-denied
           :principal-id   "hp-009"
           :action         :approve
           :resource-type  "value-stream"
           :resource-id    "vs-finance-close"
           :reason         :sod-violation
           :sod-class      :submit-approve-separation
           :conflicting-cu "cu-0078"))  ; the submit CU
```

Figure 4. A SOD denial as a chain unit. The conflicting-cu field references the submit event that triggered the constraint. The denial is permanent, hash-linked, and independently verifiable.

5.5 State Reconstruction at Arbitrary Time

The chain enables a query that a conventional RBAC implementation cannot answer: reconstruct authorization state for a principal on a resource at an arbitrary past timestamp. The result is derived from immutable chain evidence, not from a mutable store correlated against an external log.

```
;; Query: what were hp-007's rights on vs-finance-close
;; at 2025-11-01T00:00:00Z (Unix: 1762000000)?

(gri:reconstruct-rights "hp-007"
                        "value-stream"
                        "vs-finance-close"
                        :at-time 1762000000)

;; Result:
(:principal-id   "hp-007"
 :resource-type  "value-stream"
 :resource-id    "vs-finance-close"
 :role           :contributor
 :granted-at     1746000000  ; cu-0041
 :revoked-at     1746007200  ; cu-0051
 :held-at-query  nil         ; revoked before query time
 :chain-evidence ("cu-0041" "cu-0051"))

;; The answer is derived by replaying the chain to :at-time.
;; No log required. No external system required.
;; The chain-evidence field names the CUs that determined the result.
```

Figure 5. State reconstruction from chain replay. The :chain-evidence field makes the result independently verifiable. An auditor can inspect the named CUs directly and confirm the derivation.

5.6 Provisioning Evidence for CC6.3

SOC 2 CC6.3 requires evidence that access provisioning follows a defined process and that provisioning changes are reviewed. The chain carries this evidence structurally. Every group membership change is a CU with the authorizing principal and an optional reference to an external review ticket. The chain is the provisioning record. No separate HRMS integration or access review spreadsheet is required to satisfy the control, though external references can be included when they exist.

```
;; Group membership added during a periodic access review
(:cu-id       "cu-0201"
 :prior-hash  "9c3d44..." ; hash of cu-0200
 :timestamp   1746100000
 :payload (:event          :group-member-added
           :principal-id   "hp-022"
           :group-id       "grg-finance-team"
           :added-by       "hp-001"
           :review-ticket  "TKT-4471"
           :review-cycle   "2026-Q1"))

;; The :review-ticket and :review-cycle fields link the provisioning
;; event to the formal access review process.
;; CC6.3 asks: 'show me that provisioning follows a defined process.'
;; The chain answers: here is every provisioning event, who authorized
;; it, when, and against which review cycle.
```

Figure 6. Group provisioning as a chain unit. The review-ticket and review-cycle fields satisfy CC6.3 without a separate HRMS integration.

6. Design Principles Named

The following principles are stated as named architectural commitments specific to the auditable authorization model. They extend and specialize the principles named in the companion paper, Auditable by Construction (General Reasoning, Inc., March 2026).

P-A1 -- Chain Primacy

The append-only chain is the authoritative record for all authorization state. The queryable rights store is a derived projection, not an independent source of truth. In any conflict between the chain and the projection, the chain governs.

P-A2 -- Decision Immutability

Every authorization decision -- grant and denial -- is a chain unit written at decision time. The gate does not return a result without first writing the unit. There is no authorization event that is not in the chain.

P-A3 -- Separation of Duty as Data

SOD constraints are declared in the assignment record at the data model level. They are not application logic. An auditor examining the data model can enumerate all active constraints without reading application code.

P-A4 -- Reconstruction Without External Correlation

Authorization state at any past time must be reconstructable by replaying the chain alone. No external log, no ticketing system, and no compensating control is required to answer the auditor's state-at-time question.

P-A5 -- Identity Chain Isolation

The gr-identity chain is separate from the domain chains of its consumer products. IAM events are not co-mingled with workflow or protocol events. The identity chain can be audited independently of any consumer product.

P-A6 -- Provenance in the Chain

Every grant and revocation CU carries the authorizing principal, the timestamp, and an optional external reference. The chain is its own provenance record. Provenance does not depend on external system availability.

7. Honest Limitations

The following limitations are stated explicitly. They are known, and addressing them is part of the ongoing architecture work. None of them invalidate the core argument; all of them constrain it.

■ The chain itself is not independently certified.

The Chandra chain provides append-only, hash-linked storage. Its tamper-evidence properties follow from the data model. The chain implementation has not been independently audited or formally verified. The immutability argument depends on the application-level protections described in the companion paper, not on storage-layer certification. A rigorous reviewer will ask who certifies the chain implementation. The answer is: it is tested, reviewed, and version-controlled, but not formally verified.

■ The projection may lag the chain.

The queryable rights projection is derived from the chain. In a high-write environment, the projection may be stale relative to the most recent chain units. Authorization decisions are made against the projection, not against a live chain replay. A grant written to the chain may not be reflected in the projection immediately. This is a known consistency window, not a correctness failure -- the chain remains authoritative -- but it must be documented and its duration bounded for any compliance claim.

■ AllegroCache is not a certified persistence layer.

The chain units and assignment projections are stored in AllegroCache. AllegroCache has not been independently certified for any compliance framework. The tamper-evidence of stored chain units depends on application-level protections and operating environment controls, not on storage-layer certification. This is consistent with the limitation stated in the companion paper and applies equally here.

■ State reconstruction is not yet exposed as an auditor-facing interface.

The gri:reconstruct-rights function described in Section 5.5 exists as an architectural capability. It is not yet wrapped in an auditor-facing reporting interface. Until that interface exists and its output is validated against chain content, the reconstruction capability is an internal tool, not a compliance deliverable.

■ Group membership chain coverage is not yet enforced at load time.

The principle that every provisioning event is a chain unit depends on every code path that modifies group membership writing a CU. Enforcement of this requirement is by policy and code review in the

current implementation, not by a compile-time or load-time structural check. This is a compensating-control situation equivalent to the limitation described in the companion paper regarding macro-enforced contracts.

■ The SOC 2 audit clock has not started.

SOC 2 Type II requires twelve months of continuous evidence. The architecture described here is the foundation for generating that evidence. It is not yet the evidence itself. Until the control logging infrastructure is operational and auditor-agreed controls are defined, this architecture is a commitment, not a certification.

8. Conclusion

The structural audit gap in conventional RBAC is not a failure of process or policy. It is a consequence of a specific architectural choice: the rights store is mutable, and the audit log is separate. That choice is so common it is treated as inevitable. It is not.

When the authoritative record for an RBAC2 implementation is an append-only hash-chained log, the gap closes. Grants and revocations are permanent. Authorization decisions are permanent. Denials are permanent. State at any past moment is reconstructable by replay. None of these properties require compensating controls. They are structural consequences of the chain.

The compliance implication is direct. SOC 2 CC6.1 (logical access controls), CC6.2 (new access provisioning), CC6.3 (access removal and modification), and AU-10 (non-repudiation) under FedRAMP all ask the same question in different forms: can you show me what access existed, who authorized it, and that the record cannot have been altered? A conventional RBAC implementation answers with compensating controls and auditor trust. This architecture answers with chain replay.

The limitations in Section 7 are real. The chain implementation is not formally verified. The projection may lag. The auditor-facing reconstruction interface does not yet exist. These are honest constraints on an honest claim, not caveats appended to marketing language.

The claim itself is narrow and falsifiable: an append-only hash-chained log as the authoritative RBAC record closes the structural audit gap by construction. If the chain does not preserve grants, if revocations modify rather than append, if reconstruction fails at an arbitrary past time, the claim is false. The architecture is testable against its own claims.

General Reasoning, Inc. · May 2026