COMMERCEGOV

# CommerceGov Architecture Whitepaper

Production Authority for governed Shopify operations

PUBLIC ARCHITECTURE EDITION

AUGUST 2026

Public product truth. Architecture and claims qualified to current repository evidence.

# Abstract

CommerceGov is a governance and control plane between proposed operational change and Shopify production. It separates technical access from Production Authority: the right for one specific production mutation to proceed under current policy, production state, operator authority, review, approval, execution, and verification conditions.

The system accepts proposals from people, applications, AI systems, spreadsheets and imports, and internal systems. A proposal carries no production authority merely because its source has credentials or Shopify access. CommerceGov evaluates the proposed change, binds human approval to the current governed decision, admits only an applicable command, executes through a controlled worker path, verifies the governed result by authoritative readback, and preserves evidence for the authority chain.

Core boundary: CommerceGov governs production authority inside the access Shopify already grants. It does not replace Shopify RBAC and does not claim to prevent direct Shopify writes made outside the governed path unless the surrounding environment separately enforces that restriction.

**Proposal** → **Policy** → **Review** → **Approval** → **Controlled change** → **Verification** → **Evidence**

# 1. Access, intent, and Production Authority

Shopify access answers what a person or system can technically reach or perform. Intent describes a desired change. Production Authority answers whether that exact change may proceed now. These are different questions.

Production Authority is contextual and perishable. An authenticated operator may lack assignment to the target store. A previously approved change may no longer be current. A valid proposal may conflict with policy, store state, a safety control, or a newer decision. An allowed action in one store never implies authority in another.

The current CommerceGov path preserves four non-equivalences:

- Proposal is not Authority.
- Review is not Approval.
- Approved is not Applied.
- Applied is not Verified.

These distinctions prevent interface state, credentials, queue acceptance, transport success, or earlier approval from being mistaken for current production authority.

# 2. System boundary

CommerceGov sits between proposal systems and Shopify production. Proposal systems may create candidates, but they do not receive the worker's production capability. The governed path is:

Proposal -> Policy -> Review -> Approval -> Controlled production change -> Verification -> Evidence

The implementation expands the controlled-change step into admitted command, queue, worker dispatch, mutation outbox, Shopify writeback, authoritative readback, finalization, and audit. UI and API layers submit governed mutation commands; they do not perform Shopify writeback. Worker execution is the controlled production boundary.

BOUNDARY CONDITION
CommerceGov can prove and enforce the path it controls. If an external actor writes directly through separate Shopify access, CommerceGov can later observe and govern its response where the integration supplies evidence; it cannot retroactively claim that the external write passed through CommerceGov.

# 3. Agency, operator, and store topology

The agency is the organizational context for operators, assignments, shared operating policy, store-specific governance context, and aggregated evidence. A store remains the Shopify credential, resource, event, failure, concurrency, and mutation boundary.


An operator's effective authority is the intersection of identity, agency membership, assigned stores, role and permission, requested operation, current workflow state, current decision identity, store policy context, OAuth readiness, and active safety controls. Assignment to one store does not imply another. Cross-store work decomposes into exact store-scoped operations so that approval, execution, verification, failure, and evidence remain attributable.

Shared policy can reduce repeated administration. Store-specific rules or exceptions may specialize the operating context where the current policy surface supports them. This architecture does not treat demo users, demo store counts, or a convenience role configuration as the recommended production separation-of-duties model.

# 4. Governed mutation model

A governed mutation identifies the store, target resource, mutation class, exact affected scope, baseline or before-state, proposed value, source and actor context, decision/version identity, and integrity bindings used by admission. The model is mutation-class extensible: new mutation classes must register semantics, carry exact authority, execute through an allowed worker route, produce authoritative verification evidence, and preserve audit lineage.

The current repository evidences verification-capable product copy fields (title, description, SEO title, and SEO description) and a registered product-field set (vendor, product type, tags, and Shopify status). It also contains governed execution paths for variant price, relationships, typed owner fields, and media-resource fields. This is not a claim that every path has the same certification level or is included in every pilot.

Scope rule: Certified production scope is selected and evidenced per engagement. Implemented mutation classes are not automatically enabled, certified, or in pilot scope. The architecture is not permanently limited to four content fields, and it does not imply universal Shopify-object coverage.

# 5. Proposal and policy evaluation

Proposal sources include AI systems and agents, applications, people and operators, CSV or spreadsheet imports, and internal systems. CommerceGov normalizes the proposed change into a governed shape before authority is evaluated. Source authentication or Shopify access does not convert a proposal into production authority.

Policy evaluation can admit, block, or route work for review according to the current store and agency context. Policy output is evidence used by the authority chain; it is not human approval. Public descriptions of policy-based autonomous low-risk execution are a general governance pattern, not a claim that the current CommerceGov pilot grants AI autonomous production authority.

# 6. Review and human approval

Review establishes what a human evaluated. Approval is a separate, attributable authority decision. The current governed production path retains a human approval boundary before Apply.

Approval is useful only when it binds to the current governed change. CommerceGov's current exact-scope admission resolves an approved decision by decision-version identity, scope key, scope revision, audit identity, and review-cycle identity. The decision must still be current, the approval must remain valid, and resource and target identity must still be available. Missing, invalidated, superseded, or lineage-mismatched authority fails closed.

One operator may hold more than one capability where policy permits, but the evidence records remain distinct. An interface that shows Review, Approve, and Apply actions does not collapse their meanings.

# 7. Exact authority and stale-authority semantics

Material change after approval creates a new authority question. CommerceGov does not stretch an old approval over a new payload, scope revision, review cycle, production baseline, or decision version.

The current admission model checks that the requested decision is the latest current decision for the persisted scope; the decision and approval share scope, revision, audit, and review-cycle lineage; store identity matches; the mutation class has a supported worker route; and current external-change availability permits Apply. For product-field admission, before-values are normalized and compared with the authoritative current product values. A mismatch is stale and is rejected.

Stale or superseded authority therefore fails closed. The safe response is reassessment: create or select the current proposal, review the current change, obtain current approval, and admit a new command.

# 8. Controlled execution and writeback

The canonical runtime path is:

submit_governed_mutation_command -> enqueue_mutation_command -> dispatch_governed_mutation -> worker -> mutation outbox/writeback -> Shopify -> verification/finalization -> audit

Admission validates the command envelope and governance context. Execution checks current worker-claimable state, human approval, allowed workflow transition, active product status, Shopify OAuth authority, mutation controls, kill switches, and per-store concurrency. Idempotency and payload identity prevent duplicate execution. UI and API components do not own the Shopify write capability.

Approved still does not mean Applied. Apply requests governed execution. A command can be blocked, fail before a side effect, fail retryably, or reach Shopify and still require verification before the governed outcome is settled.

# 9. Apply-time authoritative verification

Apply-time verification is part of completing the governed production change, not a later reporting convenience.


For governed product mutations, the writeback context carries an expected verification witness for the exact changed fields. Finalization obtains authoritative Shopify readback, resolves each governed value through its field-specific mapping, normalizes expected and observed values, and records field equality. A missing authoritative value, read failure, product absence, or mismatch cannot be presented as Verified. Bounded deferral may address transient readback conditions; exhaustion or persistent mismatch terminalizes the outcome as conflict with reconciliation required.

Verification is mutation scoped. It proves the governed change represented in the witness. Where authoritative readback makes collateral invariants observable, a mutation-class verifier may check those invariants too. This is evidence for a verified governed production state, not a universal assertion that every property of the Shopify object was verified.

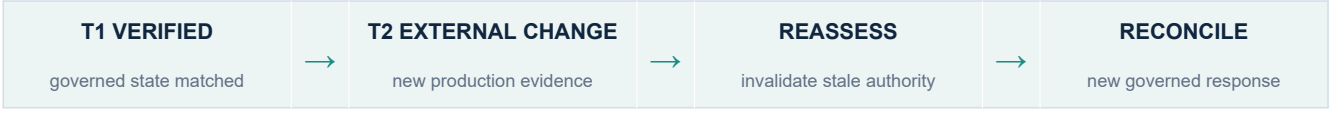
# 10. Applied is not Verified

Applied describes execution/writeback progression. Verified describes an evidence-backed comparison between the approved governed expectation and authoritative observed production state. The distinction matters because queue acceptance, a successful HTTP response, or an outbox completion marker is not itself proof that Shopify exposes the intended governed state.

The current finalization path keeps verification pending or deferred while authoritative evidence is incomplete. Equality produces verified evidence; persistent mismatch produces a conflict and reconciliation requirement. Settlement into the applied lifecycle is linked to verified write evidence for paths using this witness.

# 11. External change and later divergence

A production state can change after a valid governed execution. That later event is not retroactive proof that Apply failed.




At T1, CommerceGov can establish that the governed mutation matched authoritative production readback. At T2, a person, application, Shopify-side process, or another system may change that state. Sync, webhook, or external-change handling can then identify divergence and route reassessment, reconciliation, invalidation, or a governed recovery action.

External evidence can make the current decision or approval stale. The response is to evaluate current state and current authority, not to rewrite the historical T1 verification result.

# 12. Reconciliation, recovery, and rollback

Reconciliation determines whether evidence is sufficient and whether a safe governed response exists. The repository contains reconciliation paths that check exact write evidence, newer-audit conflicts, policy or drift conflicts, inactive state, and repair eligibility. A repair is itself controlled and attributable.

Rollback is not a universal promise. Some mutation classes have specific rollback verification support; broader rollback may be unavailable or unsafe. Where used, rollback is a new governed mutation with its own authority, execution, verification, and evidence. Historical failure or divergence is never erased.

# 13. Evidence and audit model

Evidence is part of the authority chain. Depending on the mutation class and path, CommerceGov can connect:

- proposal or change identity, source, and actor;
- prior authoritative or synchronized production state;
- proposed state and exact mutation scope;
- policy and review state;
- approval decision and binding identities;
- admitted command and execution attempts;
- Shopify writeback response and authoritative readback;
- per-field or mutation-class verification result;
- exceptions, conflicts, retries, and reconciliation actions.

The command ledger is the mutation-intent source, writeback audit records external side effects, and product/catalog state supplies the baseline. Projections are derived views and are not authority. Audit evidence is append-only; later success does not erase earlier attempts or failures.

# 14. Failure-closed invariants

CommerceGov blocks the governed path when required authority or safety evidence is absent. Core invariants include command identity and idempotency, current context binding, valid workflow transition, valid human approval, active resource state, Shopify OAuth readiness, worker-only mutation execution, kill-switch enforcement, one active mutation per store, and mandatory audit events.

Unknown is not success. Missing readback, unavailable authority, invalid scope identity, superseded decision, invalidated approval, unsupported mutation class, and unresolved conflict must not silently fall through to production or Verified.

# 15. Integration and proposal-source model

CommerceGov is source agnostic at the governance boundary. Integrations should submit normalized proposals and provenance, not share the worker's Shopify credentials. A source can be registered, attributed, limited, suspended, or rejected independently of production execution.

This paper describes source categories, not a native connector catalog. Named third-party integration is not implied unless separately implemented and documented. The architectural contract remains stable: proposal systems produce candidates; CommerceGov evaluates authority; the controlled worker operates within existing Shopify access.

# 16. Six-week Production Pilot measurement model

The public Production Pilot is a six-week measurement engagement, not a predetermined success demonstration.

- Weeks 1-2: baseline and controlled comparison under equivalent starting conditions and workload where practical.
- Weeks 3-6: governed workload through the CommerceGov path.

The measurement boundary is Work released / Proposal received -> Verified production state. Five primary pillars organize the evidence:

| TIME | REWORK | CAPACITY | AUTHORITY | VERIFICATION |
|---|---|---|---|---|
| elapsed + effort | correction + repetition | verified governed work | correct path + scope | outcome evidence |

Time measures elapsed and active effort across the defined boundary. Rework measures correction, repetition, and exception handling. Capacity measures verified governed workload per constrained resource without converting an engineering throughput test into a human productivity claim. Authority measures whether work followed the intended actor, review, approval, and scope controls. Verification measures whether governed outcomes received authoritative evidence and how conflicts were handled.

Internal calibration sets - for example a particular product or mutation count - do not define the public pilot. Any 3-5x operator-capacity figure is a pilot target or hypothesis only unless authoritative participant measurement establishes a result.

# 17. Security and operational boundaries

CommerceGov depends on the security of Shopify OAuth, its worker runtime, command and audit storage, tenant/store scoping, secrets handling, and the surrounding deployment. It adds governed Production Authority inside existing Shopify access; it is not a replacement for Shopify roles, app scopes, credential hygiene, platform controls, or organizational security administration.

Safety controls include global and per-store mutation gates, worker-only execution, command-state checks, idempotency, per-store concurrency, current-state binding, explicit terminal failure, and audit requirements. This public paper is an architecture description, not a security certification, compliance attestation, or guarantee that external actors cannot write directly.

# 18. Current scope and limitations

The current product path is human-authorized. AI can propose; it does not automatically receive production authority. Verification evidence is scoped to the governed mutation and supported observable collateral invariants. Policy inheritance, exception handling, recovery, and mutation-class coverage vary by implemented path and engagement scope.

This paper does not claim guaranteed error reduction, zero rework, achieved 3-5x capacity, universal rollback, universal whole-object verification, support for every Shopify resource, native integration with every proposal source, or prevention of writes made through external Shopify access.

# 19. Conclusion

CommerceGov makes Production Authority an explicit operational system. It preserves the difference between access and authority, proposal and decision, approval and execution, Applied and Verified, and a verified T1 state and later T2 divergence.

The result is a reconstructable path from candidate change to verified governed production state: exact scope, current human authority, controlled writeback, authoritative verification, and evidence. Agencies can coordinate operators and stores without turning shared administration into implicit cross-store authority.