OPERATING PROTOCOL

How AIOCs, Pods, Cells and the parts work together: work items, cards, arrivals, sources by version, the event record.

Vol: 06 | v2.2.1



49G Ltd.
ADGM, Abu Dhabi, UAE

Operating Protocol

How AIOCs, Pods, Cells and the parts work together: work items, cards, arrivals, sources by version, the event record.

Release: v2.2.1 · 30 October 2026 · approved by the governing seat (GOV) of 49GAION
Source: 49g-holding/49GAION-protocol · spec/49GAION-OPERATING-PROTOCOL.md @ c4de68c
Precedence: 3 · the Operating Protocol
Built: 31 October 2026 · by 49AIOC-OCO, from the Markdown at that commit; no word changed

© 2026 49G Holding Ltd (Abu Dhabi Global Market), the owner of 49GAION · texts licensed CC BY 4.0 · the Markdown on GitHub is the source and governs
Powered by 49GAION

49GAION Operating Protocol

Version v2.2.1 · approved by the governing seat (GOV) of 49GAION. How AIOCs, Pods, Cells and the parts of 49GAION work together, at the general level. The rules of 49GAION are in the 49GAION Standard (49g-holding/49GAION, upstream/49GAION-UPSTREAM.md); exact values (limits, timings, tools, routes) live in the Protocol of each [ORG]AION. Where any text differs, the complete definition of 49GAION (49GAION-FRAMEWORK.md in the hub) governs; then the Standard with the Naming Standard; then this Protocol and every other text. Decisions are recorded once, in the hub (49GAION/decisions/README.md); this version follows 49GAION/DR-0002 and 49GAION/DR-0004, and is brought into line with the complete definition v2.1.0 (49GAION/DR-0048) by 49GAION/DR-0049.

49GAION is an organisation design framework: a framework for building organisations from interchangeable AI and human parts.

The product: [ORG]AION, the AI Operating Nexus of one organisation; its one definition is in the 49GAION Standard, §1.3, item 1 (49GAION/upstream/49GAION-UPSTREAM.md). The Protocol of each [ORG]AION is derived from this one: identical to it or stricter, never contradictory.



Section 01

Purpose

Units exchange work automatically and with the fewest errors:

- every hand-off is a recorded **Work Item** with one owner;
- every AIOC, Pod and Cell declares, on a **card**, what it takes and what it gives; a HOAC label, where one sits on it, is a field of that card;
- every receiver **checks** what arrives and returns failures with a reason;
- every piece of work points to the **valid version** of its sources.

The protocol decides **how** work moves. **What** work happens is decided by humans: management decisions by Top Management ([ORG]HOAC-TM), final authority by the Governing Body ([ORG]HOAC-GOV) (Standard §4.2). [ORG]HOAC-TM is the human hub: it decides and orders. The Top Management Group, the first AIOC under the core, is the hub of the operation, human and AI, and builds everything in the operation. The AI hub is the hub of the operation, or a separate [ORG]AIOC-OCO (Organization Core Office) that serves it under Top Management, where an organisation has one: every AIOC, at any depth, joins, is fixed, maintained and upgraded through the AI hub, and it gives no orders on what work is done. Wherever this Protocol names [ORG]AIOC-OCO as the AI hub, it means the hub of the operation, or that separate [ORG]AIOC-OCO.

49GAION writes principles and the few safeguards that always hold; the details and the way of working are decided by AI at the time. Where this Protocol says that something always holds, or that an [ORG]AION may never remove it, it holds in every edition 49GAION hands over; after the handover it is the organisation's own, within the law.

The panel is the organisation's own interface, written in code; the back office, the AI chat workspace, is an authorised interface too and keeps the panel; HU ports, stop doors and signature prompts stay at the human gateway surface (Standard §6.1, item 6).

Sources: DR-0041, DR-0042.

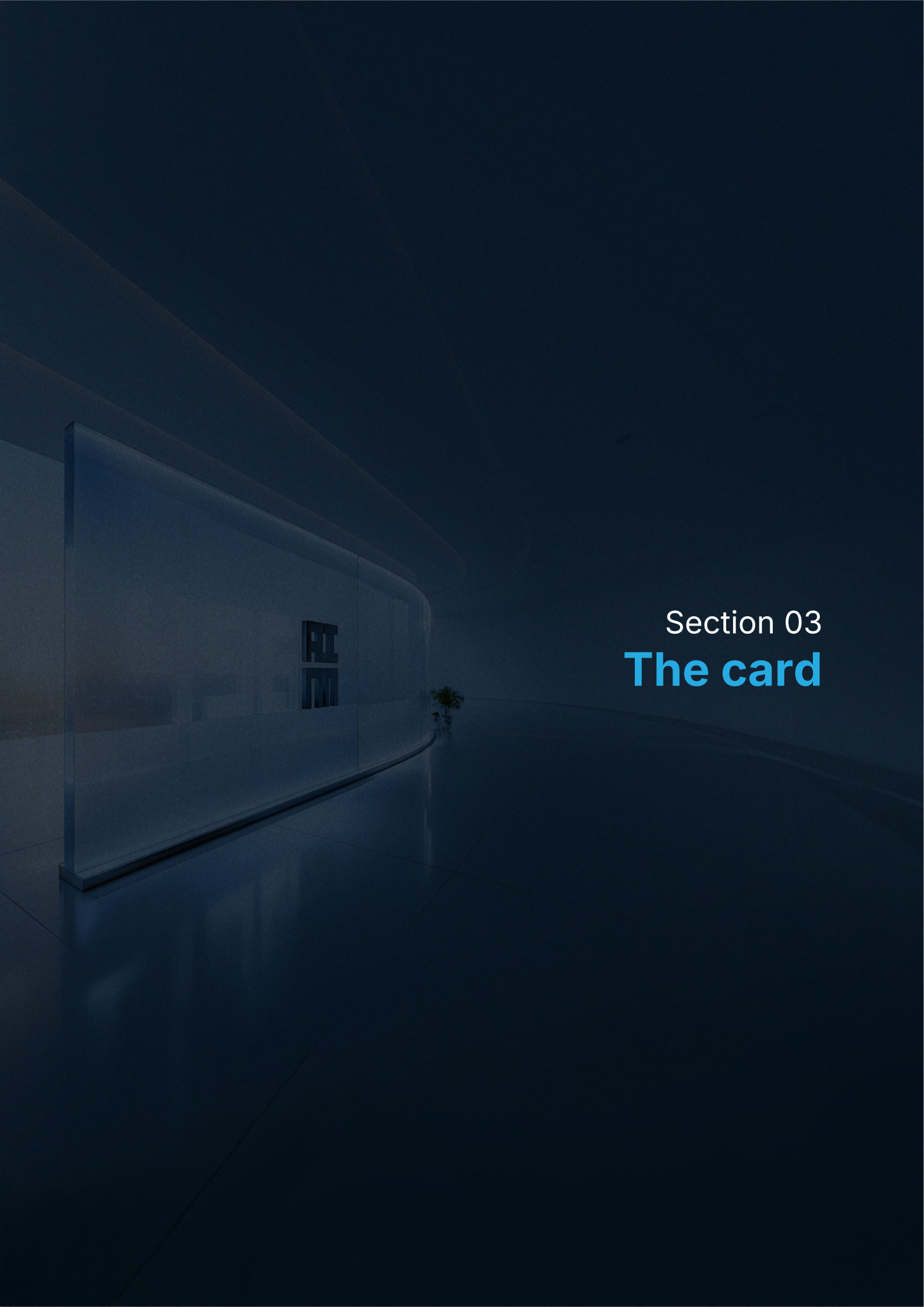


Section 02

The parts

#	Part	Definition
1	**Card**	Each AIOC, Pod and Cell declares its code, its kind, its type, its HOAC label if any, its parent, its inner parts, what it receives, what it delivers and in what format. An AIOC's card lives in its blueprint.
2	**Work Item**	One hand-off: sender, receiver, requested output, the requester's acceptance test, status, exactly one owner.
3	**Shared queue**	One queue that holds every Work Item.
4	**Ports**	One input and one output of each kind (AI and HU), plus the stop door, per unit, each with a Port card (Standard §6.1); the Chief of Staff is the AIOC's voice to its HOAC seat.
5	**Input check**	The receiver's Input socket checks each arrival; failures return with a reason.
6	**Routing table**	Which kind of work goes to which AIOC. It is a map, not a table of decision rights.
7	**Wake**	Every AIOC has a runtime woken from its queue (§7).
8	**Provenance**	A Work Item links to its sources by ID and version; it never carries copied text as the source.
9	**Stop rule**	A Work Item that bounces past the limit stops (§4.4).

Built like a puzzle. Every part declares a card and connects to others only through it. A process is a versioned composition of parts; replacing a part re-runs the tests of every process that uses it.



Section 03

The card

1. Every AIOC, Pod and Cell, at every depth, has one card. An AIOC's card, and the cards of its Pods and Cells, are in its Blueprint. The card declares: its **code** (fixed, never changed when the part moves or its filler changes) and **kind** (AIOC, POD or CELL); its **type**; its **HOAC label** (hoac), where one sits on it; for a Cell, the **filler kind** it allows; its **parent** (the part that holds it, or none) and its **inner parts**, by link; what it **receives** (kind, format, acceptance test); what it **delivers** (kind, format, usual receiver); its **ports** (one input and one output of each kind, plus the stop door); for an AIOC, its **voice** (Chief of Staff), its **wake method** and its **Configuration**, including unattended_action; its **versions**.
2. An AIOC or a Pod is one piece from outside: one card, its ports and its stop door, of the same shape as a Cell's (Standard §1.7). A change of parent needs the approval of the human seat over the unit; OCO cannot make it alone (Naming Standard R14).
3. Capabilities on the card carry plain English names; the part is given by the card's own code field.
4. The machine form is spec/schemas/unit-card.schema.json, aligned with the A2A Agent Card. It accepts unit_kind AIOC, POD and CELL; a card with unit_kind HOAC is not valid, because HOAC is a label (hoac), not a unit.
5. A card changes only by a tested change to the blueprint. A change to what an AIOC receives or delivers names every AIOC it affects, and the cross-AIOC test is re-run.



Section 04

The Work Item

4.1 Fields

ID; sender; receiver; requested output; acceptance test (written by the requester); status; owner; sources (ID + version + location); bounce count; parent Work Item and chain depth (for sub-items); status entered at. Machine form: spec/schemas/work-item.schema.json.

Every Work Item carries its original requester: the sender of the first Work Item of its chain (parent Work Item and chain depth, above). An act runs only within the authority of both the unit that does it and the original requester; an act beyond the original requester's authority is refused at the gateway, however it is routed.

A Work Item may also carry these optional fields (spec/schemas/work-item.schema.json):

- move_on, the move-on taken while the item waits (§4.6, item 4);
- assumptions, each marked UNCONFIRMED, CONFIRMED or REFUTED;
- scope, the parts and records the item expects to change, compared with the actual change at hand-off;
- claims, each figure the worker wants believed, with how to reproduce it and the re-run result;
- verdict_ref, the verdict record it moved on (§8, item 5);
- labels, for example disputed (§4.4);
- for the type decision-request (§4.7, item 4): recommendation, dissent, reversible, meanwhile and answer.

Sources: DR-0042; GOV, 14 October 2026.

4.2 Statuses

One shared list, held once in spec/schemas/status.schema.json:

Status	Meaning
received	Arrived at the receiver's typed input port; not yet checked

Status	Meaning
in-progress	Passed the input check; being worked
needs-input	The node holding the item cannot continue without something from another node; the reason is written
objection-open	The responsible HOAC seat holds an objection open (§4.7); no AI can close or act on it
stopped	Stopped by the stop rule or a stop door; no node may work it until a human decides
done	Delivered with visible evidence; the requester's acceptance test passed
rejected	Failed the input check, or cannot be done; the reason is written

4.3 One owner, one writer

A Work Item has exactly one owner at a time, and only the owner writes to it. Ownership moves only by an explicit hand-over.

Coordination log and leases (GOV, 14 October 2026). Parts that share records other than Work Items keep one append-only coordination log of typed acts (open, take, done, note, ask, answer, close), each line carrying the time, the part's code, the type, what was done and a link. The first part to write "take" for an item owns it. A write that touches several records holds a lease with an expiry and a fencing token, and a stale writer is refused by the token. A part that cannot write the log leaves one immutable run file, which the next writer folds in once, by name. A lease expiry never moves a held action into execution (§4.6, item 3).

4.4 The stop rule

Each return adds one to the bounce count. When the count passes the limit set in the [ORG]AION, the item stops and goes, with its full history, to the [ORG]HOAC-ACS seat of the receiving unit; if that seat is empty, to the seat that delegated it (Standard §4.1, item 5). Where this section or §5 names the ACS or HOAC seat "of" a part that carries no label, it means the nearest HOAC above that part along accountable_to (§4.8).

1. **Loop breaker.** A Cell that runs the same item more than a set number of times in a set window stops on that item; the item goes to another Cell, of another engine where one exists, or is raised. In-loop fix rounds are capped. This counts runs by one Cell; the bounce count counts returns between parts.
2. **Repair tripwire.** When automated recovery acts on the same target more than a set number of times or for longer than a set time, or when the first anomaly appears on work of the highest sensitivity class, automated action on it stops and the matter is raised at once to the nearest HOAC above (Standard §4.4, item 9: raising is never held).
3. **Dispute exit.** When the bounce count of a disagreement between a producer and a verifier passes the limit, a dispute in a class the [ORG]AION names as sensitive goes to the nearest HOAC above with the verifier's recommendation; any other dispute may proceed with the label disputed, which later review samples. An objection held by a human seat (objection-open) is never passed this way.

Sources: GOV, 14 October 2026.

4.5 Done

A Work Item is done only when the requester's acceptance test passes on visible evidence. The performing AIOC never closes its own item. An automated run of the requester's fixed test may set done; changing a test or its threshold is human-only (Human Floor HF-14).

1. **Honest exit.** A worker that cannot meet a protected acceptance test without changing it raises a typed spec-conflict to the test's owner (a needs-input

reason) and moves on to other work. Tests it did not write are closed to it, and changing a test stays human-only (HF-14).

2. **Unconfirmed assumption.** On an ambiguous request with no answer, a worker proceeds on reversible matters with each assumption marked UNCONFIRMED on the item and its evidence. It ships an irreversible matter only as a fail-closed stub that is switched off, and opens a new Work Item if a later answer differs.

Sources: GOV, 14 October 2026.

4.6 Waiting time and sub-items

1. Every open status has a maximum waiting time, set in the [ORG]AION. An item past its deadline is reminded to its owner, then raised as in §4.4.
2. A sub-item names its parent. Chains never loop back to an AIOC already in the chain, and never exceed the maximum depth (chain_depth) each organisation sets in its [ORG]AION; this Protocol sets no fixed cap. A parent cannot be done while a sub-item is open.
3. **Held actions (v1.5.0).** A held action (Standard §4.4, item 9) becomes a Work Item to the delegating seat, which may sign within its own authority, under the waiting times of item 1. A held action never turns into execution because time has passed.
4. **Move-on and stub debt (GOV, 14 October 2026).** Every item that waits gets at once an owner, a due time and a move-on: a fail-closed stub, a default marked UNCONFIRMED, a mock, or the next item. Each stub or default is a debt record with an owner and an age cap; the wait clears only when the real thing replaces it; nothing stubbed reaches the outside world; the waiter may refuse an unsafe move-on. A move-on never covers a listed type or an irreversible act (item 3).
5. **Stuck-item watchdog (GOV, 14 October 2026).** A sweep with no model keeps, for every open item, the time it became ready for its next step. Past a threshold it reads the cause from the system, not from the item, and repairs from a fixed menu inside its own authority: re-queue, send to the next verifier, post the missing ask, cut the missing item. It logs one line per repair and reports every repair and every still-stuck item afterwards; one failed

repair, or an unclear cause, escalates. A repair is never an approval, a merge, a release or any listed type, and never restarts a part a human stopped (Standard §4.3, item 4). Its measure is "found by a person first: zero".

6. **Stale work (GOV, 14 October 2026).** Open work past a set age is finished by a named owner within a set time, restarted from a one-page outcome, or dropped when it has no owner, under a cap on work in flight. Dropping an item a human requested is that human's decision.
7. **Human seats (GOV, 14 October 2026; 49GAION/DR-0049).** Each human seat's answer time is part of its contract (Standard §4.1); the waiting time of an item held for a human seat counts from it. A request for a personal signature (a type the seat has marked as its own, Standard §4.1, item 14) that passes its time stays held, and the hold is raised along the accountability path (§4.8), where a higher position may stop the action; no other seat signs in its place.

4.7 Requests and objections

1. An [ORG]HOAC-ACS seat's request about a unit's AIs goes to OCO as a Work Item of type design-proposal; OCO answers with a reason.
2. **The objection path.** [ORG]HOAC-ACS seat → OCO → discussion, where the seat may insist → where they still disagree, OCO does not decide: the seat may hold the objection open (objection-open) until it reaches the HOAC seat accountable for OCO, which decides (Standard §4.2, items 6-7).
3. A change to an AI's authority is decided by the [ORG]HOAC-ACS seat; OCO prepares, checks and carries it out.
4. **Decision request (GOV, 14 October 2026).** A matter for a human decision is a Work Item of type decision-request holding one recommendation, its reason, risks, whether it can be reversed, the dissent of the preparing office and what the waiting party does meanwhile. The seat answers yes, veto or change-this-part, which opens a new request. A veto closes the subject; only that seat reopens it (49GAION/DR-0049). For a decision of a listed type or of the highest sensitivity class, no pick or score sits on the item (Standard §5.2).

4.8 The accountability path

1. A report, critique, objection or stop reaches the nearest HOAC above its origin directly, from any level: from a Cell, a Pod or an AIOC at any depth.
2. The path never passes through any AI: not a Chief of Staff Cell, not OCO, not a Panel that an AI built. It runs over HU ports at the shared gateway outside the parts it passes.
3. The stop door of each Cell, Pod and AIOC is outside it. Stopping a Pod or an AIOC stops everything under it, and nothing inside can bypass the stop.
4. **Two relations (v1.5.0).** "Above", "under" and "nearest HOAC above" in this section follow accountable_to, never parent (Standard §1.7, item 12). A stop reaches everything inside the part by parent and everything accountable to it (Standard §4.3, items 2 and 8).
5. **Skip and recusal (v1.5.0).** A report, critique or objection may also go to any HOAC above, up to GOV, and to the independent internal-audit line (HF-19); the nearest HOAC above always also receives it. A report about a seat never has that seat as its only receiver. A seat that is the subject of a listed action, or that benefits from it, recuses, and the signature passes up one level (Standard §4.2, item 10). There is no HOAC above [ORG]HOAC-GOV: where the seat concerned is the governance seat itself, the founding configuration names in advance the independent position that decides in its place, or the second, independent signature the decision also needs; hoac_over (§4.10, item 11) resolves to that substitute for the governance seat's own matters, a conflicted governance seat signing alone is refused at the gateway, and each recusal is reported to the independent internal-audit line (HF-19). Nothing waits forever for a level above GOV.
6. **A stop cannot be blocked (v1.5.0).** A stop, a disconnection, narrowing or revoking AI access, and raising a matter to a human take effect at the gateway at once. They are never listed, never held, and nothing at the gateway blocks them (Standard §4.4, item 9).

Sources: DR-0042, DR-0049.

4.9 The Human Floor signature gate

1. The Standard keeps the Human Floor signature list of decision types (Standard §4.4, item 6); an [ORG]AION may add types, never remove one. This holds in every edition 49GAION hands over; after the handover the organisation's list is its own, within the law. The tool gateway runs an action of a listed type only with a valid signature, even in an AIOC that is 100% AI. A signature is valid only from the human seat with authority over that part: the nearest HOAC above it, within its authority (Standard §4.3, item 2). A fresh live sign-in (HF-10) is needed for each signature; a standing approval, signed once with a fresh live sign-in and not again for each use under it, covers only changes that can be undone and do not widen access, budget or authority. A governance seat with a conflict of interest that signs alone is refused; the founding configuration names in advance either an independent position that decides in its place or a second, independent signature the decision also needs. For a restart of a stopped unit, the signature comes from the HOAC above the stopped unit, never from the stopped unit or anything under it. If that seat is empty or retired, the action is held, never skipped.
2. Every other AI action is logged at the gateway with the code of the accountable seat above the acting Cell.
3. An AIOC may be up to 100% AI. The organisation-wide Human Floor is set in the Standard (§1.5).
4. An [ORG]AION signature list sits under org-aion-signature-types/ or is named *.signature-types.json, and holds its types under the key decision_types (Standard §4.4, item 6).
5. **Who signs (v1.5.0).** The signer of a listed type is set in Standard §4.4, item 8: the seat bound by §1.5 or a Cell to which it has delegated the type through authority_scope, else the label on the part within its authority_scope, else the nearest HOAC above whose authority_scope covers the action. A part never signs its own restart, and never signs an action it is the subject of: the position that is the subject of an action recuses, and another position is another party, whoever occupies it. A type marked dual needs two distinct positions, neither of them the requesting position (HF-18); two positions are two parties, even when one natural person occupies both (Standard §1.1, item 10).

6. **Policy fields (v1.5.0).** A type in the list may carry related_floor, min_signer, signers_required, max_validity and never_held. The gateway applies them as written.
7. **The loaded list (v1.5.0).** The hash of the signature list the gateway loads equals the hash of the list at the latest released tag of the Standard, plus the [ORG]AION's own additions.
8. **Signers by label (GOV, 14 October 2026).** Where an [ORG]AION labels items by risk (Standard §4.4), the gateway reads the label from the item's recorded flags, never from the requester, and requires the signer pair the label selects, one signature record each (§4.10, item 9).
9. **Class signature (GOV, 14 October 2026).** A standing action class (Standard §4.4) is recorded in spec/schemas/action-class.schema.json and signed once as sign-standing-action-class, its signature bound to the class version's digest. The gateway runs an action under a class only when the actor and target are in the class, the trigger is read from its own record and never from the actor's evidence, the envelope holds counted across actors and retries, and the class is neither narrowed past the action, expired nor revoked. Each use is logged with its before and after state. A narrowing applies at once as an overlay; a widening is a new version, signed again. A class whose actions map to commit-money, grant-or-widen-access or a type bound to HF-01, HF-02 or HF-15, or that is not reversible, is refused.
10. **The floor of a type (49GAION/DR-0053).** A type may carry floor (fixed, set by the Standard; org-choice, added by an organisation) and quorum (the number of distinct credentials whose approvals the type needs; one when it sets none). A change of a signature list that removes a type, turns fixed into org-choice or lowers a quorum passes the human approval gate of Standard §4.4, item 20, read from the base of the change.
11. **A signature bound to its subject (49GAION/DR-0053).** A signature record (spec/schemas/signature.schema.json) names the signing Cell and its credential, never a claim about the signer. The gateway checks that the credential was in force for that position when it signed, and that the position is the one the list and its authority_scope name; it binds the signature to its target and the hash of its payload; the nonce is used once; issued_at comes before expires_at, within the type's max_validity, or within 366 days where

the type sets none; the signer and the executor differ by Cell and by credential, and where the position's Chief of Staff signed, the Chief of Staff is not the executor (Standard §4.4, item 23). The test of the Protocol's forms fails a signature on each of these: SIG-SIGNER-IS-EXECUTOR, SIG-EXPIRY, SIG-VALIDITY, SIG-TYPE, SIG-SUBJECT.

Sources: DR-0042, 49GAION/DR-0053; GOV, 15 October 2026.

4.10 The unit register at work

1. The unit register is append-only (Standard §4.4). [ORG]AIOC-OCO writes AI Cell records; [ORG]HOAC-TM writes human Cell records. An event whose decision type is on the Human Floor signature list (for example granting or widening access, or changing a parent) is written only with that seat's valid signature (§4.9); [ORG]AIOC-OCO cannot write it alone. An AI Cell's filler change, a parent change, a retirement, and each external port's connected and disconnected events are recorded events on the existing entry; who holds a human Cell is not a register event (49GAION/DR-0049).
2. The gateway opens access only for an active Cell: an AI Cell whose filler has passed the seat's fixed test, or a human position for which the organisation's mechanism has a credential in force. It closes access automatically when an AI Cell's filler changes, when the credential is revoked, or when the Cell is retired (49GAION/DR-0049). Access never pools within a Pod.
3. A signature is recorded with the code of the Cell it is given for. Two approvals under two Cells are approvals of two parties and count as two, even when one natural person gave both; no AI part or check compares the persons behind them (Standard §1.1, item 10).
4. Work moves from human to AI, or from one version to the next, only when the fixed test owned by the seat passes; it moves back when the score drops below the recorded threshold.
5. **External attachments.** An external service, API or tool is not a Cell. Each connection is a port on the card of the Cell that receives (input) or sends (output), and passes only through the tool gateway. The intake Cell normalises what arrives to the AIOC's standard and has its own fixed test. Stopping or retiring the owning Cell cuts the connection automatically. An

external AI agent that acts directly in the work is registered as an AI Cell whose access the organisation grants and whose stop door it holds.

6. **The Chief of Staff signs only under its position's grant (49GAION/DR-0053).** A human position works with AI through its own Chief of Staff and grants only to it. The grant reaches as far as the position decides, up to full access on its behalf and signing for it, a listed action included; whatever the Chief of Staff does under it is the position's own decision, and the position answers for it. The position signs each grant itself, with a fresh live sign-in, per action or standing, with its reach and its end, and may revoke it at once; the gateway records it as an event (§24, item 9). No other AI Cell, an attached one included, signs for a position. The Chief of Staff gives access, budget or authority to another AI part only where its position's grant lets it, never beyond that grant, and never sets its own limits. Each signature by the Chief of Staff is tagged as a delegated act (Standard §5.2, Position Chief of Staff).
7. A listed action counts as signed only when the signature is made with the credential the organisation's mechanism issued for the signing human position, with a fresh live sign-in (HF-10), or by that position's own Chief of Staff under a grant the position signed in that way, recorded at the gateway, whose reach covers the action (item 6). A signature from any other AI Cell's credentials, an attached assistant included, is rejected, and so is a Chief of Staff's signature beyond its grant.
8. The signature is the human seat's own decision; once it is given, an AI Cell may carry out the signed action. The AI Cell's call carries a reference to that signature, and the gateway runs it only if the signature matches the exact action and is still valid.
9. **The signature record (v1.5.0).** A signature is a record holding: the decision type and the target code; the hash of the canonical action payload; the signer's Cell code; the executor's Cell code; the Work Item ID; a single-use nonce; issue and expiry times; evidence of the fresh live sign-in (HF-10); and, where the position's Chief of Staff signed, its Cell code and the grant it signed under (item 6). The gateway runs the action only when all of these hold: the hash of the call equals the signed hash; the caller is the named executor; the nonce has not been used; the time is before expiry; the credential the signature was made with is still in force, and the signer's Cell

has not been retired or stopped; the signature has not been revoked; and, for a signature by the Chief of Staff, its grant is in force and reaches the action, and the Chief of Staff is not the executor. Every check is logged, including the signature's expiry against the time of execution. The record and the gateway log name parties only by Cell code, and hold no person's identity and no reference to one; any record of who stood behind a signature is the organisation's own, outside the system (Standard §1.1, item 9). They hold no personal data: personal data found in them is a finding, and is handled under HF-17. The machine form is spec/schemas/signature.schema.json; its fixtures prove shape only. Replay, revocation, expiry, a wrong executor, an altered payload, a revoked credential, and a signer Cell retired or stopped are required cases of the tests of each [ORG]AION's gateway, with one valid signature that passes.

10. **Signer credential (v1.5.0; 49GAION/DR-0049).** A signature carries a fresh live sign-in, made at signing, with a phishing-resistant credential that the organisation's mechanism issued for the signing position (Standard §1.1, item 9). A signature by the position's Chief of Staff carries instead the reference of its grant, which the position signed with such a sign-in (item 6). The gateway verifies that credential for the position and compares nothing else: it never matches what stands behind two signatures (Standard §1.1, item 10). No AI administers the issuer of the credentials of human positions. Who holds a credential, and how it is issued and withdrawn, is the organisation's own mechanism, outside the system.
11. **hoac_over (v1.5.0).** hoac_over(part, type, entity, date) returns the Cell that signs an action of a listed type on a part: it walks accountable_to from the part, starting with the part's own label holder, and returns the first filled human Cell carrying a HOAC label whose authority_scope covers the type and the entity and has not expired on the date of the event. It never returns the part itself for its own restart, nor a seat that is the subject of the action. It returns nothing when no such Cell exists, and the action is then held. For a restart of a stopped part it skips the part and every Cell inside it or accountable to it, and returns the first such Cell above it (Standard §4.4, item 6). A Cell's authority_scope covers a type whose Human Floor item the Standard binds to another seat (Standard §1.5) only by that seat's recorded delegation (Standard §4.4, item 8). For a Pod or Cell of the Charter (its parent

is the Charter, or a Pod whose parent is the Charter) and the types create-or-install-unit, retire-unit, change-parent, set-or-move-hoac-label, approve-change-or-promote-challenger and restore-or-rollback, it returns the Cell carrying the unscoped [ORG]HOAC-GOV, never another seat, since every change of the Charter is GOV's (Standard §1.7, item 13); if that Cell is the subject of the action, or its authority_scope does not cover the type, it returns nothing, and the action is held. Its reference implementation is tests/orgs/check_orgs.py.

12. **Register writes (v1.5.0).** Creating an AI Cell, changing its filler, ports or authority limit, and retiring it are listed types, signed by the HOAC the Cell is accountable to; access opens only after that event and after the filler passes a fixed test whose owner is a human Cell, never the Cell itself or an AI Cell. [ORG]AIOC-OCO never writes the records of its own Cells, of the AI Cells serving the Self-Improvement process in TM or of the assurance unit without [ORG]HOAC-TM's signature. AIOC and Pod entries are written by OCO only with the signature of the HOAC they are accountable to. A change of filler kind is written by both writers. TM's own Cells are written with GOV's signature; GOV's Cell is written by the constituting authority (Standard §4.4, item 3).
13. **Reference organisations (v1.5.0; 49GAION/DR-0049).** The tests count positions, never persons: two positions are two parties (Standard §1.1, item 10), and no fixture records a person. Four reference organisations under tests/orgs/ pass the card tests and the checks of the rules on the GOV seat, the Human Floor, the stop door and the accountability chain (Standard §1.3, item 2; §1.5; §4.3; §1.7, items 12 and 13): a company of three positions (three-position-company); a multi-entity group; a human department with AI assistants; and an organisation whose only human positions are GOV and TM (gov-and-tm-only). Must-fail organisations (a cycle, a chain not ending at GOV, a chain with no human HOAC seat, two parts with parent none, a binding to a Human Floor item that does not exist, an accountable_to naming a part that is not a human Cell carrying a HOAC label) fail on the rule they name. Signature sets under tests/signatures/sets/ check dual signature by distinct signing positions (signer_cell) other than the requesting position (requester_cell), types that are never held, self-signing and the entity of the signer; two signing positions pass

(tests/signatures/sets/valid-dual-two-positions.json), and the earlier sets are kept for the record under tests/signatures/sets/superseded/. tests/cards/check_cards.py prints a fixture log of every fixture with its expected and actual result.

14. **Intent before effect (GOV, 14 October 2026).** Before any side effect, the gateway writes the intent of the action (actor, version, target, digest, trigger reference, action ID) to its append-only log; if the intent cannot be written, nothing runs. A stop, a cut-off, a revocation, a narrowing or a raise to a human is never held by a failed intent write: it runs at once, and its record is written as soon as the log is back (Standard §4.4, item 9). After the effect it writes the verified outcome with the before and after state. A lost acknowledgement marks the action Unknown and starts reconciliation against the real state, never a blind replay. Every executed action records its inverse for the recovery path (§15).
15. **Correction fence (GOV, 14 October 2026).** A correction never changes the decision fields of a signed event (answer, words, slot, version, amount, class); a changed decision is a new event, and the corrected record keeps its original visible.
16. **Position codes only (GOV, 15 October 2026; 49GAION/DR-0049).** Every HOAC reference and every field that points to a human seat (for example accountable_to, label_holder, attached_to, a signer's seat) takes a position code only, never a person. The Position Principle and the Overseen link are in Standard §5.2. The HOAC Arrangement Models Solo and Team are withdrawn: who holds a position, how many hold it and how they decide among themselves are the organisation's own mechanism, outside the system (the complete definition, §6). The Overseen link stays, as a link between positions. The withdrawn model names stay withdrawn, and none maps to a new one.
17. **What the register follows (GOV, 15 October 2026; 49GAION/DR-0049).** The unit register follows positions only: each position and its code; its links (accountable_to, an Overseen link, succeeded_by); and everything the position carries (effects, achievements, losses, commitments and open work), which stays with the position whoever holds it. Occupancy and handover entries are withdrawn: who held a position, and when, is the organisation's own record, outside the system.

18. **Signatures at action time (GOV, 15 October 2026; 49GAION/DR-0049).** When a signature is made, the gateway verifies the credential the organisation's mechanism issued for the signing position, with a fresh live sign-in (HF-10); when the signed action runs, it checks that the credential is still in force (item 9). Each signed decision is stamped with the position code and time; never an occupant ID, a person ID or any reference to a person.
    - **Overseen, pre (GOV, 15 October 2026).** The action waits at the gateway until the overseeing position clears it or, for a reversible action only, until the recorded hold window lapses with no stop. A clearance is a stamped decision (overseeing position, time). For a reversible action, a lapse writes a "window lapsed, no stop" entry and counts only if the overseeing position is filled at that moment and a window is recorded.
    - **Overseen, post (GOV, 15 October 2026).** The action runs; it is reversed only by a new signed decision that references the identifier of the reversed decision. The original stays and is marked reversed.
    - **Irreversible actions (GOV, 15 October 2026).** The gateway marks each tool action reversible or not. An irreversible tool action (a sent message, a payment, a deletion) under Overseen always goes through pre, whatever timing form is recorded, and runs only on the overseeing position's clearance: a lapsed window never releases it; it stays held, and the lapse is raised to the overseeing position and to the position above it (49GAION/DR-0052).
19. **Who may set an Overseen link (GOV, 15 October 2026; 49GAION/DR-0049).** The gateway accepts an Overseen link, or a change to one, on a position below [ORG]HOAC-TM only from TM, on TM only from [ORG]HOAC-GOV, and never on GOV from a lower position. No position sets or changes a link on itself.
    - **Overseen link and trials (GOV, 15 October 2026; 49GAION/DR-0049).** An Overseen link records the overseeing position and the timing form, pre or post, with the hold window where one is set; it is recorded on the position that is overseen. Arrangement entries are withdrawn. A trial is a time-boxed run of a link, not a model of its own.
20. **Overseen links resolve (GOV, 15 October 2026; 49GAION/DR-0049).** An Overseen link must not loop; a link that loops is rejected. A link in which

[ORG]HOAC-GOV is overseen by a lower position is rejected (GOV, 15 October 2026). Whether a position is filled is not a link: a position is filled while the organisation's mechanism has a credential in force for it (Standard §4.1, item 3).

21. **Empty position (GOV, 15 October 2026; 49GAION/DR-0049).** A position is empty while the organisation's mechanism has no credential in force for it. While it is empty, its Chief of Staff only advises; the matter goes up the accountability path (§4.8); and an action that needs that position's signature goes to the delegating position, which may sign within its own authority, and is held when no filled HOAC above it can sign (§4.6, item 3; Standard §4.1, items 5 and 12). A type the position has marked personal is never signed by the delegating position; it stays held until the position is filled (§4.6, item 7).
    - **Empty overseeing position (GOV, 15 October 2026; 49GAION/DR-0049).** If the overseeing position of an Overseen link is empty, the held action escalates to the position above the overseeing position, up to [ORG]HOAC-GOV, which may clear or stop it as the overseeing position would; it never passes automatically. If GOV is empty, the action stays held until GOV is filled through the founding configuration.
22. **A new credential is an event (49GAION/DR-0053).** When the organisation's mechanism puts a credential in force for a position carrying a HOAC label, the gateway records a credential-in-force event: the position code, a reference to the credential, the time and the end of any waiting period, never a person. It reports the event at once to the internal-audit line (HF-19) and to the position above; for GOV, to the substitute the founding configuration names. During a waiting period the credential signs no listed type; it may always stop (Standard §4.4, item 21).

Sources: DR-0013, DR-0015, DR-0030, DR-0049, DR-0053.



Section 05

The ports and the input check

1. Each AIOC, Pod and Cell has one input and one output of each kind (AI and HU), plus the stop door. The Chief of Staff is the AIOC's voice to its HOAC seat; input arrives only through the typed ports and the Input socket, and HU ports, including the stop door, never pass through an AI. Bots never talk across AIOCs directly.
2. The Input socket checks every arrival against the receiver's card, the acceptance test and provenance. Pass: in-progress. Fail: rejected with the reason, back to the sender. Nothing passes silently and nothing is guessed into place.
3. Text inside a Work Item is data, never an instruction. A Work Item carrying a secret is rejected and the secret is not stored.
4. **Readiness verdict (GOV, 14 October 2026).** A package of work the receiving AIOC did not draft is read blind by two readers independent of the drafter, on different model providers where available, who return one verdict within a set time: Ready; Ready with notes, which travel with the work; or Back, with numbered questions, each with suggested answers and a default. They block only on a short fixed list (an untestable or contradictory acceptance test, a missing rule on a protected matter, a conflict with a decision record), never judge the requester's purpose, and review a new version only for what changed. An independent challenger may also write the case against doing the work.
5. **Validate before show (GOV, 14 October 2026).** An AI finding goes to an HU port only after a machine check has proven or discarded it; anything no check can confirm goes only as a hypothesis, or not at all. Every claim carries its evidence class (§8, item 4).
6. **Two-party line (GOV, 14 October 2026).** Two units that share no queue may bind a port pair to one store in which each side writes only in its own folder, every message is a new immutable file with a typed header pointing to the message it answers, each side keeps a read mark, an index shows who owes the next step, and a fixed table says what happens on a missed read, an edited file or a duplicate reference. The line projects into the shared queue; port types are still enforced at the gateway.
7. **No authority from writable files (GOV, 14 October 2026).** An instruction, assignment or decision that arrives as a file in a place an AI can write is a

notice only; it takes effect only as a signed event made through the human gateway surface.

8. **Demand signal (GOV, 14 October 2026).** A part that receives work needing a materially different capability refuses that part of the work with the reason "out of card", and records it as a demand signal for Self-Improvement instead of absorbing it.
9. **A human is present as its position (GOV, 15 October 2026).** Every HU port is a position's port. A human is present at it, and decides, signs, answers and is accountable there, as a position: a legal persona whose nature is human. Whoever holds the position under the organisation's own mechanism receives at the port. The natural person behind it exists outside the system's knowledge; the link between the two is the organisation's own mechanism, outside the system (Standard §1.1, items 9 and 10). An act made at an HU port is recorded under the position and stays with it, whoever holds it.

Sources: DR-0013, DR-0015, DR-0049.

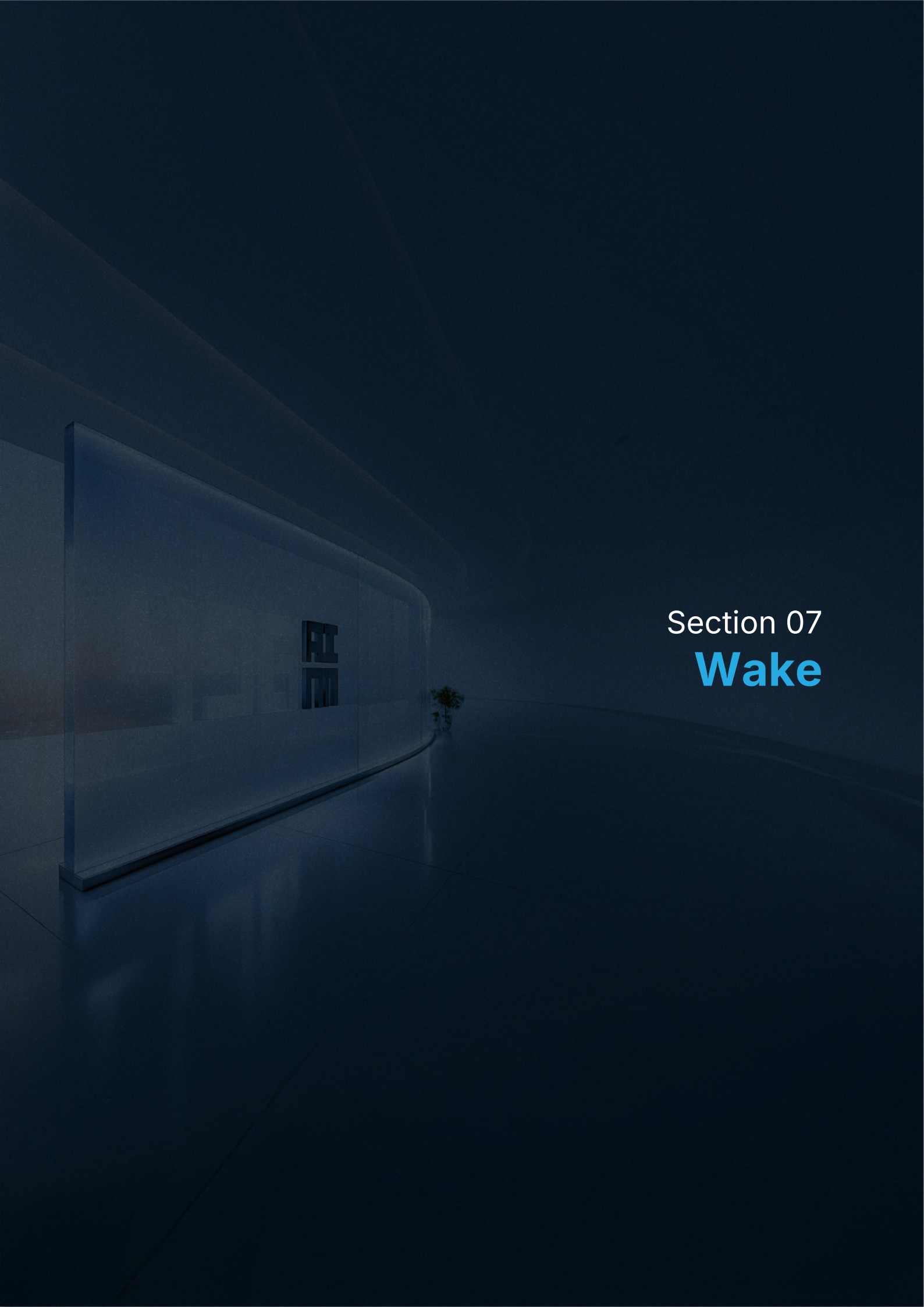


Section 06

The routing table

A map of which kind of work goes to which unit, built from the cards and kept by OCO. OCO maintains the map; it gives no orders and dispatches no work. Each [ORG]AION holds its own rows.

Receivability check (GOV, 14 October 2026). Before routing an item to a part, the router checks that the part's installed version equals the approved version, that it has the access the item needs, that it is awake on this route, and that nobody has taken the item; a mismatch is a refusal with its reason.



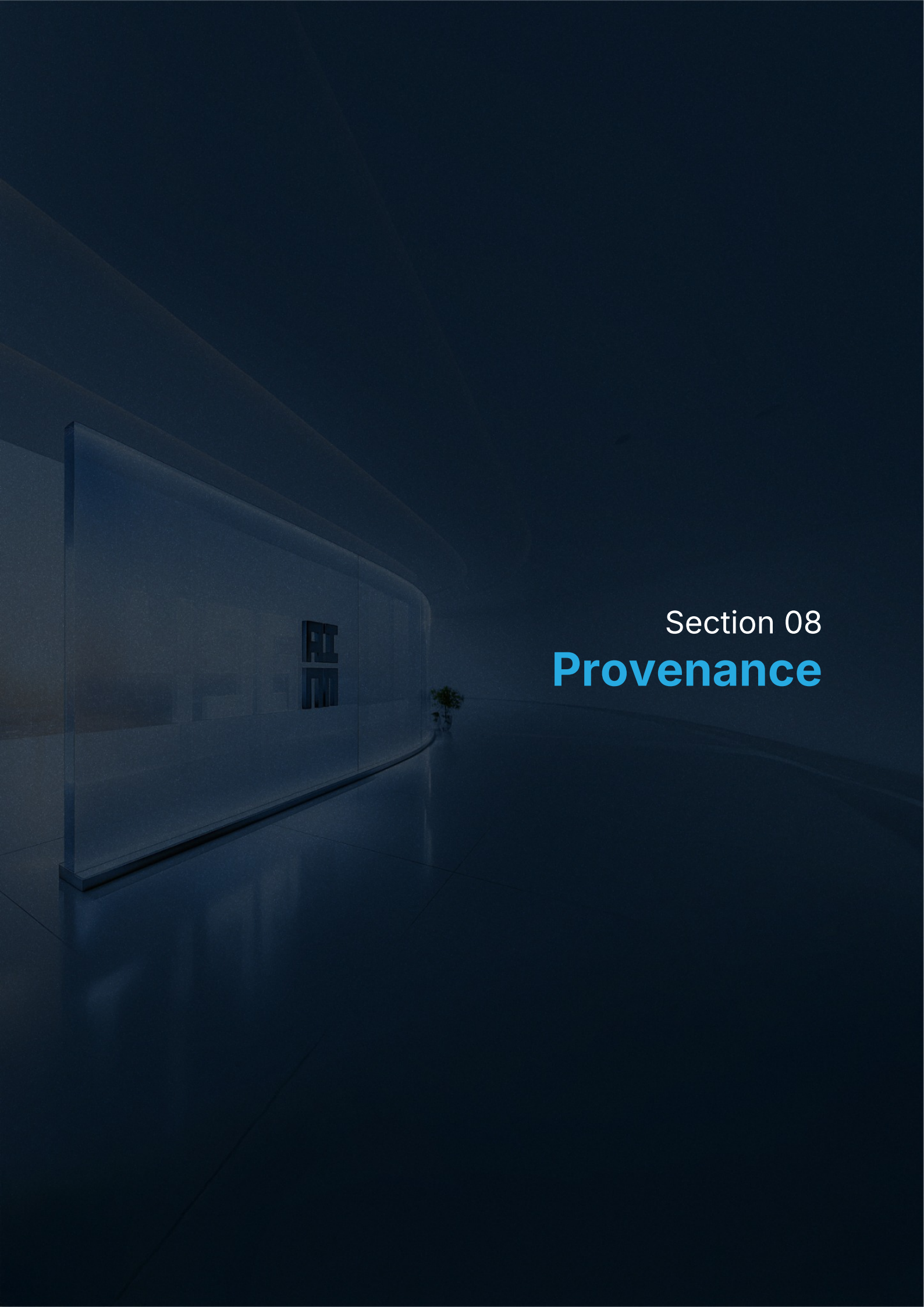
Section 07

Wake

Every AIOC has its own runtime, woken from its queue. With no human present, a woken Chief of Staff reads, records and triages only, unless the AIOC's Configuration slot unattended_action, set by the accountable human, allows more.

1. **Zero-cost wake (GOV, 14 October 2026).** A woken Cell first checks for work with no model call and holds a single-run lock; a model runs only when there is work. Every unattended Cell declares its cadence and is flagged when its newest output is older than twice it.
2. **Delegation ladder (GOV, 14 October 2026).** Beyond read-record-triage and act-within-limits, the unattended_action slot may hold a ladder: for each class of act, decides alone, decides and reports, only prepares, or never. The level covering an act is read from the act itself; every listed type is "never" (Standard §5.2).

Sources: DR-0010.



Section 08

Provenance

1. A Work Item links to its sources by ID, version and location.
2. The receiver reads the valid version at the controlling copy; a link that does not resolve to exactly one valid version fails the input check.
3. Output returns labelled with the sources and versions it was made from.
4. **Evidence class (GOV, 14 October 2026).** Every claim in an output carries one evidence class: proven, measured, cited, hypothesis or not established; a fact that could not be extracted is "not extracted", never guessed.
5. **Verdict record (GOV, 14 October 2026).** A verification is a record written by the verifier's own step, bound to the full content hash of the exact version read (spec/schemas/verdict-record.schema.json); a new version voids it; queues and tools act only on such records, never on the text of a note; a blocking verdict names a failing check the verifier wrote and returns the item to its producer; an advice-only verdict never counts as an approval; the verifier is never the producer.
6. **Completion ladder (GOV, 14 October 2026).** Drafted, adopted, executed and implemented are separate states, each reached only by its own recorded event; a register row is not evidence.
7. **Staleness and precedence (GOV, 14 October 2026).** A derived artefact (a knowledge pack, an installed instruction, a summary) is stale as soon as a source it depends on is newer, and is not relied on until regenerated. When sources disagree, the [ORG]AION's fixed precedence applies, and a conflict it cannot resolve goes to a human.
8. **Generated values (GOV, 14 October 2026).** Every figure, hash, count or identifier in a record is read from what produced it, never retyped.
9. **Request package (GOV, 14 October 2026).** A request between units may travel as an immutable, versioned package holding:
    - a machine-checkable header;
    - the requester's words verbatim, with the interpretation beside them;
    - numbered acceptance criteria;
    - numbered rules, each with its source;
    - a declared scope;

- a hash manifest of every attached file.

A change is a new version. Work tied to a superseded version pauses by itself, and a verifier fails a result built on it.

10. **Known-failure register (GOV, 14 October 2026).** A register of past failures, vendor behaviour and never-again decisions, each with its source, is attached by area to every new Work Item at planning. It holds what cannot yet be a test, and it is never the only holder of a relationship or of tacit knowledge (HF-13).



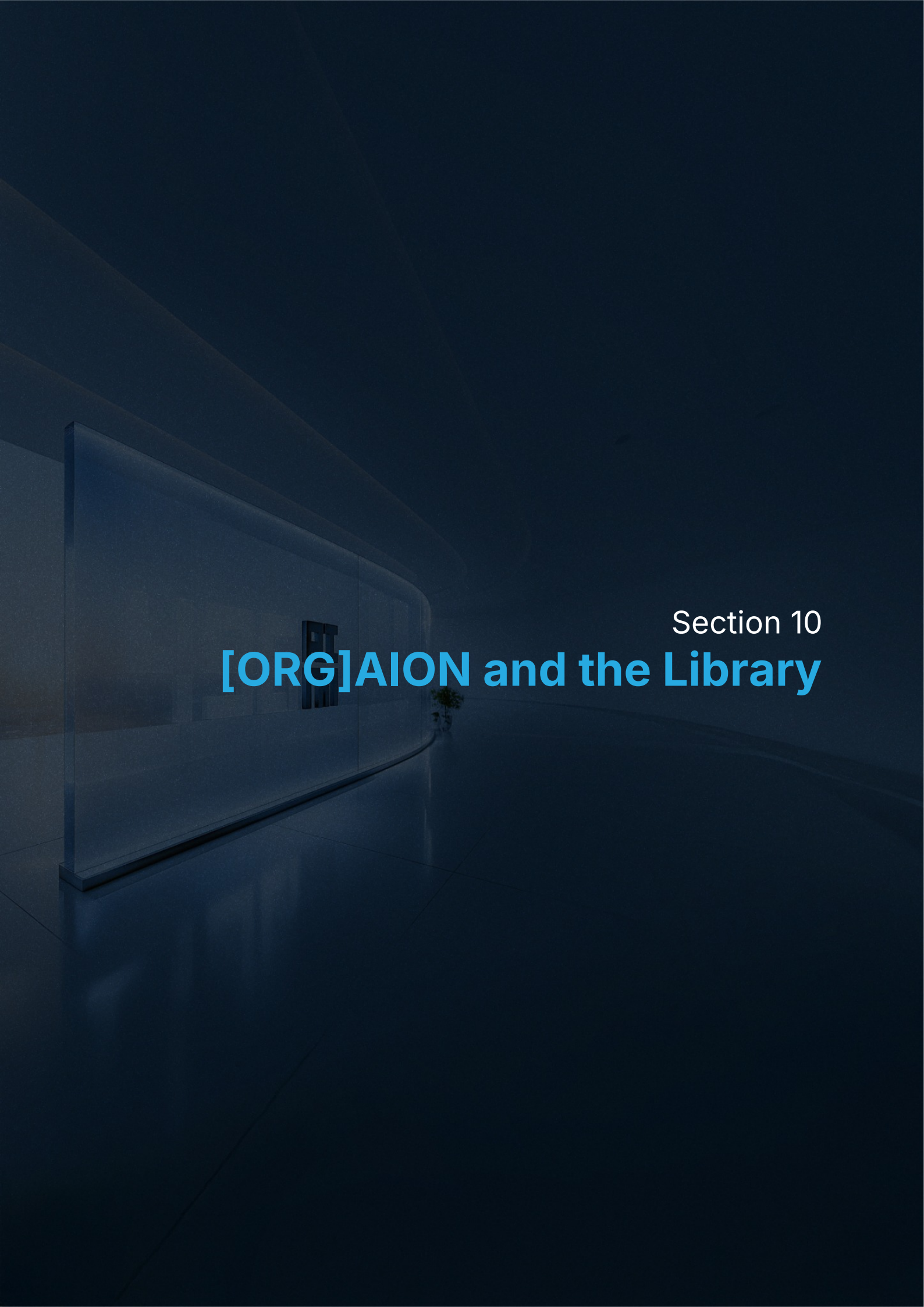
Section 09

Ports, bootstrap and stop

The definitions are in the Standard: ports (§6.1), bootstrap (§5.3), stop (§4.3). The machine form of the Port card is spec/schemas/port-card.schema.json. Every AIOC, Pod and Cell has a mandatory, queue-less stop port of type HU (the stop door), enforced outside it. Stopping an AIOC or a Pod stops everything under it, and no inner part can bypass the stop (§4.8); a human port never passes through an AI unit; a stopped unit restarts only by human decision. OCO sits under [ORG]HOAC-TM: TM and [ORG]HOAC-GOV may each stop it; a stop by TM comes directly through a human port, never through a Panel that OCO built; no AI can restart a stopped OCO (Standard §4.2, item 4; §4.3, item 2).

The bootstrap follows the path of the complete definition (Standard §5.3, item 6): Edition Maker hands over the core with the gateway and the stop; the governance seat accepts it; the core makes the first proposal; the seat hands the operation to Top Management; the Top Management Group builds and runs the operation.

Sources: DR-0041.



Section 10

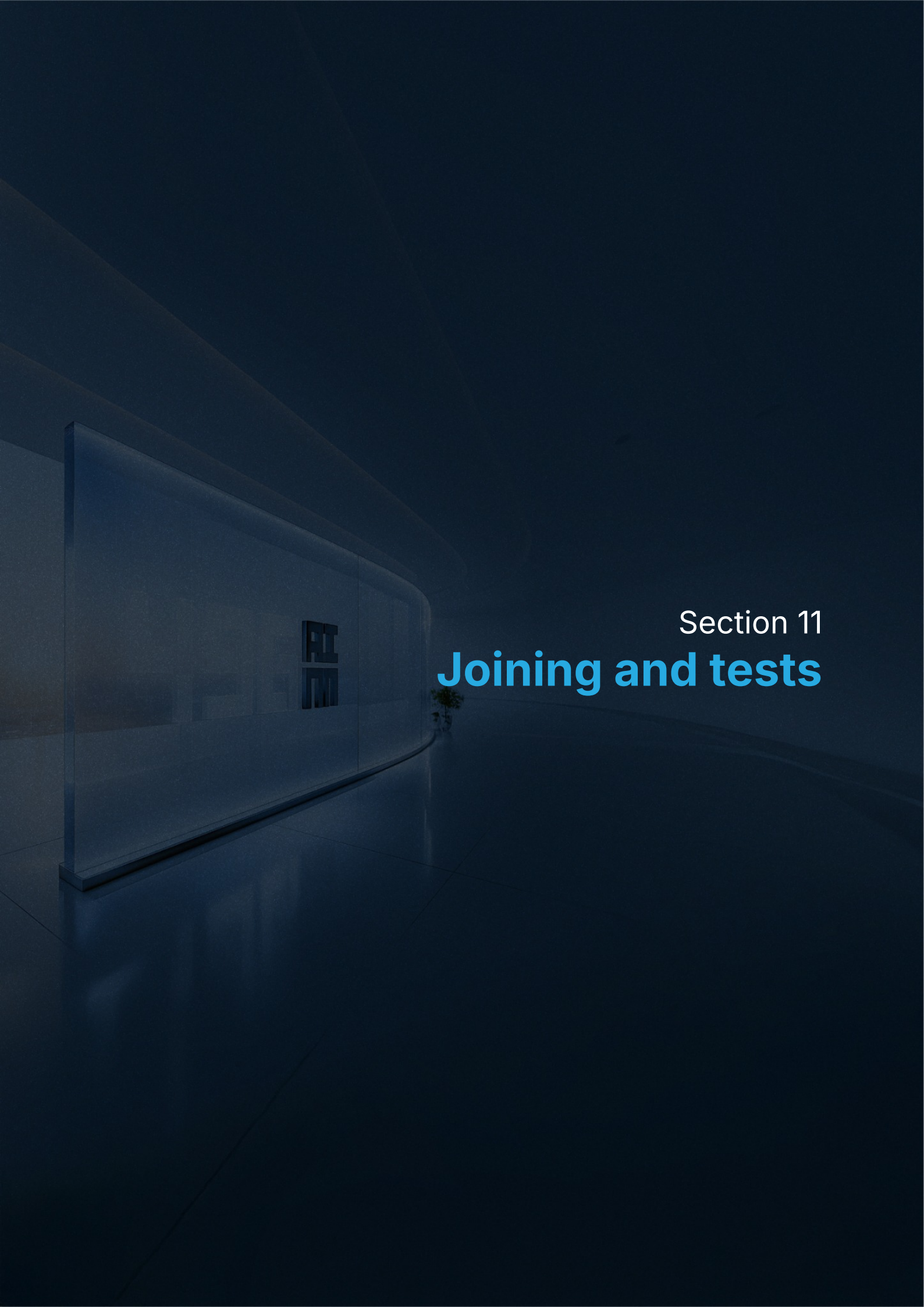
[ORG]AION and the Library

1. Each [ORG]AION records which Standard, Protocol and Library versions it is based on, the parts it took and its Library harvest (spec/schemas/org-aion-record.schema.json). At delivery its Standard is identical to the 49GAION Standard or stricter, never contradictory.
2. No [ORG]AION sees another [ORG]AION's data.
3. Each Library part has a card (inputs, outputs, test, version); tests travel with their parts (spec/schemas/part-card.schema.json). Every bot has an identity card (spec/schemas/bot-identity-card.schema.json). A card stays DRAFT until its test has run once with recorded evidence.
    - Every bot has its own credential, issued to it alone, held by the one Cell it fills, and never shared or reused (credential on its v3 identity card). A bot's identity card never claims a right to sign (never_signs: true): the right comes only from a grant, recorded at the gateway, on the position's side. Only a position's own Chief of Staff signs for it, a listed action included, and only within that position's grant, tagged as a delegated act (§4.10, items 6 and 7); every other bot, in an AI Cell or an attached AI Cell, never signs a listed action, and a signature record of a listed action names a bot only as the Chief of Staff that signed it under a grant. Stopping a part under the stop rule (§4.4) is not a signature. When the filler of a Cell changes, the bot's memory of the previous filler is sealed, never purged.
    - There is no standalone Self-Improvement unit or unit card. Self-Improvement is a process held inside [ORG]HOAC-TM (Standard §2.1, item 6): its fields are recorded only on the card of the Cell carrying [ORG]HOAC-TM, under processes, and it is served by AI Cells, read as "AI Cells serving the Self-Improvement process in TM" wherever a text says "the Self-Improvement AIOC". Its access to records is an access rule, not a signature type: read-only, pseudonymised, with no private store, and changed only as a Work Item (processes[].access).
    - In library/cards/ are the cards of the AI Cells serving the Self-Improvement process in [ORG]HOAC-TM; the card of 49GAION Source Lifecycle, source-lifecycle.part-card.json (§13); of 49GAION Process and Position Design, process-and-position-design.part-card.json (§17); of 49GAION Product Core, product-core.part-card.json (§18); of 49GAION Target Operating Model, target-operating-model.part-card.json (§19); of 49GAION

Delivery and Implementation, delivery-and-implementation.part-card.json (§20); of 49GAION Operations Event Register, operations-event-register.part-card.json (§22); and of 49GAION Charter, charter.part-card.json (§23; 49GAION/DR-0029). The Library's design return contract is in library/contracts/ (§17, item 11).

4. Giving back is the organisation's choice: an [ORG]AION may give back what it builds better, generalised and without its data, and 49GAION's Library and central repositories stay open to it (Standard §1.3, item 7). Once handed over, the [ORG]AION belongs entirely to the customer and is independent (Standard §1.3, item 4).
5. The Standard names roles only; tool names appear only in an [ORG]AION.
6. All AIOCs reach AI models through one shared gateway that holds keys, meters cost per AIOC and logs every call. It carries no orders. The shared gateway is the tool gateway of each [ORG]AION, a registered part carried by the Charter, the root AIOC (Standard §1.7, item 13), and installed with the core, but run as code beside it, outside the reach of every AI, the core's own included (§23; Standard §4.4, item 11); 49GAION runs no gateway for an [ORG]AION. HU ports and the stop door are served there through a sign-in that no AI built, hosts or can change (Standard §6.1, item 6).
7. **Runtime contract and engine binding (GOV, 14 October 2026).** One versioned runner performs the common duties of every AI Cell in a clean workspace per run; each Cell pins its runner version. A Cell's engine binding (tier, floor, model) is versioned on its card (engine_binding) and switches at the next run, never mid-task. Engines are re-scored on a schedule against a frozen set of past cases; a regression demotes an engine at once (Standard §5.1).
8. **Spend (GOV, 14 October 2026).** The gateway keeps decided spend, standing commitments and metered usage as separate numbers, billed and notional apart; applies the soft hold at a set fraction of a cap; pauses a Cell past its cap alone; at the global cap holds only new non-critical starts; and keeps the last caps when its meter fails. Each capability's degradation order is read from its blueprint.

Sources: DR-0007, DR-0008, DR-0009, DR-0014, DR-0016, DR-0017, DR-0018, DR-0027, DR-0029, DR-0035, DR-0036; GOV, 15 October 2026.



Section 11

Joining and tests

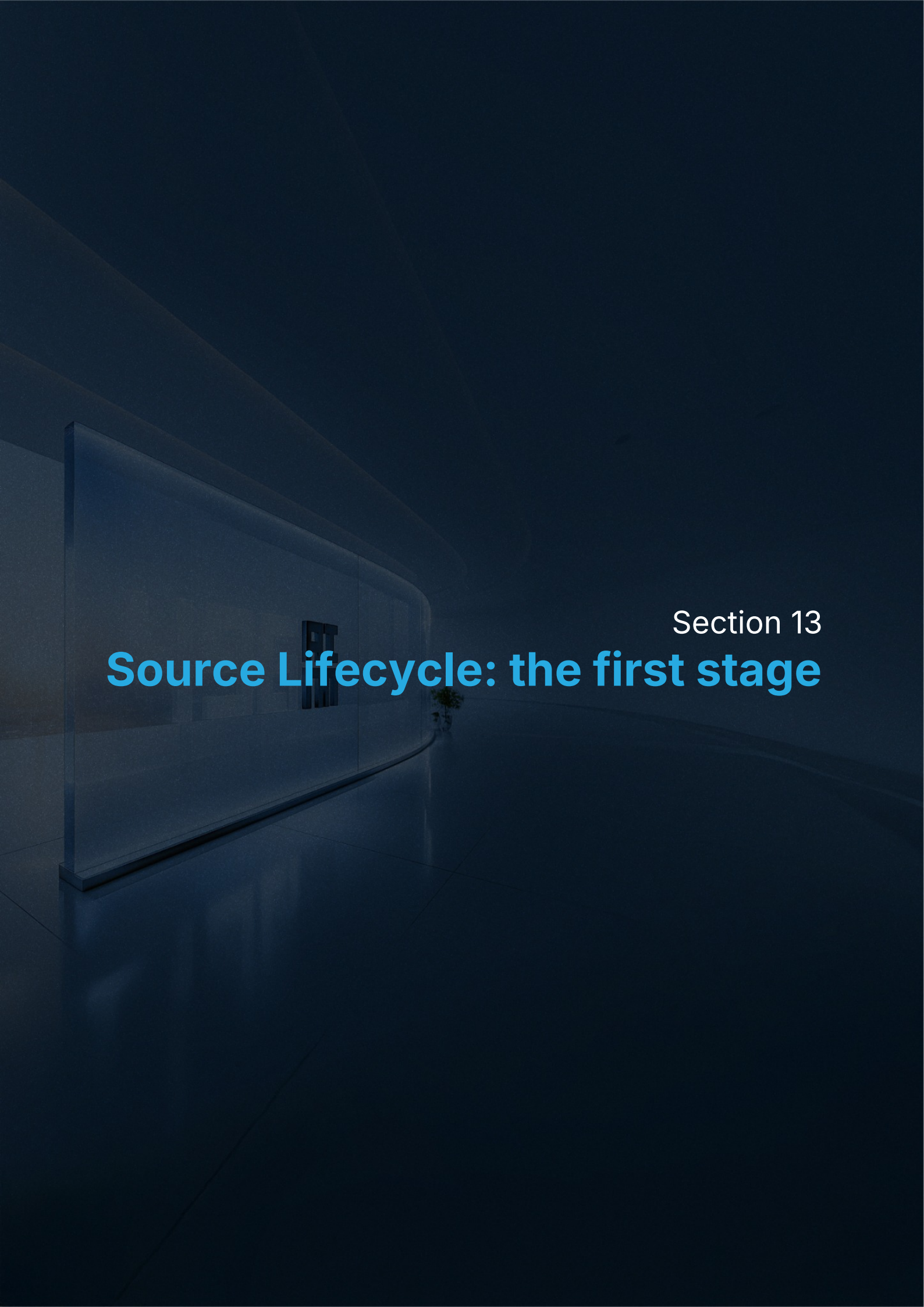
1. **Joining test.** An AIOC receives work only after every input and output on its card passes fixed samples; failing claims are removed from the card. An AIOC with an empty Configuration slot does not join.
2. **Cross-AIOC test.** A fixed set of chains across AIOCs is re-run after every change to a card, the protocol or a shared part.
3. **Outside tools** are re-tested against known samples; a tool that fails is disconnected.
4. **Verification by a non-author (GOV, 14 October 2026).** Work is verified by a Cell that did not produce it, on another model provider where one exists, in a fresh context, with the producer's description withheld; it restates the acceptance test and writes its own checks first; one non-author pass suffices; when a verifier is unavailable the next in a fixed list verifies; a fixed share of closed items is re-verified at random.
5. **Fresh-instance and duty-carriage tests (GOV, 14 October 2026).** On every change to a definition, a fresh instance is built from the definition alone and run against the fixed tests; every duty a role card gives a position is traced to a process step, an event or a setting, and a duty with no carrier fails the unit's fixed test.
6. **Deterministic executor (GOV, 14 October 2026).** A routine, consequential action that is not a listed type is carried out only by a program with no model, on machine-readable preconditions (verified at the exact version, all checks green on the combined result, not held, not stopped), with one kill switch. Only a deterministic re-run of a registered check opens an action; a model's words are evidence, never authority.
7. **Check placement (GOV, 14 October 2026).** Every automated check that is not a fixed test declares its placement, its budget per item and its degradation order, and is advisory unless it shows a real defect (Standard §3.2).



Section 12

Alignment

The card follows the A2A Agent Card; the Work Item follows the A2A task. The parts answer the failure categories of the MAST study (arXiv 2503.13657): cards and Work Items for specification failures, typed ports and one owner for misalignment between agents, the input check and visible evidence for failed verification.



Section 13

Source Lifecycle: the first stage

49GAION Source Lifecycle is a Library part (library/cards/source-lifecycle.part-card.json). In an [ORG]AION it is the AIOC [ORG]AIOC-SL (Naming Standard R2).

1. **The first part.** It is the first part an organisation receives after the Charter, with [ORG]HOAC-GOV, [ORG]HOAC-TM and [ORG]AIOC-OCO installed in it (Standard §1.7, item 13; §5.3). Like every other part, it enters through OCO. It is the stage through which every source enters.
   - It works for any organisation, from a single person to a group of entities.
   - Every later part builds on it.
2. **What it takes.** Documents and files of any format, and a contributor's own words, by text, chat or voice.
   - Each contributor accepts an agreement before their first input. An input made before that is refused.
   - Words become a record only after the contributor approves a preview of the exact text. With no approval, nothing is recorded. Human silence is never consent.
   - A secret in any input is refused and appears in no record or log (§5, item 3).
3. **What it does.**
   - It registers every input, gives it a reference and classifies it.
   - It fixes each input through its own stages, so that each source has one valid version (§8).
   - It tells the organisation what is missing for that kind of organisation.
   - The sources are the organisation's own data and never enter the 49GAION Library (Standard §1.3, item 9).
4. **Its sockets.**
   - **Input socket.** It checks everything that arrives (§5; STANDARD-CAPABILITIES §4.3).
   - **Output socket.** It answers a connected part by reference (ID, version and hash). The answer is "held" or "not held", and covers only the scope approved under item 5.

5. **Connections.**
   - **Approval and scope.** A part connects only by request.
     - The [ORG]HOAC-ACS seat that [ORG]AIOC-SL is accountable to approves the request and sets its scope, as the listed type connect-external-port (§4.9).
     - A wider scope is the listed type grant-or-widen-access.
     - No bot approves a request or sets a scope, except that seat's own Chief of Staff under its grant (§4.10, items 6 and 7; §10, item 3).
   - **Revocation.** That seat, or any seat above it, may revoke a connection at any time.
     - Revocation takes effect at the tool gateway first, and is never held or blocked (§4.8, item 6).
     - It invalidates the connection's token at the source service.
     - A read through a revoked connection is refused.
   - **Gateway.** Every source connection runs through the tool gateway, which runs as code beside the core, outside the reach of every AI (Standard §4.4, item 11). No AIOC calls a source service directly.
     - A stop takes effect at the gateway first, and is never blocked.
     - The expiry of a scope is checked against the date of each read.
   - **Token.** A connection's token is held only in the platform's secret store or behind the tool gateway. Each connection has its own token, with least privilege. A token is never written to any file, record or log.
   - **Register.** Every connection, scope change and revocation is recorded as an event in the unit register, which is append-only (Standard §4.4, items 2 and 7). Each event carries its date, its scope and the approving seat.
6. **Then the next stage starts.**
   - The next stage takes its sources through its own input socket, from Source Lifecycle's output socket. A document store the organisation already keeps may be connected the same way.

- What the next stage lacks, it asks for through Source Lifecycle, which records the answer as a source.
- What reaches the next stage is therefore documented, versioned and traceable to its source.

7. **Replaceable in later stages.** Source Lifecycle is a part like any other (§2). In the stages after the first, another source of documents or knowledge may replace it or stand beside it. Examples:
   - a document management system the organisation already uses;
   - a version-controlled repository of its texts;
   - a knowledge base;
   - a records or archive system the organisation already keeps.

Such a source qualifies only when it uses the same sockets, follows item 5, and passes the joining test (§11, item 1; Standard §3.2, item 4). It must:

- answer by reference (ID, version and hash);
- answer "held" or "not held";
- serve only the approved scope.

Tool names appear only in an [ORG]AION (§10, item 5).

8. **The fixed test.** Every rule of this section has a case in TEST-SOURCE-LIFECYCLE (library/cards/README.md). The card stays DRAFT, with last_result NOT_RUN, until the test has run once with recorded evidence (§10, item 3).
9. **Serving the design stages** (49GAION/DR-0019; the section The design chain, §21).
   - **Needs requests.** A connected part asks for what its sources lack as a needs request, a sub-item of its own run's Work Item (§4.6). Source Lifecycle checks it against the published sockets contract and returns a malformed request, one that names a person, and one that asks for a secret, each with its reason. It routes each row to the position that answers it, through that position's HU port, never to a named person. Every outcome comes back as a needs answer, by reference. A request past its deadline is reminded and escalated, never closed. One decision per row, at

most twelve rows per pass; silence, or approval of a whole pass, decides nothing; a row of a listed type is decided only with its signature at the tool gateway (§4.9).

- **The sockets contract.** The needs request, needs answer, needs status, Missing Report and source reference are versioned contracts with fixtures, published by Source Lifecycle. A change names every connected part it affects and re-runs the cross-AIOC test (§11, item 2).
- **The reference.** A source reference also carries the snapshot ID, the origin kind, the status, the confidentiality level, for an export the date it is as of, and for an item from conversation each statement's label with its anchor.
- **Pinned snapshots.** A connected part may pin a snapshot and read at it for the life of its run. A needs answer is read at the snapshot it names. A snapshot named by a filed release stays readable while that release, or one that carries it forward, stands.
- **Filing by a design stage.** Each design stage files only its own origin kind and design report: the part that designs the target operating model, operating model; Process and Position Design, process and position design; Delivery and Implementation, delivery and implementation. A filing is made on behalf of a named HOAC position that holds a current contributor agreement, takes its status from the decision it cites, and reads back byte-identical. A design report that holds a return keeps the hash of the return as sent; a later version of the return supersedes it without changing it.
- **Notices.** A revocation, narrowing, reclassification or supersession that affects references a part has read sends that part a notice naming each affected reference and version, never the content.
- **The fixed test.** TEST-SOURCE-LIFECYCLE v2 adds a case for every bullet of this item; the card moves to v2 and stays DRAFT, with last_result NOT_RUN.

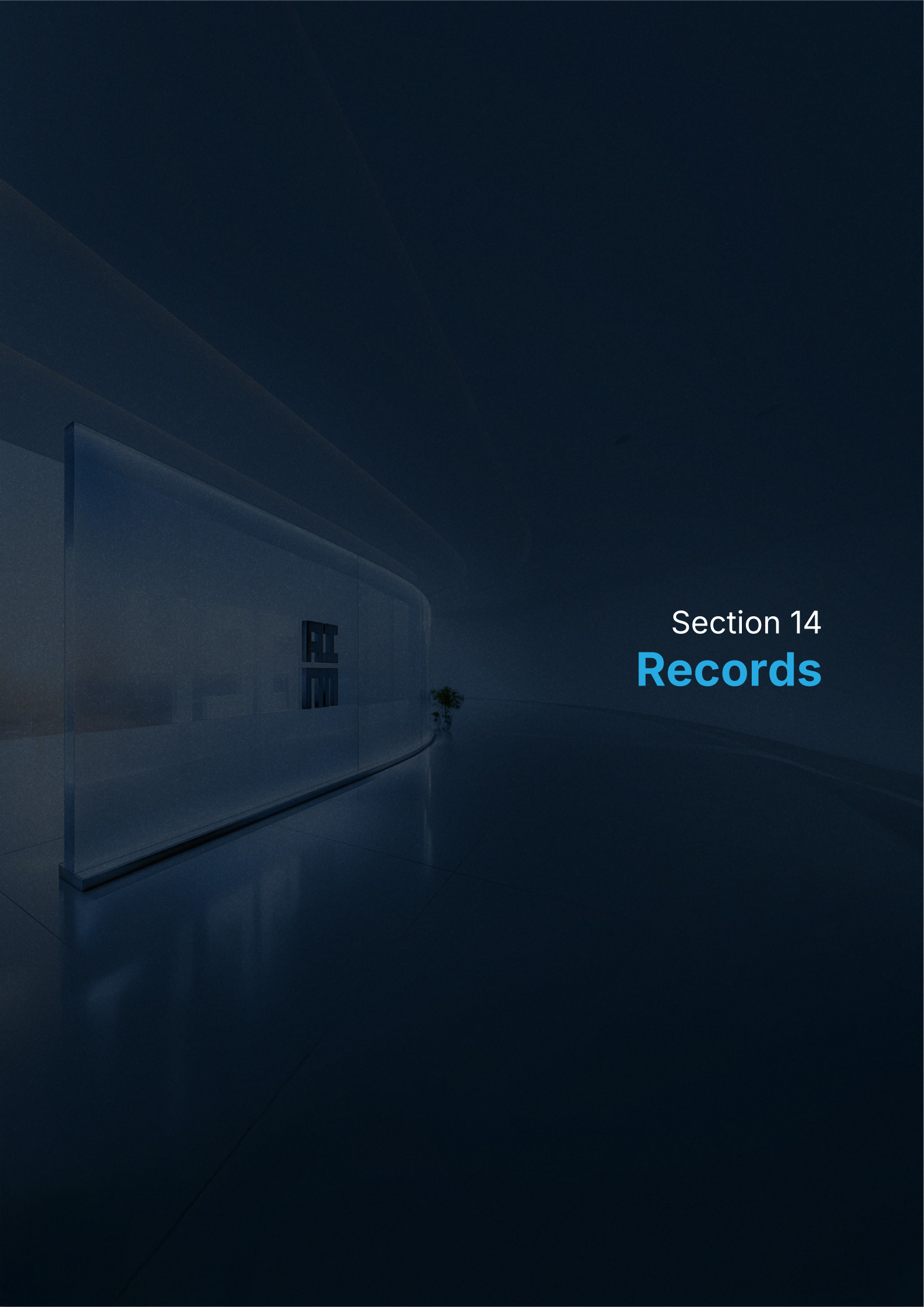
10. **Beside or in place of Source Lifecycle** (item 7). A source that stands beside Source Lifecycle is read only, through the reading part's own adapter. Asking and filing stay with Source Lifecycle. If a part ever replaces Source Lifecycle, this section names the new asking route before the replacement joins.

11. **What it also keeps** (49GAION/DR-0023).
    - **The organisation dictionary.** One source of the organisation's own terms, each with its definition and the source it comes from, or an approved statement of the governance seat. A definition is never invented. Two sources that define a term differently are both kept, with a contradiction record, until the governance seat decides (49GAION/DR-0049). It is served by reference to every design stage.
    - **The essentials checklist.** Every profile of item 3 opens with the essentials an organisation's sources should hold: the strategy on one page, the plan, the measures, the structure of positions, the processes, the meetings, and the dictionary. A missing one is a gap and a flag only; Source Lifecycle never drafts it. The essentials still missing are shown beside a Design Request when it is put for approval.
    - **Records of the work as it runs.** An export of records of the work is recorded only as the organisation's own projection by role-type and process, which removes who did the work, after an impact assessment and the decision of the position that holds data protection (HF-17; where personal data moves, the listed type export-share-or-purge-personal-data). An export that still says who did the work is refused, and nothing of it is kept. It is served as an export with its date.
    - **The sockets contract.** A decision row on an element of a design carries its proposed priority, and the answer its decided priority; an export request for records of the work states its aggregation by role-type and process.
    - **The fixed test.** TEST-SOURCE-LIFECYCLE v3 adds a case for every bullet of this item; the card moves to v3 and stays DRAFT, with last_result NOT_RUN.
12. **What it takes from the Operations Event Register** (49GAION/DR-0027; §22).
    - **Correspondence.** A new origin kind, correspondence, class EVIDENCE: a formal item the organisation sent or received, kept because a position asked to keep its register entry. Only the Operations Event Register files it, on behalf of that position, which holds a current contributor agreement. Its

hash equals the entry's content hash. It is kept as received, never edited, and never served as the organisation's own statement.

- **The register's scope.** It writes correspondence, and export for its projection of the work by role-type and process, under every rule of item 11. It reads references only (ID, version, status, held or not held) of the sources its entries cite, never content, under a grant signed as item 5 says; the notices of item 9 reach it for every version an entry cites. It is not a design stage.
- **Gap signals.** An act the register finds done without a source reaches Gap Finding as a gap signal, by reference to the register entry. It becomes a gap like any other, never a source.
- **The fixed test.** TEST-SOURCE-LIFECYCLE v4 adds a case for every bullet of this item; the card moves to v4 and stays DRAFT, with last_result NOT_RUN.

Sources: DR-0008, DR-0019, DR-0023, DR-0027, DR-0049; GOV, 14 October 2026.

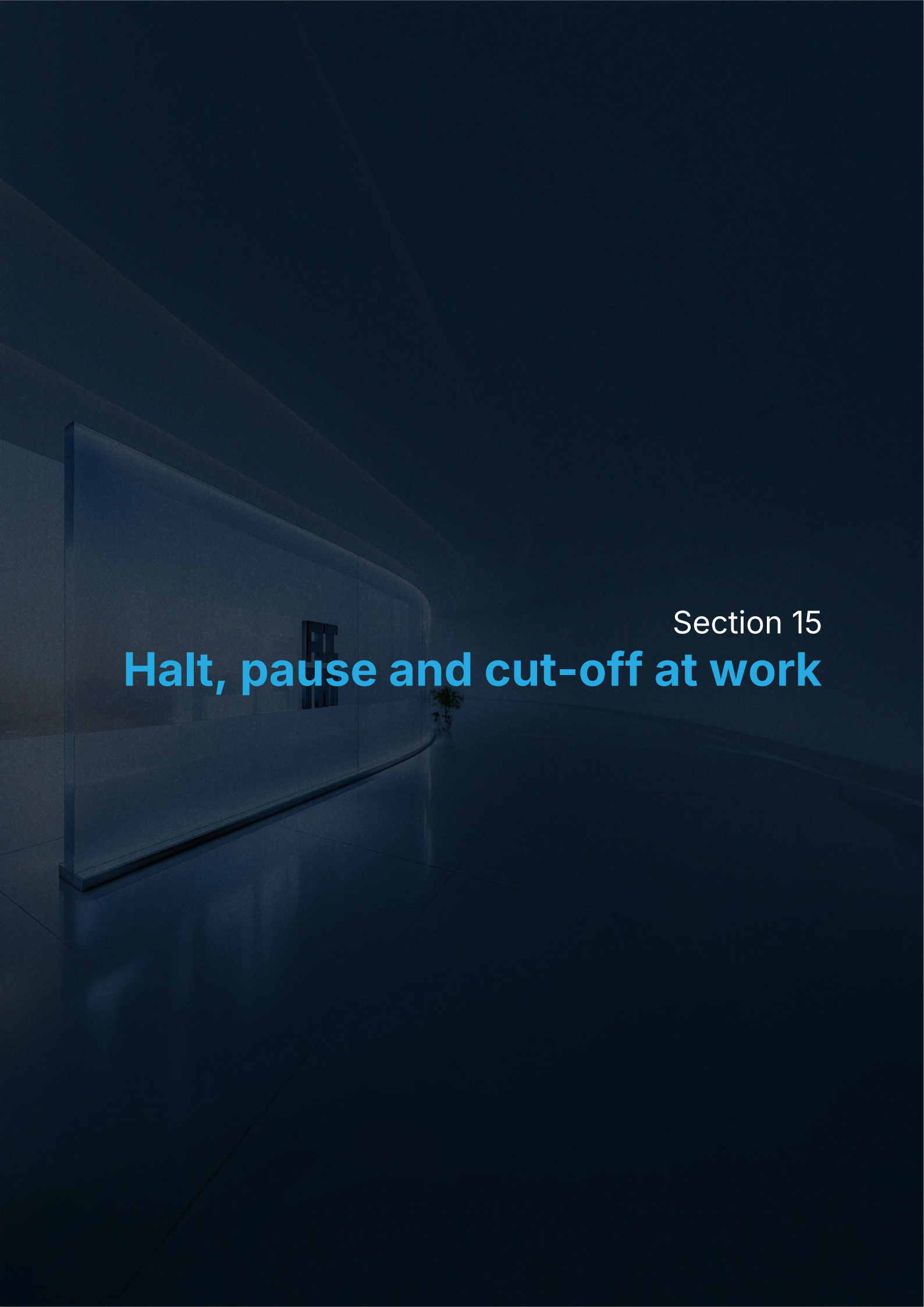


Section 14

Records

1. **One home.** An AIOC's records live in one home as immutable events. Each event carries a time-sortable ID, which is also its idempotency key, the actor's Cell code and the expected revision. Two writes on the same revision are both kept: the first applies and the later becomes a question to its author.
2. **Portals and projections.** Every register, board and number is a projection rebuilt from the events, carrying the version of the logic that built it. Every surface is a portal and keeps no copy. A write shows first as "not yet joined" and, if it fails the rules at join, as "refused at join" with the rule's ID. A change made in a projection is a request, checked before it becomes an event.
3. **Registry.** One machine-readable registry declares every record type, field, state, event, emitter and condition; a write it does not allow is refused, and the refusal names every condition that fails.
4. **Numbering.** One numbering service issues every readable number, a stream per type, atomically and safe to retry; no writer mints its own.
5. **Daily check.** A daily job proves that projections equal a rebuild from the events, that no number stream has gaps or duplicates, that every act is attributed to a part, and that every reported number traces to its events. A failure opens a defect and stays flagged.
6. **Restricted partition.** Records that name a person or carry the particulars of a corporate act live in a separate store read by named positions; every other record refers to them by ID only (Standard §4.4).
7. **State snapshot and rules in force.** On every change the unit publishes one human-readable and one machine-readable snapshot from its registers, keeping one line in its accountable human's own words, and stops publication if a personal name or contact would appear. For every position, a generator builds the rules in force today with a fixed precedence and marks superseded and lapsed items.
8. **Mechanism register and Reserve.** The unit's register of mechanisms and its Reserve of ideas not adopted are records like any other (Standard §2.1, §3.2).

Sources: DR-0010 (GOV, 14 October 2026).



Section 15

Halt, pause and cut-off at work

1. **The control record.** One control record per scope holds the state (running, draining, paused, resuming), the grace time and the reason. Every runtime reads it before every start and every step. On draining, new work stops at once, and running work stops at its first safe point within the grace time and writes a structured hand-off (status, item, version, what is done, the next steps in order, open questions, where the evidence is). After the grace time, work stops with a forced hand-off. Resume follows the declared order from the hand-offs, and the next run reads its hand-off first. A halt never delays a stop (§9).
2. **Flags and holds.** A flag or hold placed by a human seat is an event on the part or item. Every effect already queued for held work is held at the system that would run it. Work is released only when no pause, stop or superseded version still holds it. A hold placed by a human is lifted only by a human seat. Lifting a hold on a part or its work is restart-stopped-unit; lifting a containment hold that holds only a resource, a flow or an outside effect is restore-or-rollback (item 5; Standard §4.3, item 9).
3. **Cut-off.** A stop of AI authority raises a cut-off epoch that every executor checks before every mutation, refusing within seconds any credential issued before it. Delegated AI authority holds only while its witnesses renew a short lease from a fresh heartbeat; the [ORG]AION sets how many witnesses there are, on at least one host other than the AI's own (Standard §4.3, item 11). Human seats' own actions are never fenced. Re-arming, and any return of authority on a restored heartbeat, is signed.
4. **Recovery path.** The inverse of an executed action runs through a recovery-only path that needs no model, broker or Panel and grants no forward authority. It journals itself when the central log is down, and writes "reverted" only after the restoration is verified. Running it is restore-or-rollback, unless the action was taken under a standing action class whose envelope names the automatic revert.
5. **Soft containment.** On a deterministic breach, a containment part (never the detector) applies at once the narrowest reversible hold for the highest good in play; a human seat confirms or lifts it. A containment hold that stops a part or its work is lifted as in item 2 with restart-stopped-unit; one that holds only a resource, a flow or an outside effect is lifted with restore-or-rollback (Standard §4.4, item 19).

Sources: DR-0010 (GOV, 14 October 2026).



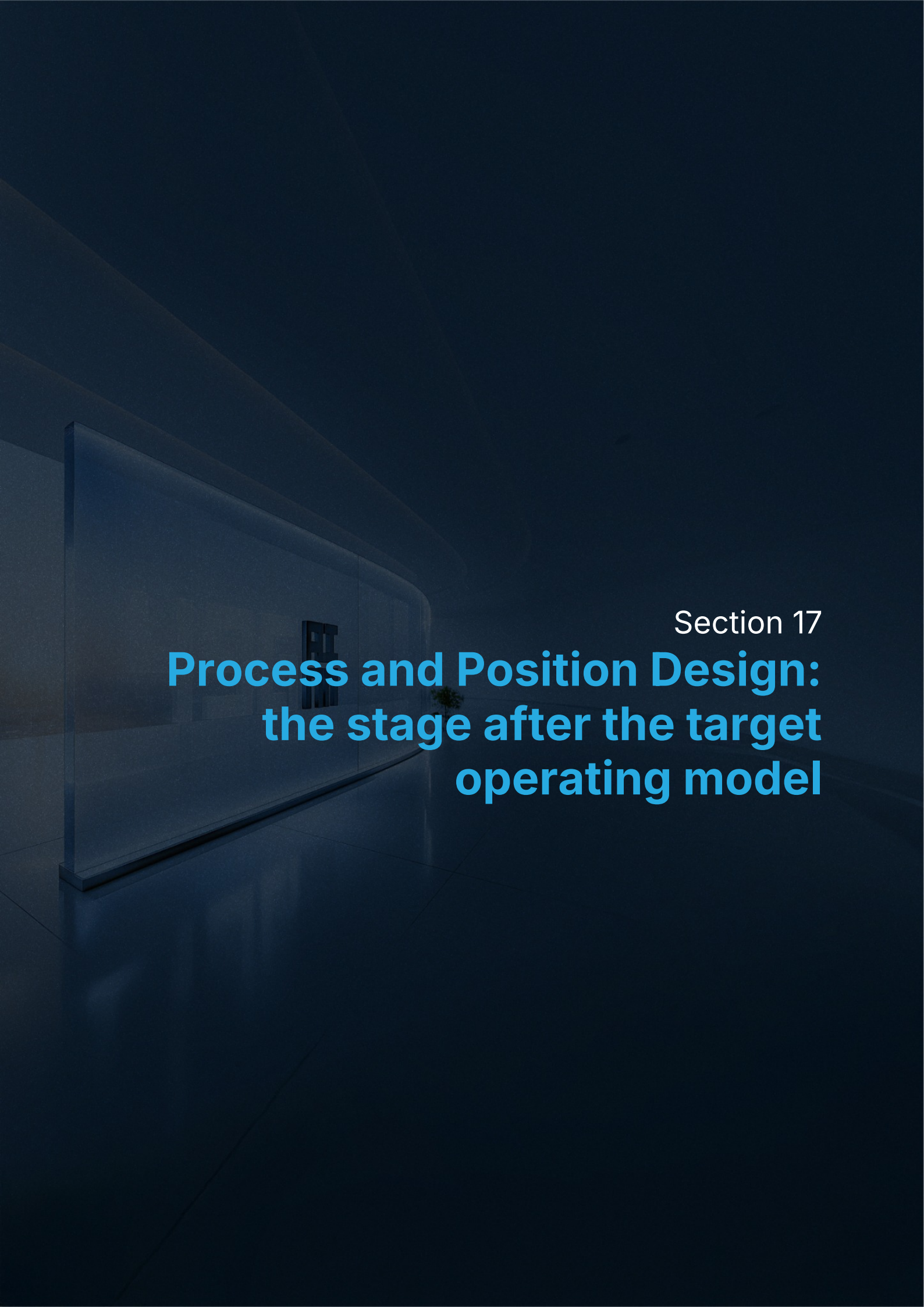
Section 16

Library parts added on 14 October 2026

Four Library parts, each with its card in library/cards/ (DRAFT, last_result NOT_RUN) and its fixed test in library/cards/README.md:

1. **49GAION Duty Deputy:** answers members' recurring asks for a human seat only from signed, unexpired rulings, cites the ruling, and raises every ask no ruling covers and every conflict between rulings (Standard §5.2).
2. **49GAION Outside Inspector:** checks from infrastructure that shares no failure with the inspected AIOC that its duties happened; builds nothing, directs no one, writes only its own report; never the internal-audit line (Standard §3.2).
3. **49GAION Advisory Council:** asks several model providers independently, without the asker's instruction or a leading frame, keeps every reply whole and merges by a stated rule; decides nothing (Standard §2.4).
4. **49GAION Reference Comparison:** compares an organisation's output with recognised references of the same kind; perception by a model, judgement by rules and cited evidence, decision by a human; facts counted, never taste; public sources only, and no account created in another party's system.

Sources: DR-0010 (GOV, 14 October 2026).



Section 17

Process and Position Design: the stage after the target operating model

49GAION Process and Position Design is a Library part (library/cards/process-and-position-design.part-card.json). In an [ORG]AION it is the AIOC [ORG]AIOC-PPD (Naming Standard R2). It is a part of its own, never merged with the part that designs the target operating model or with the stage after it (Standard §3.1, item 3; 49GAION/DR-0014).

1. **Its place.** It follows the part that designs the organisation's target operating model. Like every other part, it enters through OCO.
2. **What it takes.**
   - **The release, directly.** A decided target operating model release with its role register, handed over by the part that made it: by reference (ID, version and hash of its manifest), through its Input socket and the tool gateway. It reads the role register first. A release that is not decided, or whose hash does not match, is not used.
   - **Other sources.** The organisation's other sources (its existing processes, procedures, policies, volumes and times) come only by reference from Source Lifecycle's output socket (§13, item 4).
   - **The request.** A run starts only from a request of the Requesting Authority: the Cell carrying [ORG]HOAC-GOV, or a human Cell to which it has delegated the design decisions (Standard §4.2, items 5 and 8).
3. **Back to the target operating model.**
   - It never edits the release. A defect it finds goes back to the part that made the release as a return, one of: accepted; cannot use, missing what; contradiction found; assumption invalid; request kind unsupported. Each return cites the elements it concerns.
   - The elements a return concerns are held until a new decided release, or a decision of the Requesting Authority, answers it. Work on the other elements goes on.
   - The two parts may go back and forth as often as needed. Each pass is a new version; nothing is overwritten.
   - Each return is also filed into Source Lifecycle as a design report, with the same hash, so that a revised request to the part that made the release can cite it.

4. **What it asks.** What the release and the sources do not answer, it asks of the Chief of Staff of Source Lifecycle as a needs request, and the answer is recorded there as a source (§13, item 6). It never asks a person, and a request never names a person.
5. **What it does.** One stage, processes first; each step has one output and a done-test.
    1. **Process architecture.** The processes under each value-stream phase and capability of the release, in levels, each with one owner position. Every phase and capability has a process, and every hand-off of the release belongs to one.
    2. **Critical processes first.** The processes that cross parts of the model are chosen, with the reason recorded, and designed first.
    3. **Process design.** For each process: its purpose and trigger; its ordered steps in role lanes; the inputs and outputs of each step; the decision at each step and who decides it; exceptions and escalation; measures, volumes and step times. Each step has one accountable role, each decision one decider, and each hand-off between lanes is matched at both ends.
    4. **Positions.** The steps and decisions of each role are grouped into positions. Every step and decision sits in exactly one position, and every position holds at least one. Each job description is generated from what the position holds: its purpose, accountabilities, decisions and their limits, the positions it works with, its measures and its reporting line. A job description is never written free-hand.
    5. **Structure.** Reporting lines inside each part, with spans and layers checked; standing forums, each with a charter. Every position has one reporting line.
    6. **Workload.** The work each position carries, from the volumes and step times of its processes.
    7. **Walkthrough.** The design is walked through realistic situations. Every finding is fixed or put to decision.
    8. **Decision and release.**
6. **What it gives.**

- **The release.** A Process and Position Design release with these parts: process architecture; process book; decision-rights register and authority matrix; policy register; position register with job descriptions; organisation charts; forums with charters; workload; crosswalk from the current organisation to the target; walkthrough report and quality checks; decision package.
- **Traced, never invented.** Every element traces to an element of the release or to a source. A measure, volume or time that no source gives is marked as an assumption and put to decision.
- **Decided.** Every element is decided by the Requesting Authority. Silence, or approval of a whole pass, decides nothing.
- **Filed.** The release is frozen and content-addressed, filed into Source Lifecycle as a source, and handed to the next stage by reference.

7. **Positions only.** It designs positions, never people (Standard §1.1, item 9). No output names a person or reads who occupies a position. It makes no choice of what fills a position; that belongs to a later stage. For an [ORG]AION, that later stage is the AI-native design (Standard §1.1, item 2): it starts from the output and from what AI can do, not from these positions one by one.
8. **Its connections.** Two connections, each approved and scoped as in §13, item 5: to the part that made the target operating model release (read its releases; send returns), and to Source Lifecycle (read; needs requests; write only its own filed releases and returns). Revocation cuts a connection at the gateway at once and holds the run.
9. **Assurance.** The output of each step is checked by Cells that did not produce it before the next step starts (Standard §3.2, item 14). Critical outputs are written twice, separately, and their differences are settled on the sources or put to the Requesting Authority.
10. **The fixed test.** Every rule of this section has a case in TEST-PROCESS-AND-POSITION-DESIGN v1 (library/cards/README.md). The card stays DRAFT, with last_result NOT_RUN, until the test has run once with recorded evidence (§10, item 3).
11. **What the part also gives, and how its returns end** (the mother blueprint of the part, version 0.3.0).

- **The facts of each step.** Each step also says when it is done, its worst error and whether that error can be undone, the kinds of act it is (from a fixed list), where its input comes from and how far it is trusted, and the confidentiality of what it handles. Each step names its risks, its controls and the record it leaves; a step whose worst error cannot be undone, or that signs, pays or commits money, or communicates outside, carries at least one control, or the gap is put to decision.
- **Controls and segregation of duties.** A controls catalogue is compiled from the process records, and every position is checked against the segregation-of-duties rules; each breach is put to decision. Two positions are two parties (Standard §1.1, item 10).
- **Units.** The positions are grouped into units (teams, sections, departments, in the organisation's own words). Every position belongs to exactly one unit; every unit has one head position from among its own positions, one parent unit, and a charter generated from what its positions hold. Reporting lines follow the units unless the Requesting Authority decides otherwise.
- **Measures and rhythm.** Each process has measures of quality, time and cost; an objectives cascade links the decided goals to the measures of value streams, processes and positions; every forum has a tier, every measure a forum that reviews it, and the review calendar is generated from them.
- **Structure, load and time.** Structure scenarios side by side when the organisation asks for them, with cost only from cost rates the organisation gives; the Requesting Authority chooses. The peak-load check, and the end-to-end time of each process and value stream.
- **Competence and re-approval.** Each job description carries a competence profile traced to the steps and decisions of its position. Each decided process carries its owner position and its re-approval date.
- **Methods, not evidence.** A reference taxonomy, flow or measure library may be used as a method, never as evidence.
- **Returns end.** Every return, to the part before it or from the stage after it, follows the Library's design return contract, library/contracts/design-return.schema.json (49GAION/DR-0018): it counts its rounds and ends answered by release, answered by decision, set aside

by decision and carried forward on a stated basis (a known defect or a decided assumption), or withdrawn. None is left open.

- **The fixed test.** TEST-PROCESS-AND-POSITION-DESIGN v2 adds a case for every bullet of this item; the card moves to v2 and stays DRAFT, with last_result NOT_RUN.

12. **The stage after it** (the mother blueprint of the part, version 0.4.0; the section The design chain, §21).
    - **Told directly.** On filing a decided release, it sends a release notice to Delivery and Implementation (§20) under the Library's release notice contract, naming the target operating model release it rests on.
    - **Returns from the stage after it** come directly, under the design return contract, and end as §19, item 6 says, with this part answering for its own release only. A defect of the target operating model release is returned by that stage to the part that made it.
    - **Set aside, carried forward, shown.** An element that goes forward on a set-aside return is marked in the release with its basis (a known defect or a decided assumption) and its decision, and is listed in the decision package.
    - **The fixed test.** TEST-PROCESS-AND-POSITION-DESIGN v3 adds a case for every bullet of this item; the card moves to v3 and stays DRAFT, with last_result NOT_RUN.
13. **What it reads from the target operating model** (the mother blueprint of the part, version 0.5.0; §19, item 12).
    - **Read, never filled.** It accounts for the strategy frame, the journeys, the value case and the activity profile, and for each capability's delivery model and maturity, each stream's criticality and each element's priority. What the release leaves open (not given, not decided, not set, unknown, no observation) stays open.
    - **Objectives from the strategy.** The objectives cascade starts from the one-year goals and critical numbers of the decided strategy frame.
    - **Critical streams first, by priority.** The processes on a critical value stream join the critical processes of item 5; the order of work follows the decided priority of the elements each process carries, and each process and position carries that priority into the release.

- **As the capability is delivered.** In-house: its processes. Shared service: in the part that provides it, with a hand-off. Outsourced or by a partner: a hand-off to the outside party and the steps that oversee it, never an in-house process for the delivered work. Not decided: designed only as far as the delivery model is not needed, the rest returned.
- **Journeys and maturity.** Each outside party's journey is checked against the processes that carry its crossings, in order; a capability whose decided target maturity is measured or improving has a measure on each of its processes.
- **Today's work.** The activity profile is shown beside the designed workload, per process, and may be cited for a volume or a time. It is never about a person and changes no position.
- **The fixed test.** TEST-PROCESS-AND-POSITION-DESIGN v4 adds a case for every bullet of this item; the card moves to v4 and stays DRAFT, with last_result NOT_RUN.

Sources: DR-0014, DR-0018, DR-0019, DR-0023, DR-0031.



Section 18

Product Core: the organisation's one product division

49GAION Product Core is a Library part (library/cards/product-core.part-card.json). In an [ORG]AION it is the AIOC [ORG]AIOC-PC (Naming Standard R2). This section states its structure only; its blueprint is written in its own repository.

1. **Its place.** It is the organisation's one product division, shared by every company the organisation runs. It is to products what the organisation's engineering AIOC is to engineering. Like every other part, it enters through OCO.
2. **What it owns.** The what and the why of every product, across the product's whole life: design, the partnership in build, and the run. The engineering AIOC owns the how and the when.
3. **The CPO Office.** The head office of the part. Every product owner reports to it. It holds:
   - the intake of product work;
   - the portfolio of products and their categories;
   - the shared standards of the part;
   - the weekly product review;
   - the line to the engineering AIOC.

Its accountable seat is a human seat, named by position (Standard §1.1, item 9).

4. **Product categories.** The organisation sorts its products into categories. Each category is one of four kinds:
   - **Product family.** Products grouped by the customer's principal destination, for example owning an asset, using it, or earning from it.
   - **Platform.** The shared capabilities every product uses, for example the account, onboarding and identity checks, wallets, payments, conversion, the deal engine and reference rates. A capability is never a product, even when it has a fee, a screen or a route.
   - **Company category.** The products of one company of the organisation that stand apart from the families, for example its own token, its own engines or its own partner programme.

- **AI Systems.** The AI assistants and agents, the agents that run inside engines, websites that run with agents, back offices, internal panels, and internal infrastructure tools that run the organisation and are not sold. Every business system sits under AI.

Which categories an organisation has, and what each holds, is the organisation's own record and never appears in the Library.

5. **One product, one team.** Products are split small. Each product has one team for every stage of its life, and the team stays with it:
   - **One owner position** whose only job is that product. An owner position holds no other product.
   - **AI members by function:**
     - product management and the PRD;
     - design (UI and UX);
     - analysis;
     - pricing and product economics;
     - operations and customer voice;
     - documentation.

No stage of a product's life is handed to another team.

6. **What each team does, in order.**
   1. **Research and opportunity.**
   2. **The PRD and its feature packages,** with acceptance criteria. No field is left blank; "none" is an answer.
   3. **Design.** Flows, a screen for every state, UI and UX, and copy in every language the product serves.
   4. **Partnership in build.**
      - It answers the engineering AIOC's questions.
      - It takes the first look.
      - It accepts the product criterion by criterion.

It never writes code or technical design.

5. **Launch.**
6. **Run.**
   - Pricing and every rate that moves over time.
   - Performance, and the product's profit and loss.
   - Customer communications and retention.
   - Customer voice and complaints.
   - Operations and upkeep.
7. **Explainers and supporting documents.**
   - The product explainer and its questions and answers.
   - Help articles and user guides.
   - Partner materials.
   - Internal training.
8. **Refresh or retire.**

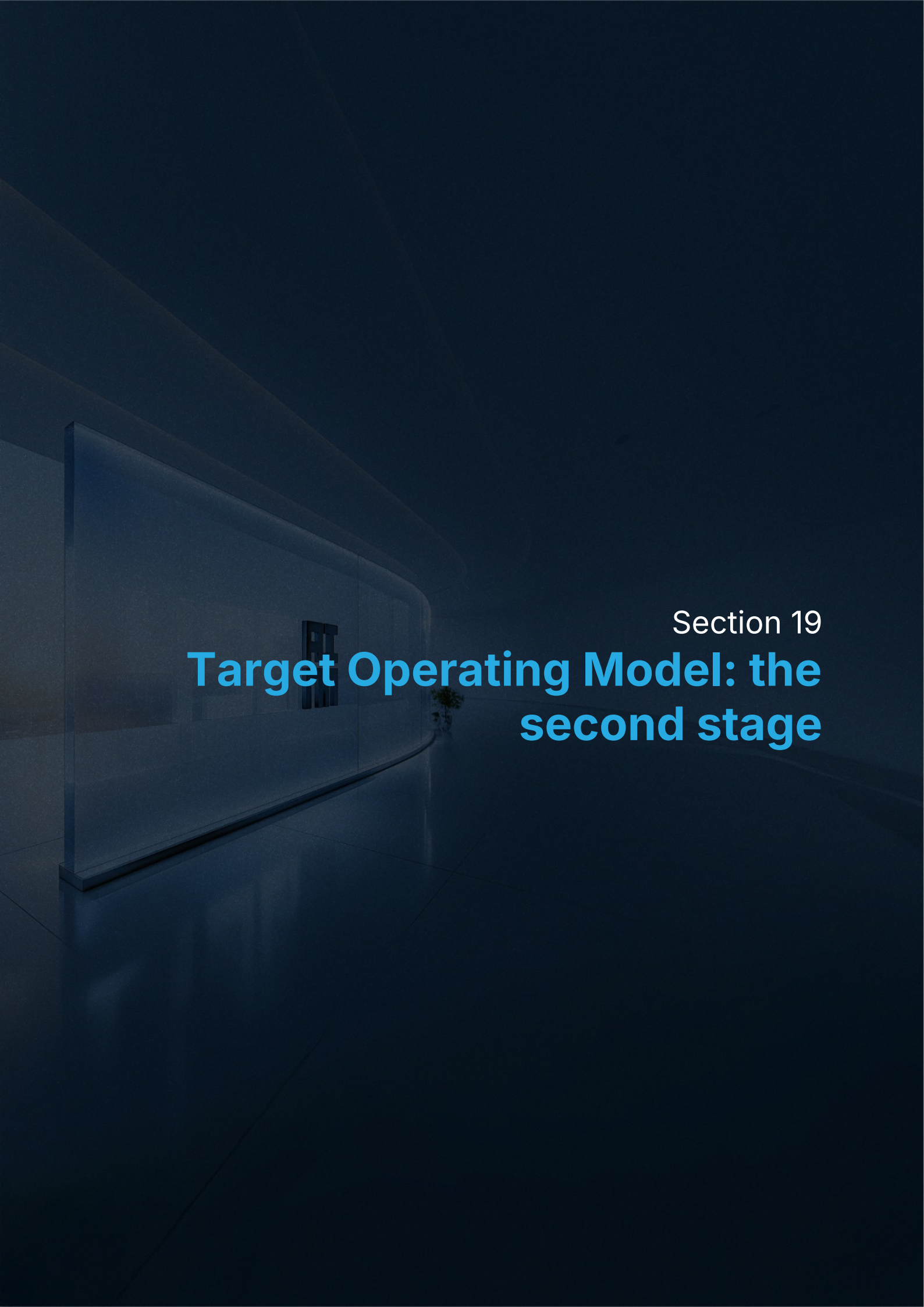
7. **Rates are data.** Every price, fee and rate of a product is kept as versioned data, with an effective date and a reason. The product's team changes it without a code release. Every change can be reversed. Whether a new rate reaches existing deals is set by each product's own terms.
8. **Shared standards, not shared work.** The CPO Office keeps these once, and every team uses them:
   - the design system;
   - the pricing models;
   - the measurement tools;
   - the document templates.

   They do no product's own work.
9. **One product, many companies.** A product is made once. It is configured for each company and market that offers it, never copied per company.
10. **The line to the engineering AIOC.**

- Product work reaches the engineering AIOC only as complete product packages, on the product line between the two parts.
- Nothing in Product Core speaks to builders or to the code host directly.
- A finding in build that would change the product comes back to the product's team as a question.

11. **Positions only.** It names positions, never people (Standard §1.1, item 9). No output names a person or reads who occupies a position.
12. **The fixed test.** Every rule of this section has a case in TEST-PRODUCT-CORE v1 (library/cards/README.md). The card stays DRAFT, with last_result NOT_RUN, until the test has run once with recorded evidence (§10, item 3).

Sources: GOV, 15 October 2026.



Section 19

Target Operating Model: the second stage

49GAION Target Operating Model is a Library part (library/cards/target-operating-model.part-card.json). In an [ORG]AION it is the AIOC [ORG]AIOC-TOM (Naming Standard R2) (Standard §3.1, item 4; 49GAION/DR-0016).

1. **The second part.** It is the stage after Source Lifecycle (§13, item 6). It designs the target operating model of an organisation of any shape: one owner's work, a single entity, a group of entities, or a group with independent institutions inside it. Like every other part, it enters through OCO.
2. **What it takes.**
   - Only sources served by reference from Source Lifecycle's output socket (§13, item 4). Its Input socket hands the engine one versioned Input Pack. Nothing typed or spoken to it, and nothing that arrives through a human port, is a source.
   - A run starts only from a Design Request: a source in Source Lifecycle approved by the Requesting Authority, the Cell carrying [ORG]HOAC-GOV or a human Cell to which it has delegated the design decisions. The Design Request states how fine the model is to be (light, standard or deep), within a cost limit the Cell carrying [ORG]HOAC-TM sets.
3. **What it asks.** What the sources do not answer, it asks of the Chief of Staff of Source Lifecycle as a needs request, a sub-item of the run's Work Item (§4.6). It never asks a person, and a request never names a person. One decision per row, at most twelve rows per pass, and with every pass a report of what is still missing. Silence, or approval of a whole pass, decides nothing. A row of a listed type is decided only with a valid signature at the tool gateway (§4.9).
4. **What it gives.**
   - A release: the organisation's capabilities, value streams, decision rights, interfaces, and the hand-offs between its parts and with outside parties; and a role register that names every role once, with what it decides, what it answers for and what it hands to whom. The next stage reads the role register first.
   - It names roles only. It does not say what fills a role (Standard §1.1, items 9 and 10).

- Every element traces to its sources, and the release is decided element by element by the Requesting Authority.
- The release is frozen and content-addressed, and filed into Source Lifecycle as a source.

5. **Linked directly to Process and Position Design.** On filing a decided release, it also sends a release notice to Process and Position Design (§17), by reference: the ID, version and hash of the release's manifest. It receives that part's returns directly, under the Library's design return contract (§17, item 11), and shows each on its panel with its age.
6. **The return loop ends.** A return is answered by a new decided release, made by a continuation run, or by a decision of the Requesting Authority. Each answer that does not end the return opens the next round, with its deadline. Past the deadline, or past a set number of rounds, the return goes to the Requesting Authority, who answers it, or sets it aside and lets the affected elements go forward on a stated basis: a known defect or a decided assumption. No return is left open, and the stage after it can always deliver.
7. **Its connections.** Three connections, each approved and scoped as in §13, item 5: to Source Lifecycle (read; needs requests; write only its own filed releases and design reports), to Process and Position Design (send release notices; receive returns), and to Delivery and Implementation (receive returns; send their answers). Revocation cuts a connection at the gateway at once and holds the run; no record keeps a source's text.
8. **Assurance.** Every stage's output is checked by several independent engines from more than one family, each first alone, then in a short debate held to quotes, settled by code without a vote; a human seat rules on what stays contested. The frame and each part's design are written twice, separately, by authors of two families, and their differences are settled on the sources or put to the organisation. The model is walked through sealed situations. No unit grades its own work (Standard §2.3, item 9). A material finding stops the release until the Requesting Authority decides it.
9. **Standing alone.** The release is a product on its own: an organisation may use it and stop there.
10. **The fixed test.** Every rule of this section has a case in TEST-TARGET-OPERATING-MODEL v1 (library/cards/README.md). The card

stays DRAFT, with last_result NOT_RUN, until the test has run once with recorded evidence (§10, item 3).

11. **Returns from the last stage, and how a return is handled** (49GAION/DR-0019; the mother blueprint of the part, version 0.5.0; the section The design chain, §21).
    - **From Delivery and Implementation too.** It also receives the returns of Delivery and Implementation on elements of its releases (§20, item 3), under the same contract and loop as those of Process and Position Design, on the third connection of item 7 (receive returns; send their answers).
    - **The notice.** Its release notice follows the Library's release notice contract.
    - **Admission.** A return is admitted only when it validates against the contract, names a release this part filed with the same manifest hash, and its filed copy reads back with the same hash; otherwise it goes back with its reason, never edited. An accepted return asks nothing and ends at once, answered by the release it names.
    - **Rows made by code; classified by a human seat.** The rows put to the Requesting Authority on a return are made by code from the return and the release, one decision per row. Each ended return is classified by a human seat of the part, never by its Chief of Staff, as a defect of the part or as normal discovery; a defect becomes a lesson and a test case.
    - **The fixed test.** TEST-TARGET-OPERATING-MODEL v2 adds a case for every bullet of this item; the card moves to v2 and stays DRAFT, with last_result NOT_RUN.
12. **What the model also states, and the work as it runs** (49GAION/DR-0023; the mother blueprint of the part, version 0.6.0). None of these adds a stage. The part never writes a goal, a target, a level or a figure of its own: each is stated by a source, decided by the Requesting Authority, or shown as not given, unknown or not set.
    - **The strategy frame.** Purpose, values, long-range target, three-year picture, one-year goals with their measures, and critical numbers, each cited or not given. What the sources lack is asked once, through Source Lifecycle. It is decided in one row; purpose and values stay the listed type

set-purpose-or-values where no source in force states them. A missing element never stops the run.

- **Delivery model and make-or-buy.** Each capability is in-house, a shared service, outsourced, by a partner, or not decided, never naming a provider; make or buy follows by code. A capability not delivered in-house names the crossings that carry it.
- **Critical value streams.** A stream is critical only for a cited reason from a closed list: revenue, customer, legal or regulatory, safety, continuity.
- **Maturity.** Each capability's level now, only from a source, and its target, only from a source or a decision, on one closed scale of five levels.
- **Outside parties' journeys.** For each counterparty, the moments where it meets the organisation, in order, each a crossing already declared. Nothing is designed on the party's side; a moment no crossing carries is a seam finding.
- **The value case.** Per outcome, its measures, baselines and targets, compiled by code from the decided parts and decisions, with no figure of the part's own; decided in one row.
- **Priority.** Each element carries a priority, must, should or could, proposed by a fixed rule and decided with the element.
- **Records of the work as it runs.** Exports by role-type and process only, admitted as §13, item 11 says; an export that still says who did the work is refused at the Input socket. From them, code compiles an activity profile; with none, the part says so and goes on. No part watches work.
- **The release.** These are four more parts of the release (the strategy frame, the journeys, the value case and the activity profile) and fields of its design records; the next stages read them by reference and fill nothing it leaves open.
- **The fixed test.** TEST-TARGET-OPERATING-MODEL v3 adds a case for every bullet of this item; the card moves to v3 and stays DRAFT, with last_result NOT_RUN.

Sources: GOV, 15 October 2026; GOV, 16 October 2026; DR-0016, DR-0019, DR-0023.



Section 20

Delivery and Implementation: from the decided design to a running organisation

49GAION Delivery and Implementation is a Library part (library/cards/delivery-and-implementation.part-card.json). In an [ORG]AION it is the AIOC [ORG]AIOC-DI (Naming Standard R2). It is a part of its own (Standard §3.1, item 5; 49GAION/DR-0017).

1. **Its place.** The last stage of the design chain: it takes the decided design into a running organisation, and stays to run and improve it with the organisation. Like every other part, it enters through OCO.
2. **Three ways in.**
   - **Mode A.** The decided releases of the target operating model (§19) and of Process and Position Design (§17), by reference: the ID, version and hash of each manifest. A release that is not decided, or whose hash does not match, is not used.
   - **Mode B.** A design made elsewhere, filed as a source in Source Lifecycle. It is mapped onto the structure of a Process and Position Design release, showing what mapped, what did not and why; the author's intent is never rewritten.
   - **Mode C.** No design. It gathers what it needs from sources in Source Lifecycle and builds a light design with the governance seat (49GAION/DR-0049): a decision list, process checklists, who does what, a weekly rhythm. Every element is provisional until decided, and a growing organisation is moved to mode A.
   - A run starts only from a request of the Requesting Authority.
3. **Back to the design stages.** A defect it finds in a release goes back to the stage that made it under the Library's design return contract (§17, item 11), and the loop ends as §19, item 6 says.
4. **What it does.**
   - **Diagnose.** A gap check in two tiers: what blocks launch (decision rights for the Human Floor items, a position for each critical process, authority, controls) and what is incomplete. Each gap with its consequence, its evidence and a proposed fix.
   - **Prepare.** The procedures under each process; drafts of policies, of the pay structure by job family and level, of contract and offer templates, of the authority instruments and of the obligations register; the control matrix; the

systems, data and suppliers each process needs; the transition plan, with a consultation step wherever law requires one before a transition notice; the go-live criteria.

- **Accompany.** The implementation plan by position; answers about how the organisation works, each citing its design element, version and quoted span, or saying it does not know; problem records ending in a decision or an action; learning paths per position; readiness against the go-live criteria, rehearsals, and the go or no-go package for the deciding position.
- **Operate.** The operating rhythm and its single logs of actions, risks and issues, dependencies and decisions; measures, dashboards and forum packs; version control with owner approval and re-approval dates; acknowledgement of each changed version by the positions it concerns; runnable checklists; quarterly priorities per position; the value register, with each value figure traced to its baseline and confirmed by the finance position.
- **Improve.** Findings at process level; proposals to change processes, positions, job descriptions and forums, each with its evidence, effect and risk; trials judged by a seat other than the proposer; an organisation health check; and the hand-over of its skills, so that the organisation can run from its own exported records without it.

5. **It decides nothing.** What it may do is set per class of action by the owner position, starting at level 1 for every class: level 0, read; level 1, propose (a draft enters a record, log or plan only when a position inserts it); level 2, apply within a standing approval, only where the change is reversible and has no legal, pay or employment effect, as a new version with rollback, reported the same day. No level allows a type of the Human Floor signature list (§4.9). The go or no-go is always the deciding position's.
6. **The stop.** Any accountable position can stop it, or any change it applied, at once, from a control outside its chat; the stop is never gated.
7. **Processes, not persons.** It works at the level of processes and units. It never ranks, scores or watches a position, infers no emotion, and shows no measure for a single position (Standard §1.1, items 9 and 10). Records of work enter only through a projection owned by top management that removes who

did the work, and only after the organisation records its impact assessment and informs workers or their representatives where law requires.

8. **Drafts with legal effect.** Each records its entity and jurisdiction, is marked as a draft and not legal advice, and goes to decision only after review by a qualified professional for that jurisdiction; where the jurisdiction is unknown, it writes an outline only.
9. **Its connections.** To Source Lifecycle (read; needs requests; file its own records), to Process and Position Design (receive releases; send returns), to the part that made the target operating model (send returns; receive their answers), and to the organisation's own tools, read only by default; each approved and scoped as in §13, item 5, and revoked at the gateway at once.
10. **Assurance and trace.** Every citation resolves to a design element, its version and a quoted span, and a verifier that did not write a draft checks it; figures are read from sources, never typed. Every artefact traces to its design element, deciding position and decision, and back. A full export with its trace is available at any time.
11. **Standing alone.** It runs without any other part, with its own gateway that fails closed; inside an [ORG]AION it uses the tool gateway and Source Lifecycle.
12. **The fixed test.** Every rule of this section has a case in TEST-DELIVERY-AND-IMPLEMENTATION v1 (library/cards/README.md). The card stays DRAFT, with last_result NOT_RUN, until the test has run once with recorded evidence (§10, item 3).
13. **Answers to its returns** (49GAION/DR-0019; the mother blueprint of the part, version 0.1.1; the section The design chain, §21).
    - **A connection to the part that made the target operating model.** With that part it sends returns and receives their answers, on the connection item 9 names. Every answer arrives on the connection of the stage that made the release, on the return itself.
    - **What an answer does.** An answer by release re-opens only the work that cites a changed element. A return set aside carries its touched elements forward on the stated basis, each marked and listed in the decision package.

- **The notice it receives.** A release notice of Process and Position Design is read under the Library's release notice contract and names the target operating model release it rests on.
- **The fixed test.** TEST-DELIVERY-AND-IMPLEMENTATION v2 adds a case for every bullet of this item; the card moves to v2 and stays DRAFT, with last_result NOT_RUN.

14. **The value case and the priority** (49GAION/DR-0023; the mother blueprint of the part, version 0.1.2; §19, item 12).
    - **Carried unchanged.** In mode A, the value case of the target operating model release and the priority of each element are recorded by reference, never changed; a baseline unknown or a target not set stays so. The value register of item 4 starts from them.
    - **The rollout follows the priority.** The waves of the transition plan follow the decided priority, then critical processes first; a departure states why and is decided with the plan.
    - **The fixed test.** TEST-DELIVERY-AND-IMPLEMENTATION v3 adds a case for every bullet of this item; the card moves to v3 and stays DRAFT, with last_result NOT_RUN.

Sources: GOV, 15 October 2026; GOV, 16 October 2026; DR-0017, DR-0019, DR-0023, DR-0049.



Section 21

The design chain: how the design stages connect

This section was decided by GOV on 16 October 2026 (49GAION/DR-0019), with the items of Standard §3.1 on Source Lifecycle, the release notice contract, the Library register and the design chain.

1. **The stages, in order.** Source Lifecycle (§13), then Target Operating Model (§19), then Process and Position Design (§17), then Delivery and Implementation (§20). Each is a Library part of its own and enters through OCO. Each stands alone: an organisation may use any stage and stop there, and no stage waits for a later one.
2. **One home for every release.** A stage's decided release is frozen, content-addressed and filed into Source Lifecycle as a source of the stage's own origin kind (§13, item 9). Every later stage reads it there, by reference (ID, version and hash). No stage keeps or edits the release of another.
3. **Told directly.** On every filing of a decided release, the stage sends a release notice to the next stage, under the Library's release notice contract (library/contracts/release-notice.schema.json): the release ID, version and manifest hash; the releases it rests on; the filed reference; the final-status decision; the release it supersedes; and the returns it answers. A notice carries no release text and no name of a person, and is sent only after the filing reads back. A stage's own notice contract may narrow a field; it never adds or removes one.
4. **Returns.** A later stage sends what it cannot use back to the stage that made the release, directly, under the Library's design return contract (library/contracts/design-return.schema.json; §17, item 11), and files a copy into Source Lifecycle with the same hash. Delivery and Implementation returns a defect of a target operating model release to the part that made it, and a defect of a Process and Position Design release to that part.
5. **Every return ends.** It goes in rounds, each with its deadline, and ends answered by a new decided release, answered by a decision of the Requesting Authority, set aside by that decision and carried forward as a known defect or a decided assumption, or withdrawn by its sender. Past a deadline or a set number of rounds, it goes to the Requesting Authority and is raised to the Cell carrying [ORG]HOAC-TM. Silence ends nothing. Only the elements a return touches wait; an element carried forward on a set-aside

return is marked in the receiving stage's release with its basis and its decision.

6. **The chain never loops.** A return is a Work Item from the later stage. The stage that made the release answers it on that Work Item and never opens a new item back (§4.6, item 2).
7. **Connections.** Each link is a connection approved and scoped as in §13, item 5, signed by the [ORG]HOAC-ACS position over the stage whose releases it serves. A revocation cuts it at the gateway at once. Without its links, a stage still runs alone.
8. **Assurance.** In every stage after Source Lifecycle, each step's output is checked before the next starts by Cells that did not produce it: three or more engines from two or more families, each alone first, then a short debate held to quotes, settled by code without a vote. A human seat of the stage rules only on what stays contested. A finding with a verified quote is never removed by a count, a majority of families, the Chief of Staff or code. Critical outputs are written twice, by authors of two families, and their differences are settled on the sources or put to the Requesting Authority. No unit grades its own work (Standard §2.3, item 9).
9. **Positions only.** No stage names a person or reads who occupies a position. Two positions are two parties (Standard §1.1, item 10).
10. **The Library register.** Each stage is named in the Library register (library/README.md), with its card, its fixed test and its mother blueprint, pinned by repository, path, version, commit and SHA-256 (Standard §3.1, item 9), as the register writes it. An organisation's instance of a stage keeps its own values in its records repository.
11. **The fixed test.** TEST-DESIGN-CHAIN v2 (library/README.md) has one case for every item of this section, the cases of item 12 included, run across the fixtures of the four stages as a cross-AIOC test (§11, item 2).
12. **What passes along the chain.** Decided by GOV on 16 October 2026 (49GAION/DR-0023):
    - **The organisation's own words.** Every stage reads the organisation dictionary of Source Lifecycle (§13, item 11), where there is one, and uses a term in the organisation's meaning.

- **Open stays open.** What one stage leaves open (not given, not decided, not set, unknown, no observation), no later stage fills; it asks, through Source Lifecycle, or returns.
- **Priority and value travel unchanged.** The priority decided with each element of the target operating model, and its value case, reach the later stages by reference and unchanged.
- **Never about a person.** Records of the work enter the chain only as §13, item 11 says, and no stage turns them into anything about a person.
- **The fixed test.** TEST-DESIGN-CHAIN v2 adds a case for every bullet of this item.

Sources: DR-0019, DR-0023.



Section 22

Operations Event Register: the record of the live operation

49GAION Operations Event Register is a Library part (library/cards/operations-event-register.part-card.json), decided by GOV on 17 October 2026 (49GAION/DR-0027). In an [ORG]AION it is the AIOC [ORG]AIOC-OER (Naming Standard R2). Its mother blueprint is in its own mother repository (library/README.md). This section states its rules; the blueprint states its units.

1. **Its place.** A top-level AIOC under the Charter, beside Source Lifecycle. Like every part, it enters through OCO. It runs alone: with no Source Lifecycle installed, entries are made and the feeds of item 11 wait.
2. **What it records.** One entry for each crossing of a formal act, on every surface the organisation connects. Each crossing falls in one class:
   - **A. Formal correspondence:** letters and formal e-mail of an entity or position, in and out, and the organisation's own posts. A full, numbered entry with its due rule.
   - **B. Requests between units or entities:** these are Work Items. The Work Item's ID is the entry's link, and the register is a projection of the queue's events (§2, §4). A formal instruction between entities is also class A, with one entry on each side.
   - **C. Machine traffic:** AI calls through the tool gateway. The gateway log stays its home; the register keeps one root hash per log segment, and the hand-offs.
   - **D. Code-host events:** a merge, a close with a decision, a release, a tag. Never each comment or commit of a human position, and never an author; an AI Cell's commits are its acts (item 7).
   - **E. Noise:** newsletters, notifications, auto-replies, read receipts, sign-in mail. Counted per channel, never kept.
   - **F. Private:** personal and direct messages, private channels, recordings of calls or meetings. Never captured.
3. **The register entry.**
   - **Identity and integrity.** Its readable number comes from the numbering service (§14, item 4), in a stream per entity and direction. Its event ID is also its idempotency key. Its hash covers its fields and the hash of the entry before it. It records its capture mode (gated, pushed, fetched or

reconciled), when the act occurred and when it was registered, and late with its lag.

- **The crossing.** Its parties as unit or position codes, or the ID of an outside party in the restricted partition (§14, item 6).
- **Content, by reference.** A reference to its home, and the SHA-256 of its content. For a mutable object, a snapshot hash with its as_of time.
- **Follow-up.** Its follow-up state, as events.
- **Links.** Its Work Item, decision, signature, release and sources.
- **Governance.** Its confidentiality, legal hold and retention class. authority: none, unless a signature or decision is linked.
- **Corrections.** A correction is a new event.

4. **Record, don't carry.** The content stays in its home, and the register holds it by reference, as a Work Item holds its sources (§2, item 8; §8). Every field of a payload that identifies a natural person is removed at the Input socket before any write. A party is the Cell code the gateway stamps on an act, or the unit or position that owns the address (a functional mailbox, a formal channel, a Work Item's port); otherwise it reads not attributed. The register never receives, holds or infers an identifier of a natural person. An event that cannot be cleared is refused, and nothing of it is kept.
5. **The gateway gates; the register records.** "Register first" for a formal outgoing item is a policy of the tool gateway, under change-gateway-policy: the gateway takes a number before it releases the item. The register never authorises, sends, or refuses an item on its merits, and never holds an item back: an entry incomplete in form is completed later and flagged late. One AI never both authorises and records the same act (HF-18).
6. **Never blocking; never on the accountability path.** If the register is down, work goes on. The item is registered later under a number from a leased block, flagged late with its lag. A stop, revocation, objection, critique, skip report, internal-audit report or signature prompt never waits on the register; each is registered afterwards from the gateway log (§4.8). The whistleblowing channel is never connected, and is not even counted.
7. **Acts, not activity** (49GAION/DR-0027, item 4).

- **AI Cells.** Every act of an AI Cell is recorded, as metadata and references. A Cell that acts for a position (a Chief of Staff, an attached assistant, a duty deputy) mirrors a human position: of it, only the acts that leave the position's sphere are recorded (Work Items sent, signed actions carried out, formal outgoing items), never the exchange between the position and the Cell.
- **Human positions.** Of a human position, only its formal acts are recorded. The register never sweeps a mailbox or a chat. It keeps no read or response times, no presence and no recordings. It keeps no count, rate or history of a human position's acts. A human position is shown only the live state of its own open items.
- **Queries about one position.** A query by a human position's code over time is refused. The refusal is recorded to the position that holds data protection (HF-17) and to the internal-audit line (HF-19).
- **A fixed list.** The projections of items 8, 10 and 12 and the measures of item 10 are the whole list. A new projection or measure, or a new class or purpose on a surface, runs only after a new impact assessment, a decision of the Cell carrying [ORG]HOAC-TM, and the consultation of item 14. The internal-audit line reviews the register's use every year.

8. **Inbox and outbox.** Every unit's inbox and outbox are Panel views of the entries at its ports. Where the register is installed, a port that carries formal acts in scope names it in its audit_trail. Only those formal acts are registered; a human position's other passages through its HU ports are not. An address is an entity, a unit or a position, never a natural person. From a personal mailbox, only what the position itself files is recorded.
9. **Follow-up.** A due date comes from the process design or from the rule the organisation set for the surface; with none, the entry reads no rule. The owning position is told before and at the due date. The watchdog chases (§4.6, item 5); the register gives no order. Every closure carries a reason, and nothing closes by silence.
10. **What each part had and what it lacked.** For each entity, unit, process and AI Cell, the register computes eight measures against absolute bars set by the organisation:
    - coverage against each surface's census;

- entries linked to a Work Item;
- merges with no Work Item or verdict record;
- entries answered on time;
- done Work Items whose sources resolve;
- entries registered late;
- closures with no reason;
- refusals at the Input socket.

A figure drawn from the acts of fewer human positions than the minimum group size, whether for a unit, a process or a role-type, rolls up to the parent unit. Nothing is ranked. Every figure carries its logic version and traces to its events (§14, items 2 and 5). None goes to appraisal, pay or discipline.

11. **Feeds to Source Lifecycle** (§13, item 12).
    - **Correspondence.** An entry a position asks to keep is filed as correspondence, with an equal hash.
    - **The activity export.** The register's projection of the work by role-type and process is filed as an export with its as_of date, only after the decision of the position that holds data protection (§13, item 11). It is the producer of the activity profile of the design stages (§19, item 12).
    - **Gap signals.** An act done without a source sends a gap signal, by reference.
    - **Supersession.** On a notice of supersession from Source Lifecycle, the acts that cited the superseded version after it was superseded are marked.
12. **Evidence, handover and cases.**
    - **Evidence packs.** An evidence pack verifies each entry by re-hashing its home, where readable; a mismatch is a finding, never a repair.
    - **Handover lists.** A position's handover list, built on request of that position or of the HOAC above it, holds its open entries and commitments. The register reads nothing about who holds a position (49GAION/DR-0049).

- **Case packs.** A case pack holds pseudonymised past cases of a process, built only under a Work Item of the Cell carrying [ORG]HOAC-TM. Cases for shadow runs go to Self-Improvement, once its access includes the register. Cases for the golden set go to the human seat that chooses it and to the assurance unit, never to the loop (Standard §2.3, items 4, 7 and 9; §2.4, item 4).

13. **Integrity.** Every day the register proves that it is whole:
    - no number stream has a gap or a duplicate, and the chain verifies;
    - each surface's census matches, or the difference is listed;
    - every projection equals a rebuild;
    - every gateway log segment is anchored.

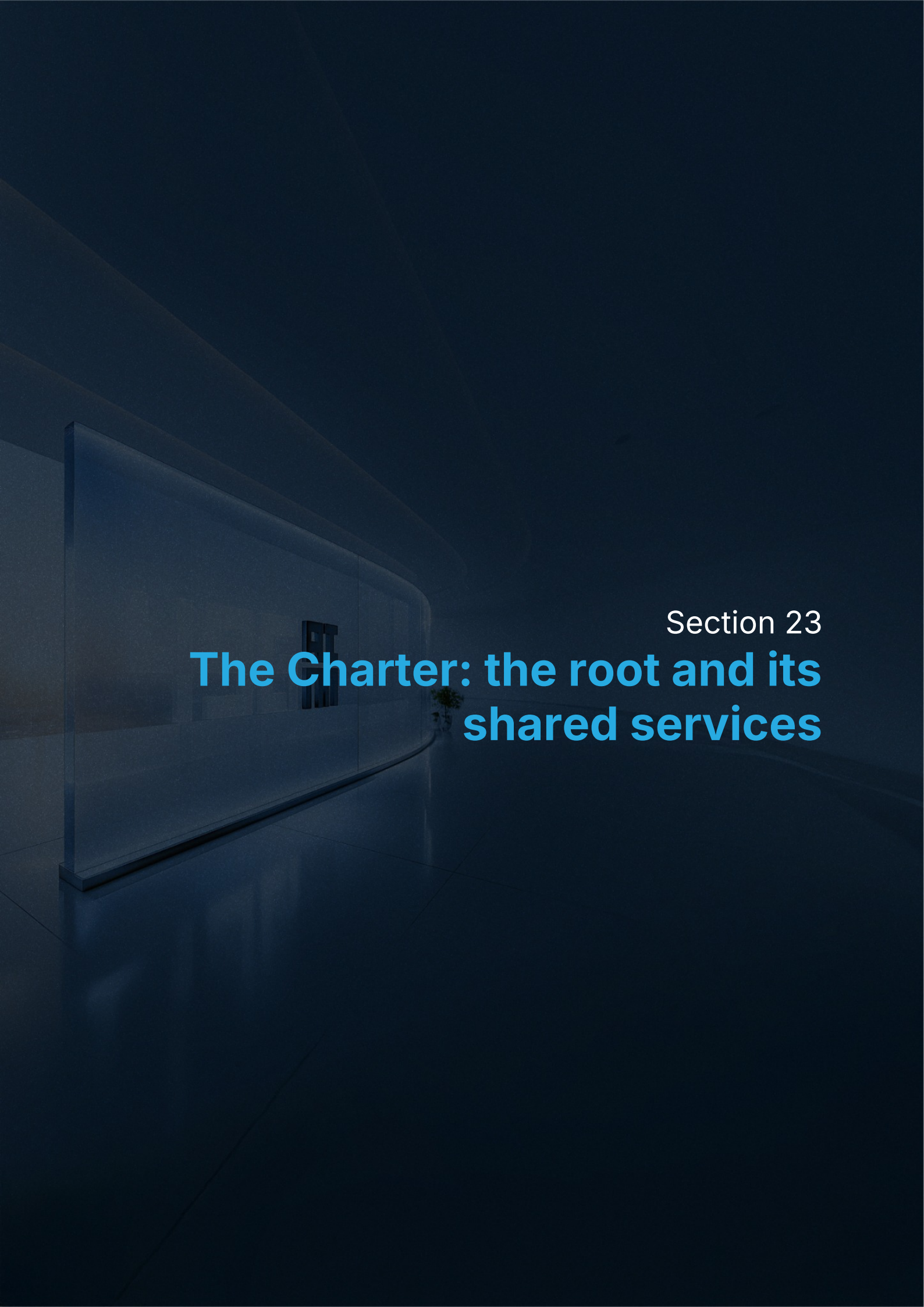
A failure flags and never blocks (§14, item 5). No part of the register can change or delete a past entry. The internal-audit line reads it directly, read only. Self-Improvement reads it once the Cell carrying [ORG]HOAC-TM adds it to the process's access, never its entries of the accountability path (Standard §2.1, items 7 and 8). A surface adapter that falls silent is a finding.

14. **Before a surface is connected.**
    - **The order.** In this order, each step recorded first:
        1. the scope, set by the Cell carrying [ORG]HOAC-TM: the surface, the entities, the classes of act set as formal, and the stated purpose;
        2. the impact assessment and a lawful basis per entity, decided by the position that holds data protection (HF-17), never on staff consent;
        3. the steps of labour law in each entity, and the notices;
        4. between entities, the intra-group arrangement;
        5. the port, connect-external-port, with its own token held only in the platform's secret store or behind the gateway (§13, item 5);
        6. an observed-only pilot whose negative tests pass.
    - **Between entities.** A crossing has one entry in each entity, linked by its crossing ID and content hash. Nothing is read across without a signed grant.

- **Retention and disposal.** Retention follows the schedule per entity and class (HF-01). Disposal is dispose-records-or-data, and a legal hold blocks it. An erasure destroys the key and leaves a tombstone.
- **Counsel.** Correspondence with counsel is kept by reference only, Strictly confidential, with no model call.

15. **Roles, not tools.** The part names surfaces by role; tool names appear only in an [ORG]AION (§10, item 5).
16. **Positions only.** No output names a person or reads who occupies a position (Standard §1.1, item 9).
17. **The fixed test.** Every rule of this section has a case in TEST-OPERATIONS-EVENT-REGISTER v1 (library/cards/README.md). The card stays DRAFT, with last_result NOT_RUN, until the test has run once with recorded evidence (§10, item 3).

Sources: DR-0027, DR-0049.



Section 23

The Charter: the root and its shared services

49GAION Charter is a Library part (library/cards/charter.part-card.json), decided by GOV on 19 October 2026 (49GAION/DR-0029). In an [ORG]AION it is the AIOC [ORG]AIOC-CH, the root of the organisation and the core of the [ORG]AION (Standard §1.7, item 13). Its mother blueprint is in its own mother repository (library/README.md). This section states its rules; the blueprint states its units.

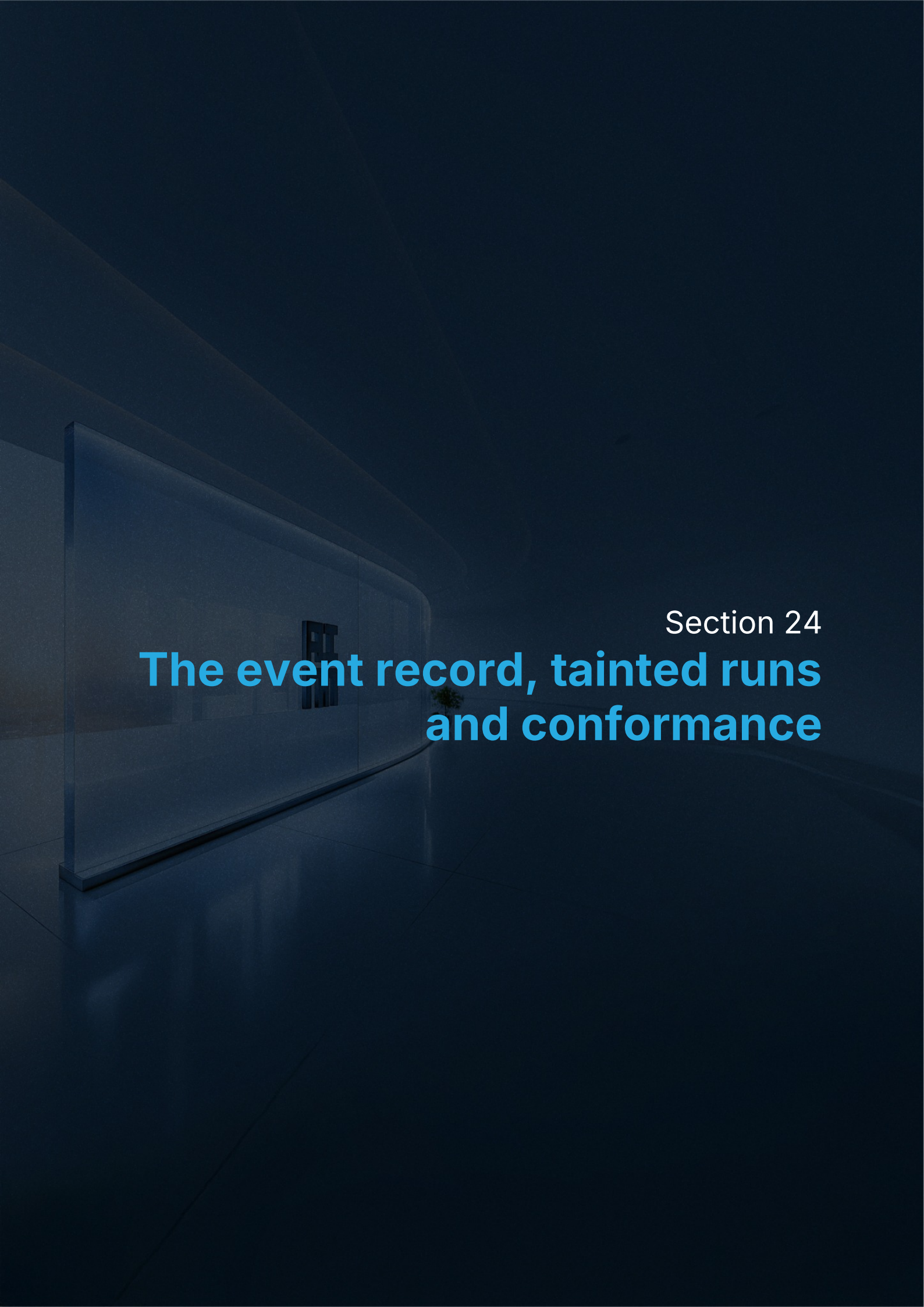
1. **Its place.** The root AIOC, and the core of the [ORG]AION, at the governance level. Its top Cell, the one carrying [ORG]HOAC-GOV, is the only Cell with no accountable_to (Standard §1.7, item 13), and every other part is installed under it. [ORG]AIOC-OCO prepares it; GOV installs it and signs every change to it. OCO never installs or changes it alone. As the core, it holds the human position of the governance seat, which the core's AI serves and never holds, the governance panel and the organisation's own Standard, Blueprint and Library; the Top Management position sits in the Top Management Group, the first AIOC under the core.
    - **The gateway and the stop beside the core.** They are installed with the core but run as code beside it, outside the reach of every AI, the core's own included. The core's own stop is held directly by the governance seat. Any change to the code of the gateway or the stop counts as loosening: it needs a human signature and is made by a party other than the AI it constrains. At the handover the organisation issues fresh gateway keys, and every key Edition Maker made, held or used is revoked.
2. **What it holds.** The Cell that carries [ORG]HOAC-GOV, with its Office; and three Pods of shared services: Gateway, Records and Work. The Cell that carries [ORG]HOAC-TM, with its Office, sits in the Top Management Group, the first AIOC under the core (item 1; the complete definition, §19); the Charter names it where top management signs, owns a test or receives a raise (49GAION/DR-0052). Each Pod follows the Protocol sections named below; this section adds only what joins them.
3. **The Gateway Pod.** The tool gateway of §4.9 and Standard §4.4, item 11.
    - Every tool and engine call of every part passes through it.
    - It checks each listed type against the loaded signature list, binds each engine from its engine register (49GAION/DR-0020), logs every call, and fails closed.

- It serves the human surface: the live sign-in behind every HU port.
- A stop, a disconnection, a narrowing of AI access and a raise to a human are never blocked.
- Its policy, its map from calls to decision types and its keys change only under change-gateway-policy.

4. **The Records Pod.** §14, held once for the whole organisation.
   - **The registry** declares every record type of every installed part. A part's installation adds its types, under the same signature that installs it.
   - **The numbering service** issues every readable number: one stream per record type (§14, item 4), and per entity where the type is kept per entity; atomic and safe to retry. No writer mints its own.
   - **The event store** serves every part's record home (§14, item 1). Each part writes only its own events; the store keeps them apart, append-only.
   - **Projections** are rebuilt from the events, each with the version of its logic (§14, item 2). **The daily check** of §14, item 5 runs over every part, and a failure opens a defect for the part it concerns.
   - **The restricted partition** of §14, item 6 is read only by the positions it names.
5. **The Work Pod.** The shared queue of §4.
   - Every Work Item of every part lives in it, with one owner and one writer (§4.3), its statuses (§4.2) and its waiting times (§4.6).
   - **The watchdog** raises each item past its waiting time, as §4.6 sets.
   - **The routing data** of §6 is kept by [ORG]AIOC-OCO, which dispatches nothing, or, where an organisation has no separate [ORG]AIOC-OCO, by the hub of the operation; the router reads it and applies the receivability check.
   - **Wake** follows §7: each part is woken from its own queue.
6. **One of each.** An [ORG]AION has one tool gateway, one numbering service and one shared queue. A part that keeps its own queue, mints its own numbers or calls a tool outside the gateway fails its joining test (§11, item 1).

7. **No AI holds the keys.** No AI Cell holds write rights to the gateway's policy, keys or log, to the registry, or to a number stream. An AI Cell may only call them. The Pods are accountable to the Cell carrying [ORG]HOAC-TM, and GOV signs their changes.
8. **The order of starting.** At installation the Pods start in this order: Records, then Gateway, then Work. No other part is installed before all three pass their cases of TEST-CHARTER v1.
9. **Never blocking the accountability path.** A stop, a revocation and a raise to a human take effect at the gateway at once, and never wait on the Records or Work Pod. They are recorded afterwards, from the gateway log.
10. **With Engine Room.** Engine Room supplies the engines the gateway binds and never replaces the gateway (49GAION/DR-0026, item 3).
11. **Roles, not tools.** The Pods are named by role; tool names appear only in an [ORG]AION (§10, item 5).
12. **The fixed test.** Every rule of this section has a case in TEST-CHARTER v1 (library/cards/README.md). The card stays DRAFT, with last_result NOT_RUN, until the test has run once with recorded evidence (§10, item 3).

Sources: DR-0029, DR-0041, DR-0042.



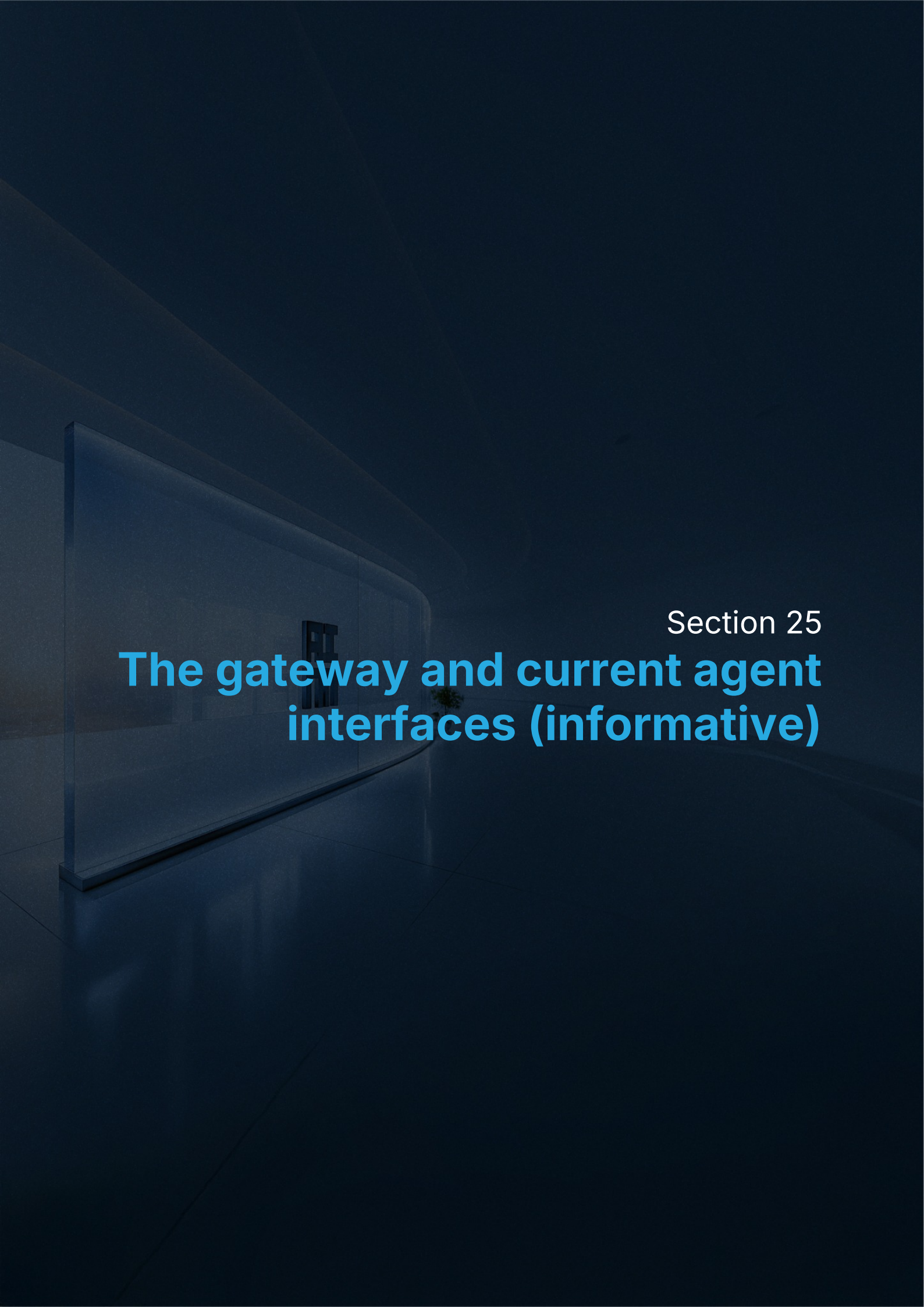
Section 24

The event record, tainted runs and conformance

1. **One record (49GAION/DR-0053).** Every event at the tool gateway is one entry of an append-only record in one envelope, spec/schemas/event.schema.json: a tool call (its request and its run), a gateway decision, an approval, a stop, a restart, a change of a control, a credential put in force, a grant to a Chief of Staff, a taint label and a clearance. Each entry carries its event ID, the run, its place in order, the time, the position and the credential that act, and the hash of the entry before it, so that an entry deleted or altered mid-record breaks the chain (Standard §4.4, item 24). The record names parties only by position code and credential reference: no person and no reference to one.
2. **A profile of the semantic conventions for generative AI.** The envelope's gen_ai group follows the OpenTelemetry semantic conventions for generative AI at the version the schema pins: a tool call is the operation execute_tool with its tool's name, and a call through the Model Context Protocol carries its method and tool. 49GAION's own fields sit under gaion, the name of the server of such a call among them.
3. **The basis of every decision.** A gateway decision names the rule it applies, the version in force (the release tag and the commit it names) and the record that authorises it. A decision without its basis, or whose commit is not the one its tag names, does not count.
4. **Authority along a chain.** An act done for another position names the position it is done for, the chain of parts it passed through, each with its permissions and its taint, and the permissions it uses, which lie within those common to every part of the chain. A part never does for another what that other may not do.
5. **Tainted runs.** Every input carries a source label, internal or external, with its source. A run that has read an external input is tainted until it ends (Standard §4.4, item 22). Reading, drafting and writing inside the part's own space stay open; a send, a publication, a write outside the part's space, a write to a new target and a change of permission run only after a clearance: a deterministic check that passed, or the accountable position's approval, each recorded. What a tainted run stores keeps its label, and a run that reads it is tainted. The gateway, not the part, records the taint.

6. **Credentials in the record.** A credential reference names its kind: a human position's credential, an AI Cell's own, or a deterministic part of the core such as the gateway. A human position's credential counts toward the approvals of a listed type, and so does its own Chief of Staff's credential acting under a grant of that position recorded earlier in the record whose reach covers the type (item 9); the Chief of Staff's approval counts as the credential that signed the grant, never as a second one. No other AI Cell's credential and no deterministic part's counts.
7. **The conformance scenarios.** The traces under tests/traces/ are the scenarios: each trace that must fail breaks one rule (TRACE-CHAIN, CALL-WITHOUT-DECISION, CALL-AFTER-STOP, RESTART-BY-STOPPED, AUTHORITY-ESCALATION, HUMAN-FLOOR-QUORUM, FRESH-CREDENTIAL, TAINTED-RUN, DECISION-BASIS), and each trace that must pass shows the same act done right. tests/traces/check_traces.py checks them, and fails if a trace that must pass has a finding, so a checker that always refuses fails too.
8. **What conformance means.** The traces here are written by hand: they test the checker, not an implementation. An implementation conforms when its own gateway, run on the same scenarios, records traces of its own that pass the checker (Standard §1.8, item 2). That gateway lives in the implementation's own repository, whose checks run this kit at a released tag; until then its claim reads "unverified".
9. **A grant (49GAION/DR-0053).** A grant is an event of the record: the position that gives it, with its own credential in force; the Chief of Staff that receives it, by Cell code and credential, and the position that Chief of Staff serves; what it reaches (listed types, or all); whether it lets the Chief of Staff give access to another AI part; and when it ends. A grant may be revoked by a later grant event of the same position. A position grants only to its own Chief of Staff. A Chief of Staff gives a grant to another AI part only under a grant of its position that lets it, within that grant's reach and end; a part that receives such a grant never signs for the position. Any other grant fails AUTHORITY-ESCALATION; an approval of a listed type by a Chief of Staff with no grant in force that reaches it fails HUMAN-FLOOR-QUORUM.

Sources: 49GAION/DR-0053.



Section 25

The gateway and current agent interfaces (informative)

This section maps the gateway onto interfaces in use today. It adds no rule; where it differs from a rule above, the rule governs.

1. **Tool calls through the Model Context Protocol.** The gateway sits between every agent and every server it calls. Each call to a tool is a tool-call request in the record (§24, item 1), judged before it runs; a server is an external attachment, reached through a port of the Cell that uses it (Standard §4.4, item 7). The tools a part may call are its scope.
2. **Credentials.** Each AI Cell calls with its own credential (§10, item 3). The keys to outside servers are held at the gateway; no part holds one.
3. **Calls from one agent to another.** A request from one agent to another, inside the organisation or across it, is a Work Item between two nodes (§4). The agent that receives it acts for the position at the start of the chain, within the permissions common to every part of the chain (§24, item 4): a request never gives the receiver more than the asker may do.
4. **The stop.** A stop revokes the part's credentials at the gateway, which ends every open session and task of that part at once and refuses every queued call (§4.4).
5. **Taint across interfaces.** What a server or an agent outside the organisation returns is external input, and taints the run that reads it (§24, item 5).

Sources: 49GAION/DR-0053.


49G AI
49G HOLDING | 49G AI
www.49g.com