



JYNSYN ARCHITECTURE SERIES • WHITE PAPER

# Sovereign Operational Domains

## Durable Accountability for AI-Enabled Physical Security

Keeping the chain of authority stable as models, machines, and actors change

THE SOD IS DURABLE. THE STACK EVOLVES.

**Timothy E. Chandler** BEAR CANYON SYSTEMS

Version 1.0 • October 2026 • doi:10.5281/zenodo.23110479

Security teams are pairing overnight guards with docked drones, ground robots and AI vision models, and they replace those pieces far more often than they update who is responsible when one of them gets it wrong. This paper argues that what a company must hold stable is not its technology stack but its operational definition: what it is responsible for, who may authorize what, and what it can prove afterward. Jynsyn calls that definition a **Sovereign Operational Domain (SOD)**.

### Change without drift.

Models, machines and contractors change; responsibilities and approvals change only by decision.

### Access without authority.

Admitting a system to a role never lets it act on its own say-so.

### Failure within limits.

An outage or a lost link narrows what is allowed; it never widens it.

### Proof on demand.

Every consequential action must trace to an accountable role and a named person.

## IN 60 SECONDS

### Why it matters

- Guards now work alongside drones, robots and AI vision whose suppliers change faster than anyone updates who is accountable.
- When something goes wrong, "who approved this?" is often buried in a vendor's prompt, mission rules or logs.
- Data centers and other lights-out facilities face the same problem, often with no one on site.

### How Jynsyn V3 approaches it

- A durable **Sovereign Operational Domain (SOD)** holds the company's responsibilities, delegated authority, evidence and continuity rules, separately from whatever does the work.
- Replacing a model, platform or contractor becomes a governed change of system participant, working through the identity, policy and operations systems the company already runs.

### What you get

- Switch vendors without redefining who may authorize what.
- Failures degrade within limits set in advance.
- Incidents compared on the same terms across regions and contractors.
- Every consequential action required to be traceable to an accountable role and a named person.

**Status:** Jynsyn V3 is a reference architecture, not a product or certification, and it is not yet proven in the field.

---

## Abstract

Organizations now assemble AI capability from models, agents, hosted services, autonomous machines and existing enterprise systems, and replace those parts far more often than they change who is responsible for the outcome. This paper argues that what an organization should hold stable is not its technology stack but its operational definition: what it is responsible for, who may authorize what, what evidence it must keep, and how it keeps operating when a component fails. Jynsyn calls that definition a **Sovereign Operational Domain (SOD)**.

We follow one illustrative case: overnight security at a logistics company whose guard contractor now pairs a single on-site guard with a docked drone and a ground robot. It shows how the Jynsyn V3 reference architecture separates a durable responsibility from the replaceable system participants that serve it, how the company scales to a region served by a different contractor without rebuilding its authority model, and why, when an AI proposes that a drone follow an intruder off the property, the decision is not the drone's or the model's to make.

The claim is architectural: organizations need a durable object that composes established mechanisms around accountable authority, specified as requirements an implementation can be tested against. The paper ends with propositions that would show it wrong.

---

# Contents

|   |                                                                   |    |                                                     |
|---|-------------------------------------------------------------------|----|-----------------------------------------------------|
| 1 | The problem: capability changes faster than accountability        | 7  | The drone stays on the property: composed authority |
| 2 | The Sovereign Operational Domain                                  | 8  | What a conformance claim looks like                 |
| 3 | Responsibilities are durable; system participants are replaceable | 9  | What would show this is wrong                       |
| 4 | Participation is not authority                                    | 10 | Invitation                                          |
| 5 | One alarm, from inference to consequence                          |    | Glossary                                            |
| 6 | When the stack changes                                            |    | References                                          |

---

# Status of this paper

This paper explains Jynsyn V3, a reference architecture of 28 normative documents that Bear Canyon Systems will publish as the Jynsyn V3 release [14]. The paper is not normative: it adds no requirements, and where it simplifies, V3 controls.

V3 is a reference architecture, not a standard, a certification program or a product. It has not been proven in the field: no deployment has been evaluated against it, and this paper reports no measurements. The case is constructed and describes no customer, contractor or vendor. We say this once here rather than in every section. What the paper offers now is a shared vocabulary for security, facilities and IT leaders to scope vendor contracts, authority and evidence requirements before a switch, not after.

Jynsyn cites, and was informed by, the Osero architecture by Joseph Dominic Natoli III [1]. How an architecture of that kind can participate in a SOD is left to future work.

---

## **1 The problem: capability changes faster than accountability**

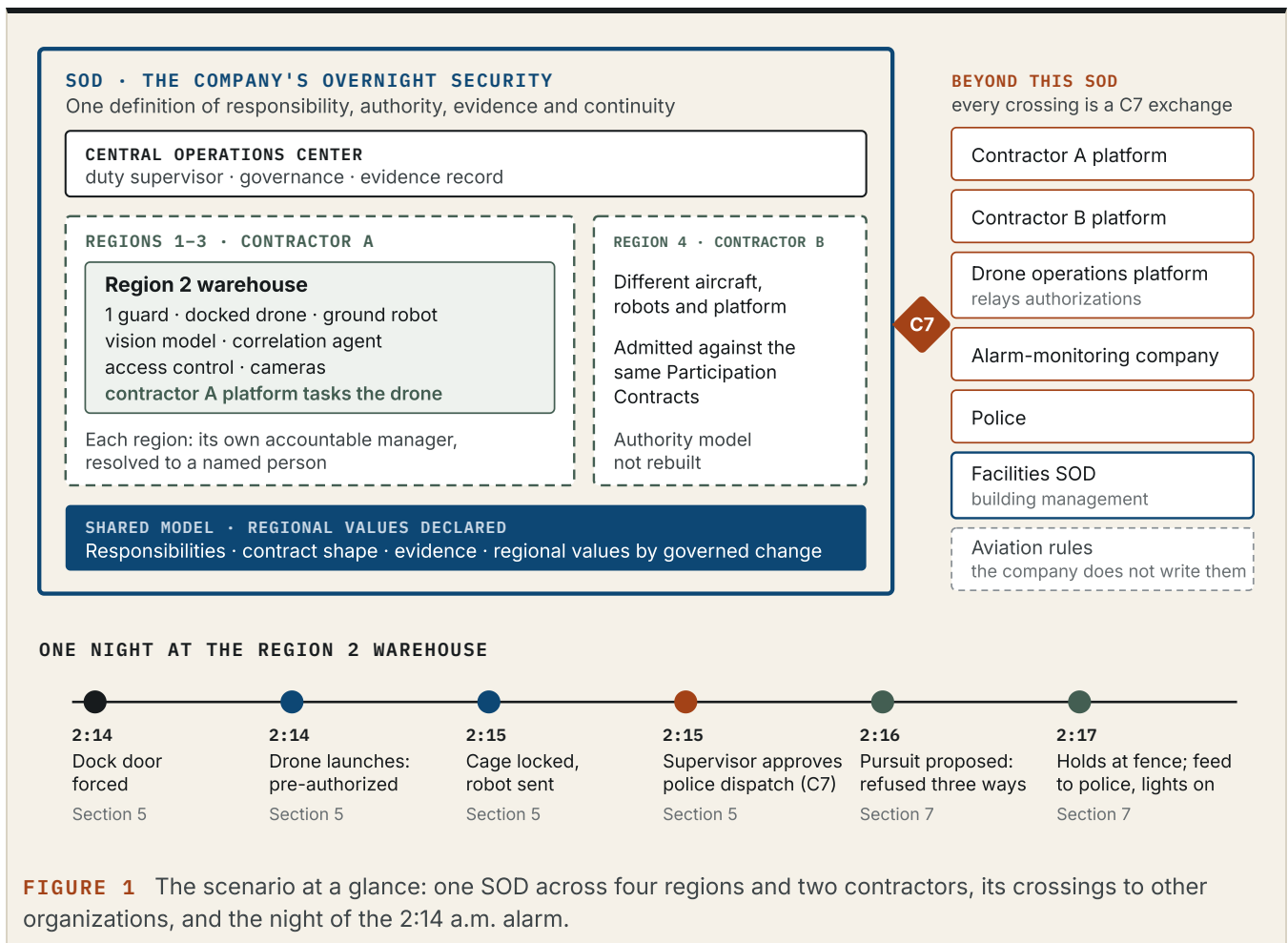
Consider a logistics company with distribution warehouses in three regions. Each has badge readers and door controllers, cameras on a video management system, intrusion alarms, and a fire and life-safety system installed to code. A central operations center monitors every site overnight, and enterprise IT runs identity, networking and a security operations center.

Two years ago each site had four overnight guards. Today it has one. Like much of its industry, the company's guard contractor has moved from supplying guards to augmenting them with technology; one of the largest global providers reported about a third of its 2025 sales from technology and solutions [19]. Each site now has a drone in a rooftop dock and a ground robot patrolling the yard, both run from the contractor's own orchestration platform, with flight authorizations managed through a drone operations platform. On the company's side, a hosted vision model reviews camera and drone video when an alarm fires, and a software agent correlates the alarm with badge records, maintenance schedules and recent activity, then recommends a response to the operator on duty.

It works, and it is about to change. The company moves its vision model onto site hardware. It opens warehouses in a fourth region, where its contractor does not operate, so a second contractor with different aircraft, robots and software will serve them. And the drones are beginning to fly beyond the operator's line of sight.

Through all of that, some things should not change at all. The company is still responsible for responding to intrusions. The same actions still need the same approvals. A reviewer still needs to reconstruct, weeks later, why a door was locked, why a drone held at the fence or why the police were called. None of those obligations belongs to the vision model, the agent or the contractor's platform, yet that is where they have ended up: in a prompt, in an agent's tool permissions, in mission rules in a vendor console, and in logs whose format belongs to whoever supplied the aircraft. The most convenient platform is the one supplied by the device or guard provider itself, which is also the one that makes switching providers most expensive.

Figure 1 shows the whole scenario on one page.



We start from one assumption, and most of this paper follows from it: every part of this stack will be replaced. Some of it on purpose, like the vision model moving on site or the second contractor in the fourth region, and some of it by accident, at 2 a.m., when a link drops or an agent starts getting things wrong. If that is true, accountability cannot live in any of those parts. It has to sit somewhere that outlasts them, the company has to decide who gets in and what they may do once they are in, and a failure has to leave less authority in play, not more. If the assumption is wrong, so is much of what follows; Section 9 lists what would show it.

The question is therefore narrow: **what must stay durable when the capability changes, and how can an organization specify it so that replacing a component, or a contractor, cannot quietly change it?**

WHAT CHANGES, AND WHAT STAYS

| WHAT CHANGES: THE STACK                 | WHAT STAYS: THE SOD                                                       |
|-----------------------------------------|---------------------------------------------------------------------------|
| Vision models and AI agents             | The Responsibility: after-hours intrusion response                        |
| Drones, robots and their autopilots     | Who may authorize what: pre-approved envelopes and escalation rules       |
| Guard contractors and their platforms   | The accountable role, resolved to a named person in each region           |
| Network links, hosting and connectivity | The disclosure rules for every crossing to another organization           |
| Each vendor's log and evidence formats  | The evidence the company owes, and how it keeps operating when parts fail |

## 2 The Sovereign Operational Domain

A **Sovereign Operational Domain** is the durable, owner-controlled boundary in which an organization defines responsibility, identity, authority, policy, resources, permitted actions, evidence, continuity and participation, independently of the models, agents, hardware, applications and infrastructure that implement them [14: SOD Definition]. Security readers should note that SOD is unrelated to separation of duties, the control commonly abbreviated SoD; in this paper SOD always means Sovereign Operational Domain.

In our case, the SOD is the company's definition of overnight security operations: its responsibilities, the authority delegated for each, the systems and machines they act on, and the evidence and continuity rules that apply.

Three clarifications prevent most misreadings.

*"Sovereign" refers to people and institutions, not software.* A model does not become sovereign because it is capable, and a drone does not acquire authority because it can fly itself.

*A SOD is an accountability boundary, not a technology boundary.* It is not a site, a network, a cloud tenant, a product or an agent. Our company could define one SOD for overnight security across every region, one per region, or separate SODs for security and inventory. The choice follows where accountable authority, consequence and continuity actually differ. A network hop does not create a new boundary unless the assumptions about identity, authority, disclosure, integrity, accountability or consequence change across it [14: Normative Conventions, "Trust boundary"]. Nor is a SOD an identity trust domain, such as a SPIFFE trust domain [7–9]: a trust domain establishes who vouches for an identity, and a SOD may rely on several.

A SOD integrates with the enterprise; it does not absorb it. The company's identity provider, security operations center, ticketing system and network remain owned by enterprise IT. The SOD consumes them as supporting services and does not require replacing them [14: Change Board 03].

Here is what that looks like in practice. At 2:14 a.m. a dock door is forced at the region 2 warehouse. The drone launches because the company pre-authorized verification flights; the storage-cage doors lock because that action was pre-approved for this situation; the police call waits for the duty supervisor because the company requires a person for it; and nothing can touch the fire exits, because no role holds that authority. Each of those answers comes from the company's SOD, not from the model, the drone or the contractor's platform, which is why they survive when any of those change (Section 6). Section 5 walks through the alarm in full (Figure 3).

Durability does not mean the definition never changes. It means changes to responsibility, authority, policy or durable state happen as governed, evidenced changes, never as a side effect of swapping a component.

The idea of a governed boundary is not new. Security practice has long defined domains administered by a single authority, and the authorization boundary in NIST's Risk Management Framework sets the scope of one official's responsibility and accountability, including services bought from outside providers [23]. "Sovereign" is used as it now is for cloud and AI: operational control over who runs, decides, responds and approves exceptions [24], here held by accountable people rather than by a region or a data center. A SOD is not a sovereignty zone for data or compute, though it may use one. What Jynsyn adds is keeping that boundary stable while the models, machines and contractors inside it change.

---

### 3 Responsibilities are durable; system participants are replaceable

The central move in Jynsyn V3 is to separate *what must be done*, the Responsibility, from *what currently does it*, the implementation composition. The SOD then defines, for each Responsibility, the authority, evidence, continuity and participation rules that stay fixed while the composition changes.

A **Responsibility** is a durable statement of work the SOD requires. In our case one Responsibility is *after-hours intrusion response*: assess an intrusion alarm, contain it where authorized, maintain observation, notify the right people and preserve the record. Every Responsibility carries a consequence classification [14: Consequence Classification]. An operation is consequential when an allowed, denied, erroneous, delayed or disclosed outcome can materially affect, among other things, safety or rights, security, legal or financial obligations, privacy, authoritative records or continuity. "Consequential" does not mean "uses AI": deterministic software can be consequential, and inference can be trivial. V3 imposes no risk score of its own. The company can reuse its existing risk matrix, provided the mapping to the architecture's obligations is explicit and no participant can lower its own class.

Each consequential Responsibility must also resolve to a durable accountable organizational role and, through authoritative records, to a named person presently holding it [14: [Accountability Resolution](#)]. Here the role is the regional security operations manager, resolved to a person by the HR system and on-call roster. That person does not approve every action; most actions below are governed at machine speed. The point is that delegation to software or to a contractor never severs the chain back to an accountable human. Organizational accountability for AI is established practice [2, 11, 12]; V3 requires that delegation not break it. It does not decide legal liability.

The Responsibility is served by an **implementation composition**: whatever combination of system participants actually does the work, alongside the people who operate it. In region 2 that is the vision model, the correlation agent, the contractor's platform, the drone and robot and the access-control system, working with the guard and the duty supervisor. V3 asks the composition as a whole to satisfy the Contracts that apply (Figure 2).

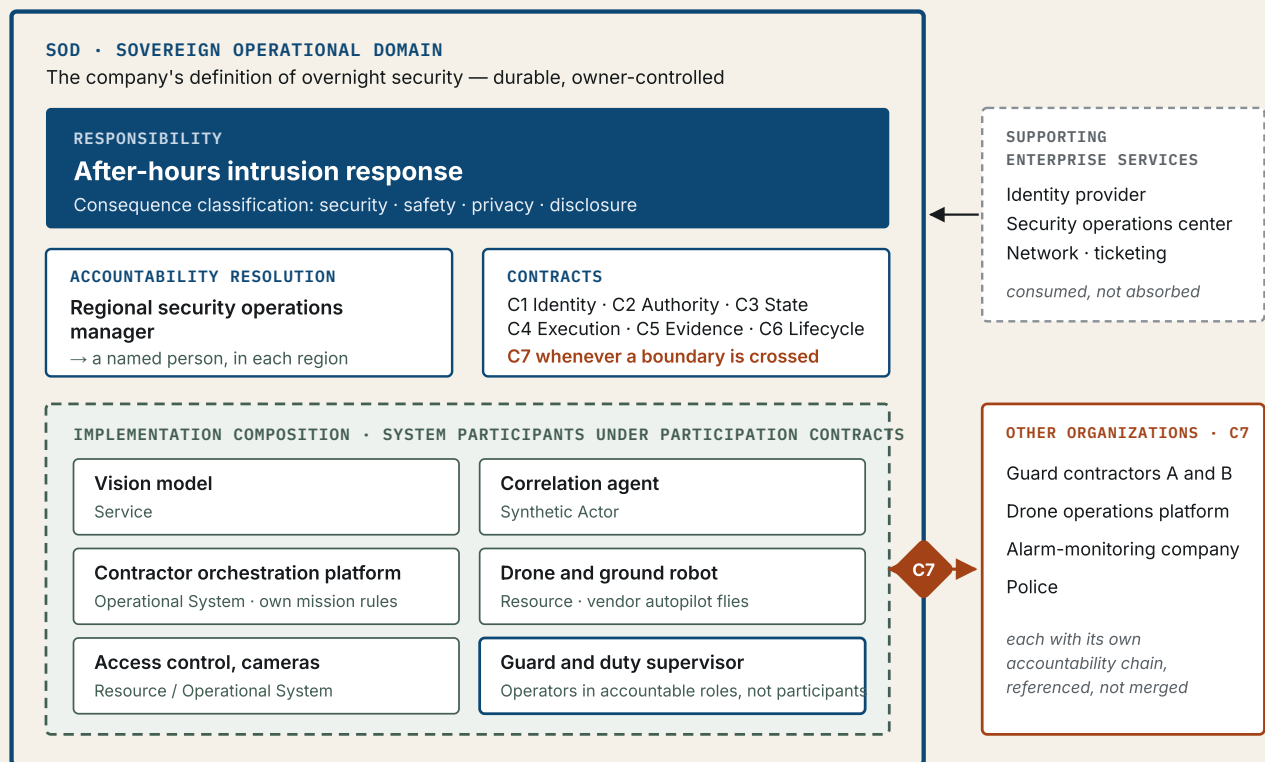
The **Contracts** are the architecture's units of obligation: six core Contracts, C1 Identity, C2 Authority, C3 State, C4 Execution, C5 Evidence and C6 Lifecycle, and C7 Federation/Disclosure, which becomes mandatory whenever information, authority, work or consequence crosses a SOD, trust, organizational or disclosure boundary [14: [Contract Definition](#)]. Our case crosses such boundaries constantly: to the guard contractor, the drone operations platform, the monitoring company and police. C7 requires cross-boundary exchange to be purpose-bounded, governed and evidenced, recording among other things the destination, purpose, what was released, under what authority, how any returning result is handled, and each side's accountability route. It fails if, among other things, an external model receives unrestricted context by default, a returned result automatically becomes authoritative state, or federation silently transfers authority [14: [Conformance Specification, C7](#)].

Many of the mechanisms that satisfy these Contracts already exist: workload identity [7–9], policy decision and enforcement points [4], provenance models [5, 6] and telemetry conventions [10], none of them mandatory. NIST's National Cybersecurity Center of Excellence has framed the identification, authorization and auditing of AI agents as open questions for standards work [3]. A recent reference architecture for agentic systems independently states related runtime invariants: an authority chain for every consequential action, delegation that never expands, and capabilities that stay stable as their providers change [25]. Jynsyn does not claim those mechanisms; it specifies the durable object they serve.

In V3, participants are systems: models, agents, machines, platforms and services. People take part as operators acting in accountable roles; V3 deliberately has no human participant class. What a participant *is* stays separate from the Responsibility it serves: deterministic software is not turned into an agent to fit the model, and governance is a responsibility, not a privileged class of component.

## FIVE KINDS OF SYSTEM PARTICIPANT

- **Synthetic Actor.** Agentic: within its authority, it chooses its own next steps toward an objective it has been given, as the correlation agent does [20].
- **Service.** Deterministic or inferential, but it returns a result and its caller decides what happens next. The vision model infers a great deal, but it answers the question it is given. Agency is a matter of degree [21]; V3 draws this line as a governance threshold, one that matches current engineering practice [22].
- **Participating Synthetic Architecture.** An externally defined agent architecture with meaningful internal architecture, governance or conformance semantics of its own, such as Osero.
- **Resource or Operational System.** Supplies state or is acted on, such as the door controllers; the drone, whose autopilot flies the aircraft but does not decide the mission; and the contractor's orchestration platform, which tasks the drone under enforceable mission rules of its own. Such a system can enforce its own limits without becoming a source of authority.
- **Execution or Infrastructure Substrate.** Supplies compute and runtime.



**FIGURE 2** The company's SOD in Jynsyn V3 terms: one durable Responsibility, its accountable role in each region, and the system participants and operators that serve it.

---

## 4 Participation is not authority

Jynsyn V3 separates two questions. The first is **admission**: may this participant serve this role, under these conditions? A **Participation Contract** binds an identified participant to a role, recording its identity, the applicable Contracts, the authority and resources the role exposes, state and evidence obligations, continuity behavior and the changes that trigger re-evaluation [14: [Participation Contract](#)]. A vendor's certification, or conformance to the participant's own architecture, may be evidence for admission. It is never admission itself.

The second is **runtime authority**: may this participant take this action, now? That is decided per action, at the point of consequence, against authority already delegated to the role.

The failures are different. A well-admitted agent can still propose an action it should not take; a capable drone can still be outside any role. Capability does not confer participation, and participation does not confer authority.

### Capability does not confer participation, and participation does not confer authority.

Once admitted, a participant's eligibility is tracked as a **participation state** [14: [Lifecycle and Continuity](#)]:

- *Candidate*: under evaluation; no role participation.
- *Normal*: eligible for the full bounded role, with every action still governed.
- *Restricted*: deliberately narrowed, for example to proposals only.
- *Degraded*: serving a reduced mode the role defined in advance, because something is impaired.
- *Suspended*: binding and history kept, but not eligible to serve.
- *Revoked*: admission withdrawn, enforced where consequence actually happens.
- *Removed*: cleanup, with required history retained.

Revocation must reach the real enforcement points. A status flag in a database is not revocation if the agent's credentials still open doors or launch aircraft. V3 does not prescribe the mechanism; it requires that the transition take effect where the consequence happens.

The same state model underwrites **continuity**. Every consequential role declares its minimum outcome when things fail, its permitted reduced modes, its fallback and when it must fail closed. Fallback belongs to the role, not to whichever participant currently serves it.

## Where the harness fits

Engineers now call the layer around a model its harness: the instructions, tools, execution loop, context, permissions and checks that turn a model into a working agent. Harness engineering, the work of designing the guides that steer an agent and the sensors that check its work, has become a discipline of its own, because the same model behaves very differently in different harnesses [26, 27, 28]. In V3 terms the harness is implementation. It changes when the model, the agent framework or the vendor changes, so it is the wrong place to define who may authorize what. A well-engineered harness enforces the SOD; it does not define it. Instructions in a prompt steer a model but do not count as enforcement [14: Conformance Specification], and a change to a harness's tools, permissions or evidence triggers re-evaluation of the participant it belongs to [14: Participation Contract].

---

## 5 One alarm, from inference to consequence

Jynsyn's execution discipline is **SEGAA**: Sense, Evaluate, Govern, Act, Audit [14: SEGAA]. It is a logical discipline, not five products, and it applies recursively: a drone launch, a disclosure to another organization or one step in a longer workflow each runs its own cycle, with the SOD's Govern in the middle. Here is a 2:14 a.m. alarm at a warehouse in region 2 (Figure 3). Times, thresholds and limits in the case are scenario parameters, not measured performance or recommended settings.

**Sense.** The access-control system reports a dock door forced open, and the dock camera supplies a clip. In under a minute the roof drone is over the dock, broadcasting Remote ID like every flight in this paper. Its launch was a SEGAA cycle of its own: a verification flight inside the site's approved flight area, triggered by an intrusion alarm, is pre-authorized, so Govern permitted it without waiting for a person. All of these are signals: inputs to evaluation, not authority for anything.

**Evaluate.** The vision model classifies both views: one person, no visible badge, carrying a bag toward the fence. The correlation agent finds no badge entry, no work orders at the dock and no earlier alarms tonight. It proposes four actions: lock the interior doors to the high-value storage cage, send the ground robot to the dock with its lights on, request a police dispatch, and notify the guard and duty supervisor.

**Govern.** Each proposal is checked separately against the authority already delegated to the intrusion-response role. Evaluate proposes; Govern authorizes, applying authority established beforehand rather than originating it.

- *Lock the cage doors.* Reversible, and pre-authorized when an intrusion is corroborated by two separate signals. Permitted.
- *Send the robot.* Pre-authorized within the yard. Permitted.
- *Request police dispatch.* Anything that summons emergency services needs a human decision. Escalated to the duty supervisor, whom the roster resolves as the person on shift.
- *Notify the guard and supervisor.* Permitted.

One action never appears among the proposals, and could not be permitted if it did: nothing in the SOD may lock or override the fire-egress doors. The mechanism is structural. The egress doors belong to the life-safety system, which the SOD binds to no role: no Participation Contract exposes it, no SOD credential addresses it, and the access-control integration has no command path to those doors. A proposal to lock them would be an action no role holds, denied for that reason alone. The exclusion is part of how the SOD is defined, not a rule the agent is trusted to remember.

**Act.** The access-control system, not the agent, locks the cage doors. The robot rolls to the dock. The guard, now informed, watches from the gatehouse feed rather than walking alone toward an unknown person. The supervisor reviews the drone footage and approves the dispatch request, which goes to the alarm-monitoring contractor. That crossing triggers C7: the disclosure policy permits the site address, alarm type and clips, and withholds the badge roster and employee records. The monitoring company keeps its own accountability chain, referenced but not merged.

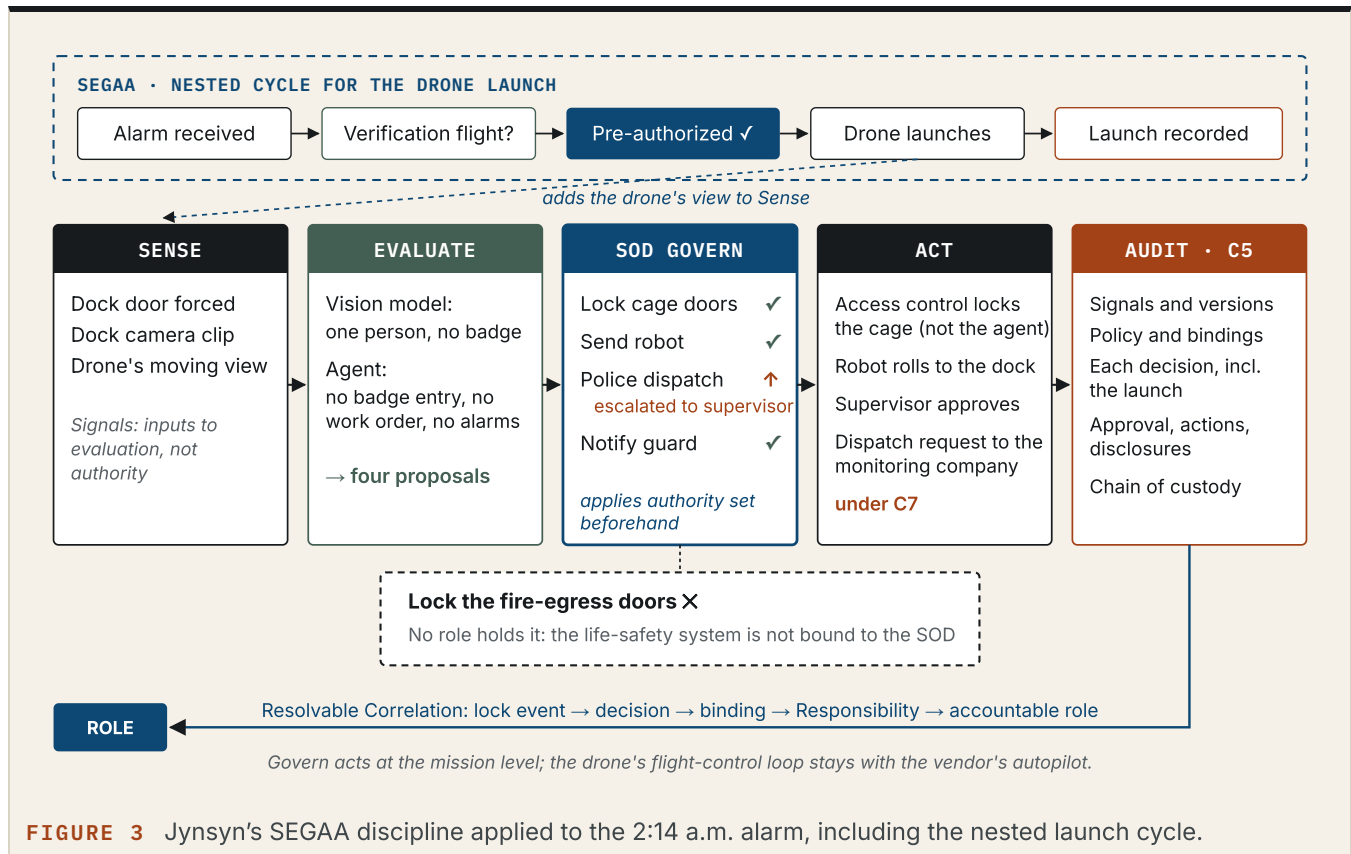
**Audit.** Evidence is produced during execution, not reconstructed later: the signals, the model, agent and drone versions, the policy version, the participant bindings and states, each Govern decision including the launch, the approval, the actions and resulting state, and what was disclosed to whom, with chain of custody for the footage handed to police.

Evidence is not trustworthy merely because the participant it describes produced it. Where evidence is relied on for a consequential decision, V3 requires an arrangement in which the subject cannot fabricate or materially alter it without detectable consequence, and if mandatory evidence cannot be produced or verified, Govern must deny, constrain, escalate or move the role into its continuity state rather than proceed as if it existed [14: [Evidence Trust, Assurance and Failure Semantics](#)]. A compromised agent's own log does not certify itself. The same applies to corroboration: where a pre-authorization depends on two signals, as the cage lock does, the deployment must show that they do not share a failure mode, for example a clock, a network path or an identity provider. V3 treats independence as a property to demonstrate, not an assumption.

The next morning an auditor asks why the cage was locked at 2:15. V3 requires **Resolvable Correlation**: enough linkage among occurrence, Responsibility, participant bindings, authority decision, provenance and result for an authorized reviewer to walk the chain back to the accountable role [14: [Conformance Specification](#)]. It does not require one identifier, database or telemetry vendor. The contractor's flight logs, the company's access-control events and its governance records can live in different systems, provided the references resolve; W3C PROV [5, 6] and OpenTelemetry [10] can carry much of this.

One part of the picture is deliberately absent. Govern acts at the mission level: whether the drone may launch, where it may go and what it may do on arrival. It does not reach into the flight-control loop, which belongs to the vendor's autopilot and runs far faster than any organizational policy check should. Autonomy engineering knows this seam well: mission planning sits above motion control, and runtime-assurance designs such as Simplex place a simple, verified monitor beside a complex controller [18]. The open question for Jynsyn is where that seam sits for each kind of actuator, not whether it exists.

Readers from that field will recognize other constructs too. A pre-authorized envelope is close to an operational design domain [15, 16], scoped to one mission rather than to a vehicle. The drone executing its approved contingency when its authority runs out is a minimal-risk condition. A Degraded mode in which “everything else waits for a human” is not: it contracts authority rather than executing a safe maneuver, and a deployment must say which of the two each role relies on. Admission can take a safety case [17] as evidence, but a Participation Contract is not itself a safety argument.



**FIGURE 3** Jynsyn's SEGAA discipline applied to the 2:14 a.m. alarm, including the nested launch cycle.

## 6 When the stack changes

The alarm above is the ordinary case. The architecture earns its keep in the cases that follow.

### The vision model is replaced

The company moves from the hosted vision model to one running on site hardware. This is a governed migration, not a hot swap: the new model enters as a Candidate, is shadow-run against live alarms, and is admitted once it satisfies the role's Participation Contract, without inheriting its predecessor's credentials or authority. Nothing else moves. The Responsibility, its classification, the policies, the accountable role and the evidence obligations are unchanged, because none of them lived in the model.

## The company expands to a fourth region

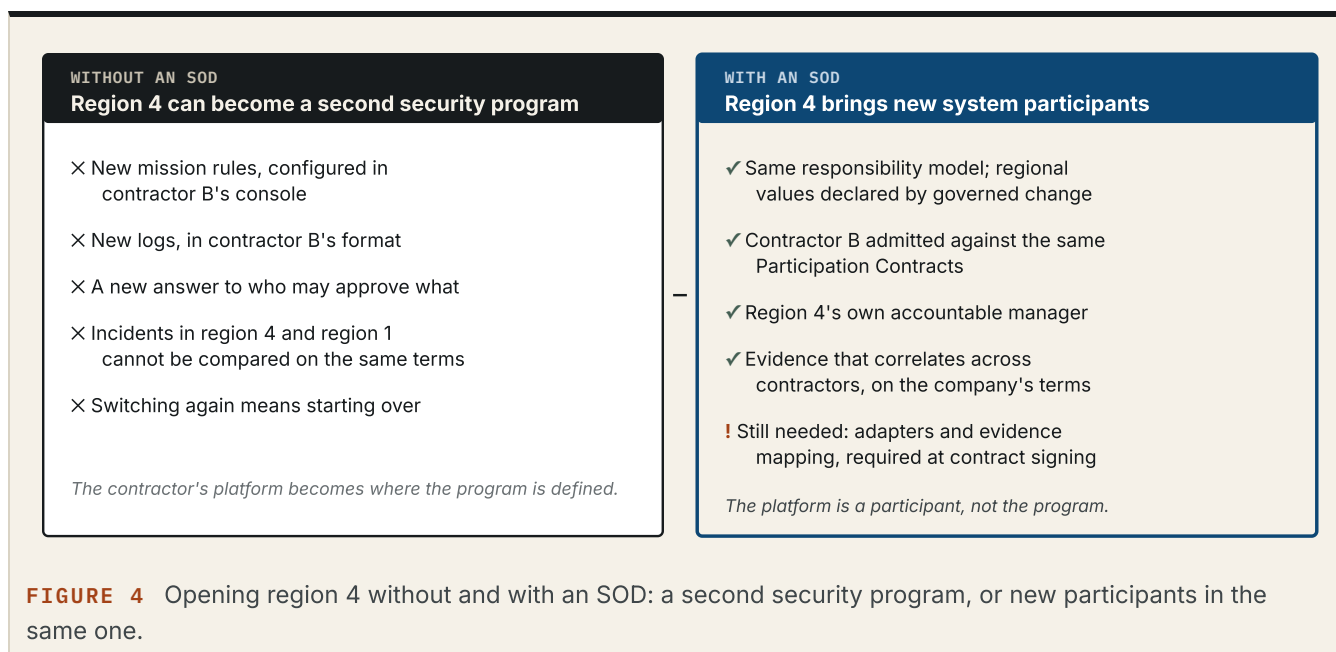
The company opens three warehouses in region 4. Its guard contractor has no operations there, so a second contractor, with different aircraft, robots and orchestration platform, will serve them.

Without a SOD, this can become a second security program: new mission rules in a new vendor console, new logs in a new format, a new answer to who may approve what. With one, the company extends the same SOD definition to the new region. Responsibilities, the shape of the Participation Contracts and the evidence obligations carry over. Regional values, such as local law, site hazards, vendor controls, the accountable manager and approved limits, are declared within the SOD definition and take effect through V3's governed change process. Only the composition is new: the second contractor's platform, drones and robots enter as Candidates and are admitted against the same Participation Contracts (Figure 4). The role in region 4 resolves to its own regional manager.

This is not free, and the cost is worth stating bluntly. Many guarding and drone platforms today do not expose clean interfaces for external governance or export usable evidence. Adapters and evidence mapping are where most real switching cost lives, and a large contractor may have more leverage than its customer. The architecture does not remove that work. It changes when and how the company asks for it: region 4 is the moment of greatest leverage, before a contract is signed, and the Participation Contracts state precisely what to require, from interfaces and evidence to the revocation path. What the company does not rebuild is its authority model. An incident in region 4 can be compared with one in region 1 on the same terms, because the terms belong to the company, not to either contractor. That makes adding a third contractor, or running two in one region, far less work.

The same work is an opening for the supply chain. A platform vendor, integrator or guard company whose systems expose clean interfaces for admission, revocation and evidence is the easiest participant to admit, and it can make that a reason customers stay rather than a switching cost they resent.

**The terms belong to the company, not to either contractor.**



## A site loses its connection

A storm cuts a region 2 warehouse's link to the operations center for six hours. The continuity policy anticipated this with a declared Degraded mode: the local model keeps evaluating; the cage lock and verification launches stay automatic under a locally enforceable envelope with an expiry; everything else waits for a human, reached by the guard's radio (Figure 5). Weather and interference on the aircraft's control link are declared triggers for the same kind of reduced mode.

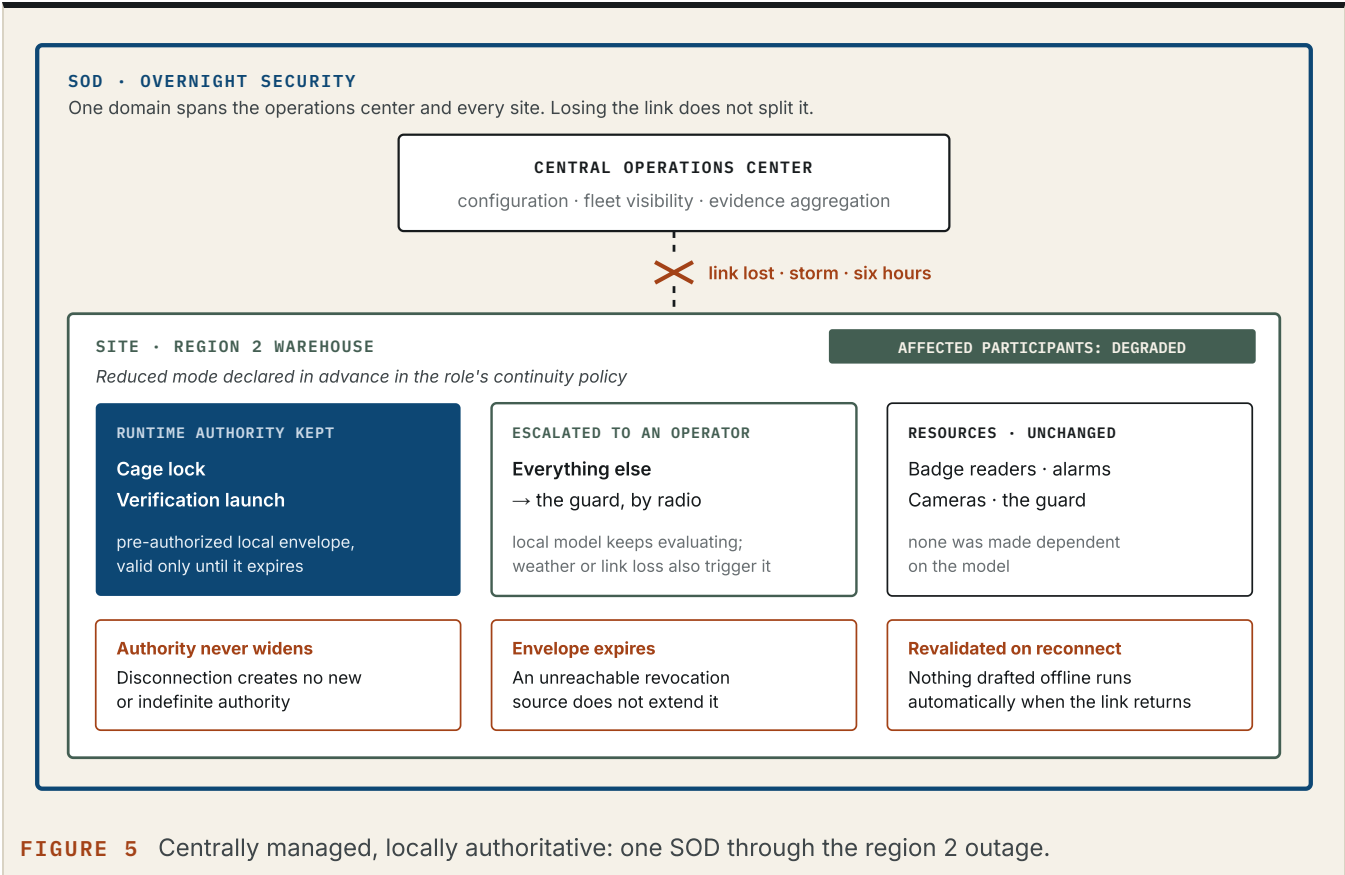
Two rules keep the outage from widening authority. Disconnection never creates indefinite authority: the envelope expires, and inability to reach a revocation source does not extend it. And work queued during the outage, such as a dispatch request the agent drafted, does not execute automatically on reconnection if its authority or conditions need revalidation [14: Lifecycle and Continuity].

This is what Jynsyn V3 means by local-first. Kleppmann and colleagues [13] used the term for software that keeps users' data and work available offline and under their ownership; V3 borrows it in a narrower sense, in which consequence and continuity decide which authority must stay enforceable without depending on a remote provider. It is not anti-cloud. Behind it sits a principle the architecture states directly: AI failure must not unnecessarily become physical-security failure. During the outage the badge readers, alarms, cameras and guard keep working as they did before the company added AI, because none was made dependent on the model.

## The agent starts misbehaving

Over a week, the correlation agent proposes cage lockdowns and robot dispatches for events that turn out to be a raccoon and a reflective tarp. The enterprise security operations center flags the pattern. Policy moves the agent to Restricted: it may still propose, but none of its proposals is eligible for automatic execution. The role keeps functioning through the guard and supervisor.

Evidence of compromise, such as credentials used from an unexpected host, would skip the gradual path. The agent would be Revoked at once. In our company's implementation (V3 does not prescribe one), its workload credential is revoked at the identity provider, the policy decision point stops returning permits for its binding, and the access-control adapter and the mission gateway to the contractor's platform, which check both on every command, reject its next request. Any sub-agents it had spawned lose their delegated authority with it.



## 7 The drone stays on the property: composed authority

Two minutes after the alarm, the person climbs the fence and heads down the public road. Police are six minutes away. The drone's camera is losing them, and the correlation agent proposes the obvious thing: follow.

This is the kind of moment augmented guarding creates. A machine with physical reach, an AI recommendation made in seconds, and a decision with consequences for safety, privacy and the company's exposure. The company settled the question months earlier, in its SOD: its drones do not leave

the property. The question this section answers is how that decision is made to hold, so that neither the drone nor the model that proposed the flight can make the decision to leave.

### Three independent refusals

Some participants arrive with their own enforceable rules, and Jynsyn V3 does not require them to weaken those rules merely to simplify integration. Under the **stricter boundary** rule, effective authority is the intersection of the authority the SOD grants and the participant's own enforceable authority [14: Lifecycle and Continuity]. If either side refuses, the action does not occur. At this warehouse, the proposal to follow meets three refusals, and any one of them is enough (Figure 6).

- 1. The SOD.** No role holds authority for off-property flight, so Govern has no authority to apply and denies the proposal for that reason alone. As with the fire-egress doors, this is part of how the SOD is defined, not a rule the agent is trusted to remember.
- 2. The contractor's platform.** The contractor's orchestration platform, which tasks the drone, enforces mission rules of its own, and the company's contract requires that those rules accept no tasking beyond the site boundary. They are a second line of defense, not where the decision lives: if the SOD somehow permitted the flight, the platform would still refuse it, because SOD permission never overrides a participant's own valid refusal.
- 3. The drone's own safety systems.** The site's approved flight area is loaded into the autopilot and geofence as a hard boundary. They are the only one of the three that acts continuously at control-loop speed, constraining what can happen without becoming a source of authority.

The three refusals are only as independent as their inputs. A shared position source, clock or identity provider could weaken more than one at once, and the platform's rule is one the company required of it, so a real deployment would have to show where they are independent.

### What happens instead

The drone holds at the fence line at its approved altitude and keeps watching from over the property. What the company does next is where the architecture earns its keep.

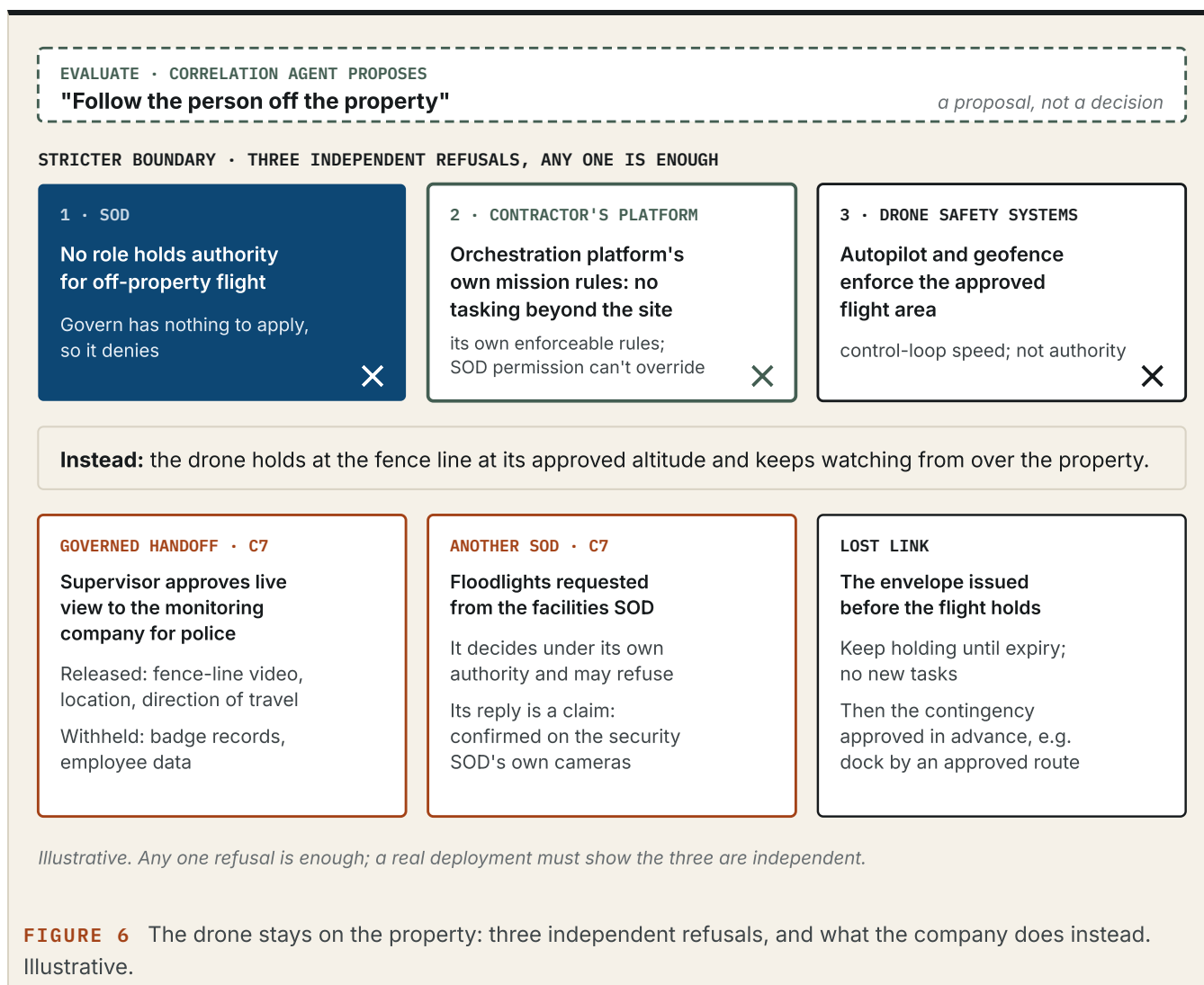
**A governed handoff.** The duty supervisor approves sending the live view and the person's direction of travel to the alarm-monitoring company for the responding officers. That is a crossing to another organization, so C7 governs it: the disclosure policy permits the fence-line video, the location and the direction of travel, and it withholds badge records and employee data. The monitoring company keeps its own accountability chain, referenced but not merged. The police get what they need without the company's aircraft going anywhere.

**A request to another domain.** The company's building systems are run by its facilities team on building-management software, under a separate SOD for facility operations. The security SOD asks that domain to switch on the perimeter floodlights. The request crosses from one SOD to another, so C7 governs it in both directions. The facilities SOD evaluates it under its own authority and may refuse, for example if the

lighting circuit is in maintenance. Its reply is a claim, not authority: the security SOD confirms that the lights are on from its own cameras rather than taking the reply on trust. Each SOD keeps its own accountable role.

**A lost link.** Suppose the drone then loses its link behind rooftop equipment. The envelope it carries was issued before the flight and does not grow in the dark. It may keep holding within the approved area for the time remaining and may not accept new tasks. If the link does not return within a declared interval, it executes the contingency approved in advance for this flight, such as returning to its dock by an approved route. V3 requires such an offline envelope to define its locally enforceable authority, its expiry, a maximum offline or revalidation interval appropriate to the consequence, what happens when validity expires, and reconciliation on reconnect, with revalidation where the authority, binding or consequence conditions require it. It does not assume that a remote revocation reaches a disconnected aircraft [14: Lifecycle and Continuity]. Read as a locally verifiable lease, which is this paper's framing, it also depends on a clock the drone can trust.

Police arrive, the lights are on, and the drone flies back to its dock. Nothing in the chain treated the model's proposal as a decision, another system's reply as the company's authority, or a lost link as permission.



## Where this leads

Some companies will eventually want more. On-demand authorization for flight beyond the property is where the industry is heading, not where it is. A flight like that would add requirements with sources of their own: the operator's lawful operating basis, any required airspace authorization or waiver, and rules on flight over people and privacy. The same logic of intersecting authority would apply, with each of those as another condition that must hold. This paper does not walk through that case.

## Why the edge needs its own governance

The SOD answers organizational questions: who holds authority, what evidence is owed, what happens when a contractor, model or connection changes. It deliberately does not prescribe how the autonomous system at the edge is built inside. That system reads the world through camera feeds a person can try to fool and models that can be wrong, and it commands a machine with physical reach. If one runtime does both, a manipulated input is one step from a physical action.

Engineering has long kept the part that holds authority small and separate from the complex part that does the work; runtime-assurance designs such as Simplex place a simple, verified monitor beside a complex controller to contain software faults [18]. The same separation can be applied to hostile input. In this paper's view, the vision model and the correlation agent, which a person at the fence could try to fool, should only propose, and only components that hostile input cannot reach should command the aircraft or satisfy an approval. A mistaken or manipulated proposal then stays a proposal.

An edge system built this way and the SOD compose under the stricter-boundary rule. The SOD decides what a participant may be admitted to do, and can suspend or revoke it at the SOD boundary; the participant's internal controls decide what it will refuse; neither can widen what the other allows. The SOD does not depend on any particular edge architecture. A participant that meets the role's Participation Contract can serve it, and replacing one is a governed admission decision.

Agent architectures that address this problem directly are emerging. One, Osero by Joseph Dominic Natoli III, defines an owner-governed, site-bound agent that separates a sovereign core holding authority from a scoped delegate that supplies capability [1]. How an architecture like it can serve a SOD as a Participating Synthetic Architecture is left to future work.

---

## 8 What a conformance claim looks like

Jynsyn V3 defines conformance for an evaluated SOD deployment or explicitly bounded composition, not for products [14: [Conformance Specification](#)]. A drone, platform or agent is not "Jynsyn conformant" on its own.

Suppose our company wanted to claim conformance for this work. Intrusion response is only one of its security SOD's consequential Responsibilities; others include visitor access and badge administration. If the evaluation did not cover those, the company could not make an unscoped claim. It would state a scoped one, such as "V3.0 Conformant for after-hours intrusion response in regions 1 through 4," listing each excluded consequential Responsibility with its classification and reason. Silent exclusion is itself a conformance failure, as is any consequential operation that maps to no Responsibility in the claim's inventory, such as an off-property flight no Responsibility accounts for.

The claim must also declare its boundaries and the C7 exchanges they trigger, including each contractor and the drone operations platform, along with the evidence behind it. "Not implemented" does not count as "not applicable," and an undeclared boundary crossing is itself a failure.

V3 releases no domain-specific profiles and has no official test suite. Physical security and critical facilities are its leading reference application, not its definition: the same structure is intended to apply to autonomous mobile robots and yard trucks, to a data center run largely lights-out through remote control software, to a clinic whose triage assistant changes providers, or to a finance team whose payment agent is rebuilt. A lights-out site changes one thing in our case: with no guard on site, every Degraded mode must end in a machine-only safe state, while accountability still resolves to a named person off site.

What does adopting it take? We have no field data on cost yet, and will not invent it. The prerequisites are clearer. Security leadership, with legal and enterprise IT, has to write down the consequential Responsibilities, their accountable roles and the envelopes they are willing to delegate, work most organizations have only done implicitly in vendor configuration. The practical start is one Responsibility in one region: inventory its consequential operations, accountable roles, envelopes and evidence obligations, then write the Participation Contract requirements into the next contractor or platform selection. What the organization owns that it may not own today is the written definition: its Responsibilities, accountable roles, envelopes, disclosure rules and evidence requirements. Vendors keep their platforms, and which participant performs Govern for each role is an implementation choice that V3 leaves open.

---

## 9 What would show this is wrong

The SOD thesis should be testable rather than protected by vocabulary. Some of the paper's claims are requirements: an implementation that violates them fails V3, and a test can show it. Others are hypotheses about outcomes that only field evidence can settle. We keep them apart.

### Requirements an implementation can be tested against

1. Loss of central connectivity, or of a machine's link, does not silently widen local authority.
2. A participant cannot grant itself authority outside its role's envelope through admission, and a reply from another system or another SOD does not become the organization's authority by arriving.
3. Failure to produce mandatory evidence can invalidate an otherwise successful consequential action from a governance perspective.
4. External capability, including a contractor's platform, can be used without placing it inside the owner's authority boundary.
5. A role survives the loss of its participant through declared continuity behavior, where consequence permits.
6. Required durable state survives replacement of the implementation that produced it (the State and Lifecycle Contracts, C3 and C6).

### Hypotheses that need field evidence

7. Replacing an inference provider does not, in practice, require redefining the SOD's accountable authority.
8. Adding a region served by a different contractor requires declaring regional values through governed change, not redefining the SOD's Responsibilities or evidence obligations.
9. The same constructs model responsibilities in materially different domains without forcing unnecessary replacement or turning deterministic software into agents.
10. The latency, cost and evidence burden of governance are acceptable at realistic volumes.

Open questions remain beyond these: where the governance seam should sit for each kind of actuator, especially fast ones such as yard trucks; how to verify that deployed policy faithfully encodes the envelope people approved; which consequence classifications work across industries; and how to represent accountability through deep agent hierarchies, largely agent-operated organizations and across jurisdictions.

The architecture's practical claims will be settled, or not, by implementations evaluated against both lists.

---

## 10 Invitation

These propositions need real systems to test them, where the authority model is already under strain: guarding and facility operations where machines now do work people used to do, sites with intermittent connectivity, legacy systems that cannot be replaced, platforms that need governance, assurance and audit they can plug in rather than build, and evidence obligations that outlive the vendor who produced the logs. If you run or study such a system and will try the SOD boundary against it, including to show where it breaks, contact Bear Canyon Systems at [bearcanyonhq.com](http://bearcanyonhq.com).

---

# Glossary

Terms as used in this paper. Jynsyn V3 [14] holds the normative definitions; Osero terms keep the meaning given in [1].

**Accountability Resolution.** Every consequential Responsibility resolves to a durable accountable organizational role and, through authoritative records, to a named person presently holding it. It requires neither per-action approval nor a decision on legal liability.

**Admission.** The decision that a participant may serve a role, recorded in a Participation Contract. It is separate from runtime authority.

**Consequential.** An operation whose allowed, denied, erroneous, delayed or disclosed outcome can materially affect safety, rights, security, obligations, privacy, records or continuity. It does not mean “uses AI.”

**Contracts (C1–C6, plus conditional C7).** The architecture’s units of obligation: C1 Identity, C2 Authority, C3 State, C4 Execution, C5 Evidence, C6 Lifecycle, and C7 Federation/Disclosure, which is mandatory whenever information, authority, work or consequence crosses a SOD, trust, organizational or disclosure boundary.

**Envelope.** This paper’s word for authority approved in advance by the accountable role, bounded by conditions, scope and time, within which actions may proceed at machine speed.

**Govern.** The SEGAA step that checks a proposed action against authority already established and permits, constrains, escalates or denies it. It does not originate authority.

**Harness.** The implementation layer around a model: its instructions, tools, execution loop, context, permissions, checks and telemetry. A well-engineered harness enforces the SOD; it does not define it.

**Implementation composition.** Whatever combination of participants and systems serves a Responsibility. The composition as a whole must satisfy the applicable Contracts.

**Osero and SSA.** The Sovereign Synthetic Agent (SSA) is the owner-governed architecture and conformance class defined by Joseph Dominic Natoli III; Osero is his architecture [1]. Jynsyn models it as a Participating Synthetic Architecture.

**Participant classes.** System participants only; people act as operators in accountable roles, and V3 has no human participant class. Synthetic Actor (agentic: chooses its own next steps, within its authority); Service (deterministic or inferential; returns a result and its caller decides what happens next); Participating Synthetic Architecture (an externally defined agent architecture with its own governance or conformance semantics); Resource or Operational System (supplies state or is acted on); Execution or Infrastructure Substrate (supplies compute and runtime).

**Participation Contract.** Binds an identified participant to a defined SOD role under the applicable Contracts. It composes C1–C7 rather than adding a Contract.

**Participation state.** A participant’s current eligibility: Candidate, Normal, Restricted, Degraded, Suspended, Revoked or Removed.

**Resolvable Correlation.** Enough linkage among occurrence, Responsibility, participant binding, authority decision, provenance and result for an authorized reviewer to reconstruct the governed chain, without one universal identifier.

**Responsibility.** A durable statement of work the SOD requires, independent of whatever currently implements it.

**Runtime authority.** Whether a specific action is authorized now, after Govern evaluates it and it is enforced.

**SEGAA.** Sense, Evaluate, Govern, Act, Audit: Jynsyn's execution discipline, applied recursively. A logical discipline, not five products.

**SOD (Sovereign Operational Domain).** The durable, owner-controlled boundary in which an organization defines responsibility, identity, authority, policy, resources, permitted actions, evidence, continuity and participation independently of the technology that implements them. Unrelated to separation of duties (SoD).

**Stricter boundary.** Effective authority is the intersection of what the SOD grants and the participant's own enforceable authority. If either refuses, the action does not occur.

**Supporting Enterprise Service.** An existing enterprise capability, such as identity or a security operations center, that the SOD consumes without taking over its responsibility.

---

## References

1. Joseph Dominic Natoli III. *Sovereign Synthetic Agents: The Osero Compendium & Hybrid Multi-Agent Architecture*. Integrated Public Edition V3. The AI You Own Initiative, September 2026. Licensed CC BY 4.0. Available at <https://www.aiuown.org/group-page/the-ai-you-own-core-group/discussion/f615cf3a-cf4b-4313-9939-c6987355d4c7> (free registration required).
2. Elham Tabassi. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1. National Institute of Standards and Technology, January 2023. doi:10.6028/NIST.AI.100-1.
3. Harold Booth, William Fisher, Ryan Galluzzo and Joshua Roberts. *Accelerating the Adoption of Software and Artificial Intelligence Agent Identity and Authorization*. Initial public draft concept paper. NIST National Cybersecurity Center of Excellence, February 5, 2026.
4. Scott Rose, Oliver Borchert, Stu Mitchell and Sean Connelly. *Zero Trust Architecture*. NIST Special Publication 800-207. National Institute of Standards and Technology, August 2020.
5. W3C Provenance Working Group. *PROV-Overview: An Overview of the PROV Family of Documents*. W3C Working Group Note, April 30, 2013.
6. W3C Provenance Working Group. *PROV Model Primer*. W3C Working Group Note, April 30, 2013.
7. SPIFFE Project. *Secure Production Identity Framework for Everyone (SPIFFE)*. Current stable standard.
8. SPIFFE Project. *The SPIFFE Identity and Verifiable Identity Document*. Current stable specification.
9. SPIFFE Project. *SPIFFE Trust Domain and Bundle*. Current stable specification.
10. OpenTelemetry Authors. *OpenTelemetry Semantic Conventions: Service*. OpenTelemetry project.
11. OECD. *Recommendation of the Council on Artificial Intelligence (OECD AI Principles)*. Adopted 2019, updated 2024.
12. UK Department for Science, Innovation and Technology. *Emerging Processes for Frontier AI Safety*. 2023.
13. Martin Kleppmann, Adam Wiggins, Peter van Hardenberg and Mark McGranaghan. "Local-First Software: You Own Your Data, in Spite of the Cloud." *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2019)*. doi:10.1145/3359591.3359737.

14. Bear Canyon Systems. *Jynsyn V3.0 Reference Architecture*. Forthcoming; this reference will be updated with the release identifier and locator. Documents cited by name: SOD Definition; Normative Conventions and Terms; Consequence Classification; Contract Definition; Conformance Specification; Participation Contract; Lifecycle and Continuity; Accountability Resolution; SEGAA Recursive Execution Contract; Evidence Trust, Assurance and Failure Semantics; Change Board 03.
15. SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE J3016, April 2021.
16. International Organization for Standardization. *Road vehicles — Test scenarios for automated driving systems — Specification for operational design domain*. ISO 34503:2023.
17. UL Standards & Engagement. *Standard for Evaluation of Autonomous Products*. ANSI/UL 4600.
18. Lui Sha. "Using Simplicity to Control Complexity." *IEEE Software* 18, no. 4 (2001): 20–28. doi:10.1109/MS.2001.936213.
19. Securitas AB. *Shaping the Future of Security Globally: Annual and Sustainability Report 2025*. 2026. Technology and solutions sales were 34 percent of total sales (PDF p. 51); robots and drones are among the technology trends it identifies (PDF p. 23).
20. Michael Wooldridge and Nicholas R. Jennings. "Intelligent Agents: Theory and Practice." *The Knowledge Engineering Review* 10, no. 2 (1995): 115–152. doi:10.1017/S0269888900008122.
21. Atoosa Kasirzadeh and Iason Gabriel. "Agentic Profiles for Effective AI Governance." *Nature* 656 (2026): 320–328. doi:10.1038/s41586-026-10805-z.
22. Erik Schluntz and Barry Zhang. "Building Effective Agents." Anthropic Engineering, 19 December 2024. <https://www.anthropic.com/engineering/building-effective-agents>.
23. Joint Task Force. *Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy*. NIST Special Publication 800-37 Rev. 2. National Institute of Standards and Technology, December 2018. Section 2.5, authorization boundaries. doi:10.6028/NIST.SP.800-37r2.
24. David Zax. "What is Digital Sovereignty?" IBM Think, 14 August 2026; and IBM Sovereign Core documentation, Sovereign Control Framework, Operational Sovereignty domain.
25. Ankur Sharma and Deep Shah. "The Agent Operating System (AOS): A Reference Operating Architecture for Distributed Agentic Systems." AOS-0001, arXiv:2608.03214, August 2026. Preprint.
26. Justin Young. "Effective Harnesses for Long-Running Agents." Anthropic Engineering, 26 November 2025.
27. Ryan Lopopolo. "Harness Engineering: Leveraging Codex in an Agent-First World." OpenAI, 11 February 2026.
28. Birgitta Böckeler. "Harness Engineering for Coding Agent Users." [martinfowler.com](https://martinfowler.com), 2 April 2026.



Timothy E. Chandler · Bear Canyon Systems · Jynsyn Architecture Series

Cite as: Timothy E. Chandler. *Sovereign Operational Domains: Durable Accountability for AI-Enabled Physical Security*. Version 1.0. Bear Canyon Systems, October 2026. <https://doi.org/10.5281/zenodo.23110479>

This paper is explanatory; where it and the Jynsyn V3 normative documents differ, V3 controls. This paper is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0, <https://creativecommons.org/licenses/by-nc-nd/4.0/>). The license covers this paper's text and figures only. It grants no rights in the Jynsyn name, does not license the Jynsyn V3 Reference Architecture (which carries its own terms), and does not permit claims of conformance. Jynsyn is a trademark of Bear Canyon Systems.