

Control Evidence Object Model

Version 0.9.10 — circulated for challenge, not adoption

Published record. This document is deposited at Zenodo with the persistent identifier DOI **10.5281/zenodo.23687472**, which is the citable form of this work. Later versions supersede it under the same concept DOI.

Licence. This specification, its field dictionary and its controlled vocabularies are released under **Creative Commons Attribution 4.0 International (CC BY 4.0)**. The JSON schema and the reference validator are released under the **Apache License 2.0**, which carries an express patent grant. Commercial implementation is permitted. Attribution is required.

How to cite. Wiggins, W. (2026). *Control Evidence Object Model, version 0.9.10*. Alscale Ltd. DOI 10.5281/zenodo.23687472

Status. Pre-1.0. The twelve objects, the four layers, the evidence scale, the reference states and the gap scale are stable. Field-level detail may still move. **Annex D states what conformance means** and is testable against the reference validator.

What is published and what is not. This is the model. The assessment practice built on it, including report formats, scenario method and position library, is not published and is not required in order to implement the model.

Changes from v0.9.9. Two documentation changes and no field moves. A **tenth output** is listed: what the records say would happen if somebody went for a named asset. It is a traversal like the other nine, it was already being produced by hand, and it is the form in which the question is actually asked out loud. And Annex A gains an entry for **model**, which is the most overloaded word in this document: the specification calls itself "this model" throughout while `Asset.kind` carries `model` meaning a language model, and one glossary entry already uses the word in the second sense. The glossary's own test is that an entry is warranted where two participants demonstrably use the same word for different things, and two did. Recorded at D120.

Changes from v0.9.8. One value, and it closes a hole that caused five rounds of talking past each other. `Action.mode` has carried `permitted` and `performed` since v0.7. There are three positions an assessment reports on, not two: what the grants say the agent may do, what the person accountable says it should do, and what it did. The middle one had no name, which is why every discussion of it slid into the vocabulary of the other two. **`Action.mode` is now `permitted`, `expected`, `performed`.** Whose expectation it is records as a Decision, so `Decision.kind` gains `set_expectation` and no field is added anywhere. Three gaps then fall out of the naming instead of having to be asserted, and the fourth grade of the gap scale is explained: it exists only where the comparison runs against `permitted`, because what the records support is inferred, and an observation always establishes something. Recorded at D118.

Changes from v0.9.7. The model gains the word for the thing it has always been about. An **inventory** is what a firm should hold for one object; a **register** is an artefact in which some of it is actually recorded; and there are zero or many registers per inventory. The word inventory appeared nowhere in this specification before now, while register appeared twenty-four times, so

the proposition could not be stated in its own vocabulary. Where several registers exist for one inventory, the designation of which one governs records as an Interpretation with no Obligation, reusing a route the model already had; no such Interpretation means nobody has designated one, and that is a finding. And the **reference states** are stated here for the first time, as a third scale beside the evidence scale and the gap scale: seven states, three of which require a reason. The seventh exists because a reference can resolve to a register that holds no entry for the identifier cited, which is worse than a register nobody maintains and is invisible until somebody opens the register. Recorded at D116 and D117.

Changes from v0.9.6. Two open items closed, both raised by v0.9.6 itself.

`Exception.status` gets five permitted values, and `recorded` is the default: a divergence that is written down with nobody having taken a position on it is the commonest real state and no vocabulary in use has a term for it. The other four require something that resolves, so a status cannot be claimed without the thing that would substantiate it. And the four pointers that look like one relation are confirmed as four: `derived_from` is provenance, `source_ref` is location, `established_from` is justification, `value_ref` is authority over a value. A proposal to collapse the last three into refinements of the first was **rejected**, because all four can resolve to different Assets on one record. Two consistency rules between them are now stated, and both are instances of the shape this model keeps finding: the thing being checked is also the thing supplying the answer. Recorded at D114 and D115.

Changes from v0.9.5. Four field-level changes, all of them narrowing rather than adding. `derived_from` becomes a shared trait on every object, so the question already asked of three objects under three names (`source_ref`, `established_from`, `value_ref`) is asked of all twelve: a record cannot evidence the thing it was derived from. The evidence scale gains a condition on its top state, because a record hard coded inside the subject previously scored COMPLETE, which made the highest score available to the architecture this model argues against. `Owner.person` is nullable, so a recognised role with nobody holding it is recordable and is no longer indistinguishable from the role not existing. And Authority gains `approved_from`, the Asset the approver relied on, because the model recorded the establisher's homework and not the approver's. All four come from checking the model against eleven published standards and frameworks, recorded at D106 to D113. **Six changes were proposed and four were withdrawn on reading this specification**, because the model already carried them; those are recorded as corrections at D109, D110, D111 and D112 rather than acted on.

Changes from v0.9.4. The evidence scale is renamed. HELD becomes COMPLETE and ABSENT becomes SILENT; PARTIAL, UNKNOWN and NOT ENGAGED are unchanged. The meanings are untouched. HELD was records-management jargon and the only verb participle in a set of adjectives, so the scale did not read as a scale; ABSENT and UNKNOWN sound like near-synonyms to a general reader, which is dangerous in a model whose standing rule is that those two are never collapsed. Renamed now because in this vocabulary the term is its own identifier, so the cost is a regeneration today and a broken mapping once anyone outside has declared against it.

Changes from v0.9.3. `Authority.granted_by` may now resolve to an Asset as well as an Owner, so a grant conferred by a mechanism rather than a person can be recorded at all — and reads, correctly, as a grant nobody conferred. Action gains `output_sensitivity`, the class of

what an action put into its destination, stated against the same scale as `Asset.information_class` so the two compare directly; null means the target asset's class stands. And the specification now states that a limit whose `value_ref` resolves to an Asset the grantee controls or produces is recorded as unenforced. All three come from a tier 1 and tier 2 assessment of a live multi-tenant data quality agent, together with eight written criticisms from the team that built it. **Five of those eight were already answered**, two of them by decisions this model had taken before that team had seen it; those are recorded at D104 rather than acted on.

Changes from v0.9.2. A prompt is an Asset of kind `prompt`, and a licence a prompt grants is an Authority record whose `source_ref` is that prompt and whose `enforced_by` is the model — so a licence nothing enforces is visible on the face of the record. The **gap scale** is now stated here rather than held in the assessor's head, and its third grade gains four reason codes, because four different defects were collapsing into one grade and they are closed by four different people. A proposal to add a durability attribute to that scale was **rejected**: D89 already answers it with two Authority records. Prompted by a tier 2 dry run against a constructed insurance claims agent, then scored against the pack author's answer key, which is where the rejected proposal came from.

Changes from v0.9.1. Authority gains `enforced_by`, a reference to the Asset that enforces a grant, so a grant nothing enforces becomes visible rather than inferred. Obligation gains `part_of`, an optional reference to a parent Obligation, so a requirement can be decomposed and reported at one level while agents link to another. Obligation widens from regulatory sources to regulatory **or contractual** ones, since a contractual requirement is imposed rather than chosen and was excluded by the old wording. An approved AI use case is recorded as an Interpretation with no Obligation, using a route the model already had. Design principles 9 and 10, the as-at resolution rule and the acyclicity rule, are new since v0.9.1 along with the firm-local scale and jurisdiction encoding rules; every field now carries a stated type.

Changes from v0.9. Dependency gains a fourth group, `containment`, with one term, `resides_in`, recording which asset holds another. Without it there was no traversable path from a physical data element to the system holding it, so reach could not be computed and the six location facts had to be duplicated onto every element. Interpretation gains the caution already carried by Permission: the model records that a position exists, not whether it is correct.

Changes from v0.8. Authority gains four fields recording how the grant was established and approved — `established_by`, `established_from`, `approved_by`, `approved_at` — with the constraint that the establisher and the approver must differ. This makes visible the case where capability was granted on an unverified assertion with no approval step, which is how most self-serve access works and how at least one documented agentic campaign was mounted.

Changes from v0.7. The Action vocabulary grows from nine verbs to eleven, across five groups. `retrieve` is replaced by `access` and `read`, distinguished by whether a copy persists in the actor; `extract` is added as its own group, for a copy that leaves. Action gains `destination_asset_id` and `retention`. Authority is restated as one grant per verb-asset pairing rather than arrays, because real grants are not rectangular. Dependency gains `refresh_mode` and `last_refreshed_observed` for derived copies. Asset gains an `index` kind.

Changes from v0.6. `pipeline` renamed to `data_pipeline` — a homonym in the energy, transport and water sectors, which are in scope. A twelfth object, **Permission** — the position a data owner takes on an asset being used for a purpose. `sensitivity{}` added to Asset, six facts, same pattern as `location{}`. Fault vocabulary explicitly excluded and assigned to the engine. Exception vocabulary unchanged after examination — see the note at object 9.

Changes from v0.5. `source_ref` on Interpretation and Authority must now resolve to a registered Asset rather than to free text. Location is not duplicated onto those objects; it is reached by traversing to the Asset, which is where all six location facts live.

Changes from v0.4. Restructured so each object is complete in one place — definition, every field with its business label and what it holds, and the permitted values inline where a field is constrained. The separate taxonomy and field-label annexes are gone; their content now sits with the object it belongs to. "Taxonomy" replaced by "controlled vocabulary", which is what these actually are. Nothing was removed.

Changes from v0.3. `instrument_ref` renamed to `source_ref` — "instrument" collides with its financial meaning in the audience that matters most. `introduced_by` added, separating the rule provision from the publication that introduced it. A business label added for every field, because half the field names are legible to a data team and opaque to the people who own the objects.

Why this exists

Supervisors have stopped testing outputs and started testing the chain behind them.

Rules, policy, data, systems, AI action, decision, evidence — and of each stage: can this be explained, can it be evidenced, can it be replayed under challenge.

That change has dates attached to it. Basel 3.1 on 1 January 2027. Operational incident and material third-party reporting on 18 March 2027. T+1 settlement on 11 October 2027. EU AI Act Annex III obligations on 2 December 2027. The UK critical third parties regime is already live.

The chain breaks in three distinguishable ways, and organisations have organised themselves accordingly. Reporting failures are visible — a submission is wrong and someone eventually notices. AI failures are fast — an agent acts thousands of times before anyone reviews one. Resilience failures are boundary failures — something outside the organisation stops working, and what can be evidenced afterwards depends on where the evidence was held.

Each has capable tooling. Between them they cover most of the chain.

The gap

Nothing joins their outputs, because they do not share a vocabulary.

The clearest illustration is the word *break*. Across the vendors serving this space, a break is variously an interpretation failure, a data defect, a configuration defect, a transformation error, a rejection, or a correction. Each definition is correct inside its own tool. None of them compose.

The consequence is stated plainly by people who have looked: **no single evidence manifest spans rule version, data, configuration, test, approval, correction and final report**. Not because such a manifest is hard to build, but because there is no shared identity between the records that would populate it.

The same failure appears wherever separate functions hold part of the picture. Eleven functions each hold a register. Each is well maintained. None shares an identifier with any other. Asked a question that crosses two of them, an organisation assembles an answer by hand, and the answer is a project rather than a query.

This is not a tooling gap. It is a definitional one — and facilitation, workshops and better cooperation do not fix definitional problems.

What this is

A controlled vocabulary. Twelve objects, their permitted values, definitions for the words that mean different things to different people, and enough machine-readable identity to let one party's output join another's.

The objects are not AI-specific, and that is deliberate. An agent is an `actor_type` on Action, alongside human and system. The claim the model makes is that an agent's action is governed by the same objects as a person's — the same authority, the same owner, the same exception, the same evidence. Giving AI its own object would collapse that claim and build a parallel governance track, which is the fragmentation the model exists to remove.

But the pressing case is autonomous actors, and that is why this exists now rather than a decade ago. When a person acts, their authority is implicit in a role and a job description, they act a handful of times a day, and they notice when the ground moves under them. An agent has none of those. Its permissions sit in a configuration file nobody signed. It acts thousands of times before anyone reviews one. And when a rule changes or a control is downgraded, it carries on doing exactly what it did yesterday, correctly by its own lights, and is now wrong.

So: **a general model, sharpened by a specific problem.** It works for a human submitting a report. It is *necessary* for an agent operating inside a business process.

What it is not. Not a product and not an architecture. Not a competitor to CDM or FIBO — those describe transactions and financial products, whereas this describes the control evidence around a decision, whoever or whatever made it. And not a governance platform: it does not enforce, monitor or block. It says what shape a record must take so that records from different sources can be joined.

Ownership. Proposed for open publication. No vendor owns it, nobody is advantaged by it, and any party may map their own vocabulary into it.

The four layers

The model is four things, frequently confused with one another. Each is separable, each has a different audience, and each can be adopted without the one below it.

Layer	What it is	What it answers
1. Objects	Twelve entities with identity, attributes and relationships	<i>What things do we keep records about, and how do they connect?</i>
2. Controlled vocabularies	Closed lists of permitted values, with hierarchy, for the nine constrained fields	<i>What values may this field take?</i>
3. Glossary	Definitions of contested terms, with owners	<i>When two people say "material", do they</i>

	and dates	<i>mean the same thing?</i>
4. Semantics	Namespace, resolvable identifiers, declared mappings	<i>Can a machine tell that your term and mine are the same term?</i>

Layers 1 and 2 are the interoperability contract — the objects establish what exists, the vocabularies establish what may be recorded. Layer 3 stops the contract being read differently by each party. Layer 4 lets it operate at scale without a human in the loop for every join.

Structurally the model is a graph — polymorphic nodes, typed directed edges, traversal as the primary operation. **Operationally it reports as tables**, because that is what a supervisor, a committee pack or a spreadsheet consumes. Every one of the nine outputs below is a tabular projection of a graph traversal. Both statements are true and neither implies a storage technology.

Design principles

- 1. States are versioned; nothing is overwritten.** Every versioned object carries `effective_from` and `effective_to` and is never overwritten. Events are not versioned: Action, Exception, Decision and Evidence happened once, and carry `occurred_at`, `detected_at`, `decided_at` or `captured_at` instead. Replay requires reconstructing what was true *as at* a moment, not what is true now.
- 2. Identifiers are stable and meaningless.** No smart keys encoding regime, entity, date — or position in a hierarchy. Those change; identity must not.
- 3. One object never carries two fields called `status`.** Lifecycle state and workflow state are different things. If an object needs both, one is wrong or belongs elsewhere.
- 4. Four kinds of time, kept separate:** `occurred_at`, `detected_at`, `recorded_at`, `decided_at`. Most reporting failures live in the gaps between them, and collapsing them destroys the evidence of the failure.
- 5. Assets are polymorphic.** One object with a kind discriminator, not seven parallel registers. A dependency graph must traverse a data element, a model and a vendor uniformly or it is not a graph.
- 6. The model records what happened, including what should not have.** Governance rules belong in assurance, not in the schema. A record you cannot write is a failure you cannot evidence.
- 7. Location is recorded once, on Asset.** Anything needing to know where something sits, who can reach it or how long it is kept traverses to the Asset. Duplicating those facts guarantees divergence.
- 8. The model governs the seams, never the contents.** It specifies identifiers, enumerations and links. What an organisation holds inside each register is its own business.
- 9. A reference to a versioned object resolves as at the referring record's event time.** Identifiers are stable, so `OWN-0003` names an identity rather than a row, and without a resolving date two implementers will reach different rows and both will believe they are right. The resolving date is the referring record's own event time: `Decision.answerable_for` as at `decided_at`, `Action.authority_id` as at `occurred_at`, `Evidence.captured_by` as at `captured_at`. Where the referring record is itself versioned, the resolving date is its

`effective_from`. Like the ownership inheritance rule, this must be written down or it will be resolved inconsistently.

10. **Same-object references form a tree, never a cycle.** `Obligation.part_of`, `Interpretation.supersedes` and `Owner.delegated_from` each point at their own kind. A cycle makes every upward traversal non-terminating, and the traversals are the point. Loading rejects one rather than discovering it on the first report that never returns.

Naming conventions

- **Field names are unqualified by their parent object.** `Obligation.source_ref`, never `obligation_source_ref`. The object supplies the context; repeating it lengthens every join and bakes the parent's name into the child.
- **The same concept takes the same name everywhere it appears.** `source_ref` on `Obligation`, `Interpretation` and `Authority` — one concept, one name, one glossary entry.
- **Reference fields are named for the role they play**, not for the object holding them: `decided_by`, `granted_by`, `detected_by` all point at `Owner` and are not duplicates.
- **`kind` is the single discriminator name.** `Action.verb` and `Action.mode` survive as distinct names because they mean something other than "what sort of thing is this".
- **Every field has a business label**, stated with the object. Field names are for the schema; labels are for the people who own the objects. Rendering a raw field name to a business audience is a failure of the artefact, not of the name.

Shared traits

Rather than repeat them per object: every object carries `id`; every versioned object carries `effective_from` and `effective_to`; every object carries `recorded_at`, `recorded_by` and `derived_from`.

`derived\_from` points at the object or Asset this record was made from. It resolves or it is null, and null means nobody could say where the record came from. It exists because the question was already being asked of three objects under three names, each added when somebody noticed it mattered there: `source_ref` on `Interpretation` and `Authority`, `established_from` on `Authority`, `value_ref` on a limit. Those three are retained, because each says something more specific and each carries its own finding. The trait asks the general question of the other nine.

Four pointers look like one relation and are not. The difference is stated here once rather than argued per object. All four can resolve to different Assets on a single record, which is the test: a grant written into a configuration file, copied from a vendor template, approved on the strength of a contract, carrying a threshold read from a rate table, has four different answers and the case is not unusual.

Pointer	The relation	Reads as
<code>derived_from</code>	Provenance	This record was made from that
<code>source_ref</code>	Location	This record is held in that
<code>established_from</code>	Justification	Somebody relied on that in deciding
<code>value_ref</code>	Authority over a value	The live number is maintained at that

Where `source\_ref` and `derived\_from` resolve to the same Asset, the record is held in the thing it was made from. That is a carried copy: a value sitting in the subject, with the subject as both its home and its origin. It is the condition the evidence scale now scores down, and stating it this way makes it computable rather than a judgement about what a record looks like.

Where `established\_from` and `derived\_from` resolve to the same Asset, the justification is circular. Somebody decided a grant was needed on the strength of a document generated from the grant. It is the same failure as a declaration layer built from the code it declares, one level up.

The rule it supports, stated once rather than per object: a record cannot evidence the thing it was derived from. Three constraints already in this specification are instances of it. The approver must differ from the establisher. A control enforced by the same Asset it governs carries no independent statement. A limit whose value the grantee maintains is not a limit. The trait makes the same test available everywhere, and a record reconstructed from the implementation it describes is set aside from comparison rather than counted as agreement.

Inventories and registers

Three words, and conflating them has caused more confusion in conversation than any other pair of terms in this model.

Term	What it is	How many
Inventory	What this model expects a firm to hold for one object: a complete set of records for that object, maintained by somebody	Twelve. One per object
Register	An actual artefact in which a firm records some or all of an inventory. A spreadsheet, a table, a document, a module in source	Zero or many per inventory
The governing register	Where several registers exist for one inventory, the one the firm designates as authoritative	One per inventory, or none

The twelve objects are inventories, not registers. The model says what a firm should be able to produce for each object. It does not say how many artefacts that is held in, and in practice it is rarely one. The opening of this specification already uses the case: eleven functions each hold a register, each is well maintained, and none shares an identifier with the others. Eleven registers, one inventory.

Multiplicity is not untidiness, it is where findings come from. On one assessment, five of twelve inventories were held across eleven registers. Seven reported contradictions were very largely the consequence: two registers for one inventory, disagreeing, with nothing designating which governs. As a list of seven they read as documentation hygiene. As one structural fact they read as what they are.

Which register governs is a position the firm takes, so it is an Interpretation. Not a field, and not a thirteenth object. See object 2. Where no such Interpretation exists, nobody has designated a governing register, and the question of whether an actor is operating on the right record has no answer available even in principle.

What stays out of scope. Whether a register is internally accurate is not this model's business, on the same footing as whether an interpretation is correct. Which of several registers governs is not a question about accuracy at all.

Layer 1 — The twelve objects

Each object below is complete in one place: what it is, every field it carries with its business label and what that field holds, and — where a field is constrained to a fixed set of values — the permitted values inline.

Four fields are on every object and are not repeated below: `id` (stable, meaningless), `recorded_at`, `recorded_by` and `derived_from`. Objects marked *versioned* also carry `effective_from` and `effective_to`, and are never overwritten.

1 • Obligation — *versioned*

A single discrete requirement, derived from a regulatory or contractual source. The source is not the obligation: FCA Policy Statement 26/2 is a source; *"maintain a register of material third-party arrangements and submit it annually"* is an obligation. One source yields many.

Field	Label	What it holds
<code>source_ref</code>	Source	The registered Asset the requirement is published in — a rulebook, a standard, a code of practice, a contract. Must resolve to a registered Asset , not free text.
<code>source_locator</code>	Where in the source	The citation within that document, as the source itself numbers it. "Principle 3, paragraph 36(c)". "Article 111".
<code>source_version</code>	Source version	The version of the requirement identified by <code>source_ref</code> and <code>source_locator</code> together, as that source identifies it — a version number where the source has one, a date-state where it does not. Positions within one document are amended on different dates, so this belongs to the position rather than to the document.
<code>in_force_from</code>	In force from	The date the obligation comes into effect. Not the date its source was published, which is frequently much earlier and is a fact about the document rather than about the requirement.
<code>introduced_by</code>	Introduced by	The publication that brought it in — a policy statement, consultation outcome or technical standard. How regulatory change functions actually track it.
<code>jurisdiction</code>	Jurisdiction	Whose rulebook this comes from.
<code>applicability</code>	Who it applies to	Which entities, products and transactions the obligation binds.
<code>part_of</code>	Part of	The parent Obligation this one decomposes. Optional, resolved as at <code>effective_from</code> under principle 9.
<code>owner_id</code>	Obligation owner	The firm's owner of this obligation record. Accountable for it being correctly stated, correctly scoped, and still current. Not for the requirement existing.

Two references, deliberately. Firms need both `source_ref` and `introduced_by` and usually hold only one. The provision is what a supervisor cites; the publication is what a change function tracks.

`applicability` is the field most often skipped and the one that decides whether an obligation applies to this organisation, this function, this transaction. Leaving it null is how scope arguments start.

Owning an obligation means owning the record, never the requirement. Nobody in a firm owns BCBS 239. What is owned is the entry: that it states the requirement correctly, that its grain and scope are right, and that it is revisited when its source changes.

Where the source is external, that accountability is surveillance rather than authorship. A source is external where its Asset carries a `provider_id`. The owner of an internally sourced obligation can change the document when it is wrong. The owner of an externally sourced one cannot, and their job is noticing when it changes. That second kind fails silently, because nothing happens when nobody watches, and a firm ends up relying on a superseded standard without knowing it.

Ownership attaches at whatever grain the firm chooses and inherits downward through `part_of`, under the object 4 rule. At paragraph grain one standard yields sixty or more obligations and a named owner on each is unworkable; one owner per standard, or per principle, inheriting to the rows beneath, is not.

An external source is still a registered Asset, and it has both an owner and a provider. `provider_id` names who published it — the Basel Committee, the PRA, a counterparty. `owner_id` on that Asset names who in the firm is accountable for the copy being current. A firm that cannot fill the second cannot know the standards it works from are the current ones, which is the same failure as lost analysis one level higher up.

Regulatory sources come in two shapes and the model holds both.

Rulebook-based. A publication amends a rulebook and the requirement lives in the rulebook. The FCA Handbook, the PRA Rulebook, the CRR. `source_ref` is the rulebook, `source_locator` is the article, and `introduced_by` is the publication that amended it. The two differ.

Publication-based. The requirement is in the publication itself and there is no separate provision. Basel Committee standards, EBA guidelines, the NCSC CAF, ISO standards, most codes of practice. `source_ref` and `introduced_by` point at the same Asset.

That equality is the signal, and it is queryable. `source_ref` equal to `introduced_by` means the requirement is self-contained in its publication. Not equal means it lives in a rulebook something else amended. `introduced_by` is never left null to mean this, because null already means not recorded, and SILENT and UNKNOWN are not the same finding.

Why a locator rather than registering each provision as an Asset. Firms work at paragraph grain or finer. An Asset is a thing in an estate with an owner, a location and a retention period; four hundred cited paragraphs of someone else's standard are none of those. The Asset is the document. The locator is the position inside it.

`jurisdiction` may be null, and the null is informative. A Basel Committee standard is not anyone's rulebook and binds nothing until a supervisor adopts it. Recording a home regulator there would assert a binding nobody can cite. The adopting instrument is its own Obligation, carrying the jurisdiction that actually binds, sitting beneath the standard through `part_of`.

A caution on `applicability`, recorded rather than fixed. The field holds which entities, products and transactions the obligation binds. For a standard reaching a firm through supervisory expectation there are three different answers: what the standard says it covers, what the adopting instrument says, and what the firm decided about itself. The first two are facts about sources. **The third is a position the firm took, which makes it an Interpretation.** The field currently carries both a fact and a judgement, which is the confusion objects 1 and 2 exist to prevent, sitting inside object 1. Left for contact with a real inventory.

`part\_of` lets a requirement be decomposed. Own funds requirements, then calculate regulatory capital, then calculate exposure value. Grouping by `source_ref` gives one level only, and one level is not enough: an agent links to the finest requirement it actually performs, while a committee reports at the level above it. The controlled vocabularies carry groups for the same reason — report high, detect low.

The inheritance rule, written down. A child takes its parent's `applicability` unless it states its own. Where it states its own and the two differ, **the child's stated value is the record** and the divergence from its parent is reported by the assessment. There is no field for a difference, and there should not be: as with the ownership inheritance rule, what matters is that two people resolve it the same way.

A control discharging a parent does not thereby discharge its children.

`Control.obligation_ids[]` points at whichever level the control was designed against. Coverage reported by rolling children up to a parent will overstate, and reported without rolling up will understate. Say which was done.

A contractual source is an Obligation like any other. A client contract or an outsourcing agreement imposes requirements rather than offering positions, which is the test objects 1 and 2 turn on. A regulator permission such as an IRB approval is an Asset of kind `document`; its conditions are Obligations whose `source_ref` points at it, scoped by `applicability` to the approved portfolios, and sitting beneath it through `part_of`. **It is not an Authority**, because `granted_by` must be an Owner and a regulator holds no standing accountability inside the organisation.

Vocabulary note. Earlier drafts used "instrument" in its legal sense. In an audience trading financial instruments that is a homonym clash, and a tradeable instrument is an Asset in this model. "Source" throughout, since v0.9.2 admits contractual sources alongside regulatory ones.

2 · Interpretation — *versioned*

The position the organisation took on what a requirement means. The object nobody models, and the origin of a large share of control failures. An Obligation is imposed; an Interpretation is a position the organisation chose.

Field	Label	What it holds
<code>obligation_id</code>	Obligation	The requirement being interpreted. May be null, where the position is internal policy rather than a reading of a rule — which is how non-regulatory cyber and security policy enters the model.
<code>position</code>	The stated position	What the organisation decided the requirement means for it.
<code>rational</code>	Reason for the	Why that reading was taken.

	position	
decided_by	Decided by	The Owner who took it.
source_ref	Source document	The Asset the position lives in — a policy, change memo, operating procedure or delegated reporting agreement. Must resolve to a registered Asset , not free text.
supersedes	Replaces	The interpretation this one replaces.

An approved AI use case is an Interpretation with no Obligation. An Obligation is imposed; an Interpretation is a position the organisation chose, and a use case is chosen. So it needs nothing new: `position` carries the use case statement decomposed into testable statements, `decided_by` the approver, `source_ref` the use case register entry as an Asset, and `supersedes` the version it replaces. Approval is a Decision referencing it, as open question 1 settles for every Interpretation.

A designation of which register governs an inventory is also an Interpretation with no Obligation. Where a firm holds several registers for one inventory, the statement that one of them is authoritative is a position it chose, and it records with `position` carrying the designation, `decided_by` whoever made it and `source_ref` the document that states it. Nothing new is needed. **The absence of such an Interpretation is the finding**, and a sharp one: where nothing governs, no reading of any register can establish whether an actor was operating on the right record.

Three things with a null `obligation_id` are told apart by `source_ref`, not by a new field. Internal policy points at the policy document; a use case points at the use case register. Both are Assets, both have owners, and the distinction is carried without `Interpretation` growing a `kind` it does not need. Adding one would put a classification inside the model that the registers already make.

Where the approver is also the builder, record it. The same rule Authority applies between `approved_by` and `established_by` holds here: a position approved by the person who proposed it has not been approved. With citizen development this will be common, and it is a finding rather than an inconvenience.

`source_ref` must resolve to a registered Asset, not to free text. A filename someone remembers is not a source. Requiring the document to exist in the estate as an Asset means it acquires an owner, a retention period and a `location{}` — and it is the difference between holding a reference and being able to produce the document under challenge.

Location is deliberately not duplicated onto this object. Where the source document sits, who can reach it and how long it is kept are all facts about the Asset, reached by traversing `source_ref`. One place to record them, one place to correct them.

Where a model rather than a person derives the mapping, the interpretation sits in the weights and nobody decided it. It still exists; it has to be elicited and recorded deliberately.

A caution that belongs in the object, not in a footnote. The model records *whether a position exists*, who took it, when it was taken and what document it lives in. **It does not record whether the position is correct.** Whether a firm's reading of a requirement is the right reading is a

matter for its second line, its internal audit and its reporting auditors, and it was their work before agents existed. What is in scope is narrower and factual: that a position exists, that a named person owns it, that it resolves to a producible document, that it is the position the actor was operating under, and that the implementation matches what it says. A position that is wrong is a complete record owned by somebody. A position that is missing cannot be attributed to anyone, and only the second is this model's business.

3 · Control — *versioned*

What the organisation does to discharge the obligation, as designed. Distinct from how it is implemented and from the evidence that it worked.

Field	Label	What it holds
obligation_ids[]	Obligations discharged	Which requirements this control exists to satisfy.
interpretation_id	Interpretation applied	The position the control was designed against.
objective	What the control is for	The outcome it secures, stated independently of how it is built.
kind	Control type	See permitted values below.
frequency	How often it runs	The band, not the schedule. See permitted values below.
owner_id	Control owner	Standing accountability for this control.
implemented_by[]	Implemented in	The Assets that carry it.

Permitted values for `kind`: preventive · detective · corrective

Permitted values for `frequency`: continuous · event_driven · daily · weekly · monthly · quarterly · annual · ad_hoc

The band, not the schedule. A control's precise timing is a property of the Asset implementing it, and recording a cron expression here would duplicate configuration the model does not govern. `ad_hoc` carries the sense `refresh_mode` gives `manual`: it runs when somebody remembers. Including it means an assessor can record that honestly rather than choosing a frequency the control does not have.

Separating objective from implementation lets a control survive a platform migration, and lets two organisations compare controls running on different systems.

4 · Owner — *versioned*

Standing accountability for one object, held continuously and independent of any event. The owner of a control, a data element, a system, a service.

Field	Label	What it holds
role	Role	The accountable role.
person	Person	The named individual holding it. Not a team, not a mailbox. Nullable: null means the role is recognised and nobody holds it.
smf_ref	Senior manager function	Where the role maps to a designated senior management function. Nullable.

scope	What they own	The objects this ownership covers.
delegated_from	Delegated from	Where the accountability came from.

A role with no holder is a different finding from no role at all, and until person was nullable the two were indistinguishable. One needs a name; the other needs a decision about whether the role should exist. A separate flag was rejected: vacancy is derivable from a null person, and a second place to record something derivable is what drifts.

One person holds many. A single service may have dozens of object owners — ten control owners, twenty data owners, thirty system owners is realistic for one reporting estate. That is not a defect; **the seams between owners are where failures occur**.

Ownership attaches at a grain the organisation chooses per object kind and inherits downward, with deliberate overrides for critical elements. **The inheritance rule must be written down**, or two people will resolve the same element to different owners.

5 · Authority — versioned

What lets a role act, and direct objects owned by others. Delegated authority, an agent's mandate, an SMF responsibility, the accountable executive for an important business service, a delegated reporting agreement — all the same shape.

Field	Label	What it holds
granted_to	Granted to	The actor receiving it — a person, a system or an agent.
granted_by	Granted by	The Owner conferring it, or the Asset that conferred it where no person did.
permitted_action	What may be done	One verb, drawn from the Action vocabulary at object 8.
permitted_asset_id	What it may be done to	One asset.
limits{}	Limits	Value, volume or scope constraints. Four separable facts — see below.
conditions	Conditions	Preconditions that must hold for the grant to apply.
expires	Expires	
established_by	Who determined it was needed	The person or system that decided this grantee should hold this. Often the grantee themselves, which is itself a finding.
established_from	What they relied on	The Asset relied upon — a contract, a role record, a screening report, a change ticket. Null means the grant rests on assertion alone.
approved_by	Who accepted it	Must differ from `established_by`. Self-approval is not approval. Null is a finding, not a blank.
approved_at	When it was approved	
approved_from	What the approver relied on	The Asset the approver relied upon. Distinct from <code>established_from</code> , which is what the establisher relied upon. Null means the approval rests on nothing recorded.

enforced_by	Enforced in	The Asset that enforces the grant — an IAM policy, an access gateway, or the application itself. Resolved as at <code>effective_from</code> under principle 9.
source_ref	Source document	The Asset the grant is recorded in. Must resolve to a registered Asset , not free text.

`granted\_by` resolves to an Owner or to an Asset, and which one it is, is itself the finding.

Grants are increasingly conferred by mechanisms rather than people: an orchestrator hands a calling agent a tool surface for the duration of one call, and that surface confers verbs over assets as surely as any signed delegation. Where `granted_by` resolves to an Asset, no person conferred the grant. There is nobody to ask whether it should exist, and `approved_by` will be null because there was no approval step to have.

Why not a flag saying the grant was conferred at runtime. Same reasoning as D66 and D89. A category is passive and invites a tick, where a pointer at the conferring artefact either resolves or it does not. What kind of artefact it is, is `Asset.kind` doing its job. The practical gain is that every runtime grant points at the one function that builds the surface, so the question a static export cannot answer — what can this agent reach through other agents — becomes a traversal from that Asset.

`enforced\_by` records what actually stops the action, and it is a reference rather than a category. A grant that nothing enforces is a sentence in a document. Pointing at the enforcing artefact either resolves or it does not, where a category such as *platform* or *application* is passive and invites a tick — the same reasoning that rejected a *basis* vocabulary at D66. Which kind of enforcement it is, is `Asset.kind` doing its job.

Where it is null, no enforcing artefact is recorded. Whether that is because none exists or because nobody established the answer is carried by object 11, not by the field. SILENT and UNKNOWN are not the same finding and the model never collapses them.

The case it exists for. A platform role can read an entire bucket; separation between customers lives only in application code, by project prefix. That records as **two** Authority records, not one. A platform grant — read, over the bucket, `enforced_by` the IAM policy. And the intended grant — read, over one project's prefix, `enforced_by` the application. A narrower grant enforced by an application, sitting inside a wider grant enforced by the platform, means the platform permits what the application is trusted to refuse.

That comparison is a traversal only if `resides\_in` is populated. Something has to say the prefix sits inside the bucket, and that is the containment edge added at D67. Without it the two grants look unrelated and the gap is invisible, which makes `resides_in` load-bearing here rather than a convenience.

A prompt is a source of Authority, and usually an unenforced one. Where a system prompt tells an agent it may do something, that is a grant: one Authority record per verb-asset pairing, `source_ref` the prompt Asset, `enforced_by` the model Asset. A model is not an enforcement mechanism, so an Authority whose `enforced_by` resolves to a model is on its face a grant nothing enforces. That is the finding, stated as a record rather than as an opinion about one.

The case it exists for. A constructed insurance claims agent held two system prompts. The one its agent card published forbade promising payment. The second, used for every drafted customer reply and reproduced in no governance record, told the model it could confirm the

settlement figure where the claim was covered and arrange payment to the customer's nominated bank account. No tool and no grant existed to do either. In this model that is Authority records whose `enforced_by` is the model and whose `permitted_asset_id` points at assets the agent holds no other grant over — conferred by a prompt, enforced by nothing, over something it cannot reach. The contradiction between the two prompts then falls out of the records, instead of depending on a reader noticing that the published prompt was not the only one.

Where ``enforced_by`` and ``source_ref`` resolve to the same Asset, say so. The thing doing the enforcing is then also the only record of it, and there is no independent statement of the grant anywhere. Same shape as `Evidence.independent`.

``limits`` is a structure, not a sentence. A threshold written into a grant is a copy of a value maintained somewhere else, and a copy that restates a value held elsewhere is the thing that drifts.

Field	Label	What it holds
<code>statement</code>	The constraint	The rule in words. "Transactions at or above the threshold require human approval."
<code>value</code>	Value as observed	The threshold at the moment it was recorded. Null where the constraint has no number.
<code>value_ref</code>	Where the value is maintained	The Asset holding the configured value. Must resolve to a registered Asset. Null means the number is maintained nowhere, or nobody could say where.
<code>observed_at</code>	When it was read	When <code>value</code> was taken from <code>value_ref</code> .

A limit whose value the grantee maintains is not a limit. Where `value_ref` resolves to an Asset the grantee controls or produces, the limit is recorded as unenforced. The live case: a nested-call policy that correctly defaulted to no raw data and no writes, read from the calling model's own tool arguments, so the constraint was set by the party being constrained. The same rule covers model-written code that a tool then executes — the expression is an Asset the grantee produced, and any limit resting on it rests on the grantee's own output.

This is the third appearance of one shape, alongside `Evidence.independent` and `enforced_by` resolving to the same Asset as `source_ref`: **the thing being checked is also the thing supplying the answer.** It is stated as a rule rather than added as a field because the fact is already expressible — whether the grantee controls that Asset is a traversal through ownership — and a flag would be a second place to record something derivable, which is what drifts.

The finding this makes available. A limit carrying a value and no `value_ref` is a number somebody typed into a grant and nobody maintains. It will be correct on the day it is written and silently wrong afterwards, and until now the model could record the limit without ever exposing that.

This is the same shape the exposure at default worked example reached by hand: the credit conversion factor is held in a rate table and the classification holds the class, not the rate. The model made that problem be solved twice. It now carries it.

``conditions`` is left as text for v1.0, on both Authority and Permission. The same drift applies — a condition naming "the retail book" depends on a definition that moves — but the case is less concrete than a threshold and the structure can follow if first contact with a real grant register shows it is needed. **That is the trigger: a condition whose meaning turns on a value maintained elsewhere.**

One grant per verb-asset pairing, not arrays. Earlier drafts held `permitted_actions[]` and `permitted_assets[]`, which implies a rectangle: every verb applying to every asset. Real grants are not rectangular. *Extract data from system 1; read data from system 2; approve updates below £10,000 in system 7* is three grants with three different limits, and expressing it as two arrays would imply approve rights on systems 1 and 2 that nobody conferred.

So an agent with three permissions has three Authority records. They are not a derived cross-product — each carries its own limit, its own conditions, and potentially its own grantor and expiry.

Permitted Actions are therefore never enumerated on object 8. They are already here, one row each.

``source_ref`` must resolve to a registered Asset, not to free text — the same constraint as Interpretation, and for the same reason. A permission held in a configuration file is not a recorded grant unless that file is an Asset with an owner. An Authority whose source cannot be produced is an Authority nobody conferred.

How the grant was established

Five fields, and the useful ones are the three that are frequently empty.

The model does not grade a justification. It records two facts a reader can check: who decided the grantee needed this, and what they relied upon. `established_from` resolving to nothing means the grant rests on an assertion — recorded without anyone having to judge whether the assertion was reasonable.

``approved_by`` must differ from ``established_by``. This is the same rule as `Evidence.captured_by` differing from the actor that performed the action. Self-verification is not evidence; self-approval is not approval. Where the two match, or where the approver is absent, the model says so.

``approved_from`` records the approver's homework, which the model previously left out. D66 requires that the approver and the establisher differ, and `established_from` records what the establisher relied upon. Nothing recorded what the approver relied upon, so an approval given on the strength of nothing looked identical to one given after reading the contract. Where `approved_from` and `established_from` resolve to the same Asset, the approver read only what the establisher produced. That is the fourth appearance of one shape in this model, alongside `Evidence.independent`, `enforced_by` resolving to the same Asset as `source_ref`, and a limit whose value the grantee maintains: **the thing being checked is also the thing supplying the answer.**

Why not an approval object carrying all of it. Proposed and rejected. The four flat fields work, a pointer at an artefact either resolves or it does not, and reification would move three working fields for the sake of a fourth. Same reasoning as the rejection immediately below.

Why not a vocabulary of justifications. An earlier draft proposed a `basis` field with terms like `self_asserted`, `role_derived`, `contract_derived`. It was rejected: a category describing a *reason* is passive and invites a tick, whereas a pointer at an artefact either resolves or it does not. What kind of artefact it is, is already `Asset.kind` doing its job.

A worked case. In the documented agentic espionage campaign of September 2025, a threat actor obtained capability by asserting they were a security firm conducting authorised testing. Every action that followed was within the authority the agent believed it held, so there is no `unauthorised_action` to record. The model instead records: established by the account holder; established from nothing; approved by nobody. Three fields, all true, and the finding needs no opinion attached.

This describes most self-serve access to capable tooling, including commercial AI. That is accurate rather than an accusation — general-purpose infrastructure is not usually vetted in advance, and the prevailing control model is to monitor and terminate rather than to approve.

The model's contribution is to make the fact visible, so an organisation can decide whether an unapproved, unevidenced grant is acceptable for what that agent can reach. For a summarisation tool, probably. For one holding change verbs over a crown-jewel asset, probably not.

This is the object missing wherever an agent's permitted action space is described as "not yet locked down". It is also the layer operational resilience has already built for one domain: **the accountable executive for an important business service is an Authority, not an Owner**.

6 · Asset — versioned

Anything a control attaches to or a dependency runs through. The kinds align with the resource categories operational resilience already requires organisations to map. A tradeable financial instrument is an `Asset`, or is referenced by one.

Field	Label	What it holds
<code>kind</code>	Asset type	See permitted values below.
<code>name</code>	Name	
<code>owner_id</code>	Asset owner	
<code>criticality</code>	Criticality	Importance to a business service . An availability measure. Not the same scale as <code>Exception.materiality</code> , and not the same axis as <code>sensitivity{}</code> .
<code>provider_id</code>	Provider	Where a third party supplies it.
<code>location{}</code>	Where it runs	Six separable facts — see below.
<code>sensitivity{}</code>	How sensitive it is	Six separable facts — see below.

Permitted values for `kind`, grouped to align with resilience resource categories:

Group	Terms
technology	<code>system</code> · <code>data_pipeline</code> · <code>model</code> · <code>index</code> · <code>euc</code> · <code>prompt</code>
information	<code>data_element_logical</code> · <code>data_element_physical</code> · <code>report</code> · <code>document</code>
third_party	<code>provider</code>

People are not Assets — they are Owners and actors. Facilities are not a kind; they are location.

`index` is a derived copy, and that is the point of giving it a kind. A vector store, a materialised view, a cached reference set, a replicated register. It is not the source, it was built at a moment, and nothing about it carries the version of what it was built from. An agent operating against an index built in March is operating on March's policy in November, and no property of the index says so.

`prompt` is the agentic end user computing problem. A system prompt determines what an agent will attempt. It is frequently written outside the change process, it is often unversioned, and in the first real pack examined there were two of them where every governance record described one. It sits in the technology group rather than the information group for the same reason euc does: it is an artefact that determines behaviour, not information about the business.

Recording the prompt as an Asset is only half of it. Where the text licenses or forbids an action, that licence is an Authority record pointing at the prompt as its `source_ref` — see object 5. Without that second half the model holds the artefact and says nothing about what it confers, which is the same defect as recording a configuration file and not the grants inside it.

`data\_pipeline`, not `pipeline`. In energy, transport and water a pipeline is a physical thing, and those sectors are in scope. Qualifying the term costs a word and removes a homonym that would otherwise reach a live estate. **Operational technology assets — programmable logic controllers, SCADA, industrial control systems — are not currently modelled as kinds.** They would arguably be `system`, and that is probably too coarse. Recorded as a gap rather than papered over.

`location` is a structure, not a string. This is where sovereignty is encoded:

Field	Label	What it holds
<code>data_residency</code>	Where the data sits	At rest.
<code>compute_location</code>	Where processing happens	The jurisdictions in which the asset executes, including any that only handle it transiently.
<code>key_custody</code>	Who holds the keys	The actor holding the encryption keys, not a place. A typed actor: the organisation, the provider, or a named third party.
<code>support_access</code>	Support access from	Which jurisdictions humans can reach it from.
<code>log_location</code>	Where logs are held	The jurisdictions holding the asset's logs. Often different from the data, and the difference is what breaks an exit plan.
<code>legal_jurisdiction</code>	Governing jurisdiction	Whose courts bind the provider and its parent.

Six separable facts. An organisation can satisfy the first and fail the other five, which is why "sovereignty" as a single word produces vendor answers containing the word *may*.

Five of the six are jurisdictions, and they are coded. `data_residency`, `compute_location`, `support_access`, `log_location` and `legal_jurisdiction` each

hold a list of ISO 3166-1 alpha-2 codes. `Obligation.jurisdiction` uses the same encoding.

Nothing else does. `Permission.permitted_destinations[]` holds Assets, not places: where those Assets sit is reached by traversing to their own `location{}`, because **location is recorded once and principle 7 forbids duplicating it onto another object**.

This is the one place the model standardises a value rather than recording the organisation's own. The reason is that comparing the six facts against each other **is** the purpose: "Ireland", "EU" and "eu-west-1" written into three of these fields describe a situation nobody can assess, and the disagreement the structure exists to expose never surfaces.

Three supranational codes are permitted alongside the country codes: EU · EEA · WORLDWIDE. A provider that answers "the EU" has not named a country, and recording that as twenty-seven codes would assert something the provider did not say. The model records the answer at the precision it was given, and the loss of precision is itself the finding.

`key\_custody` is the exception and is not a place at all. It names the actor holding the keys, and takes the typed actor shape used elsewhere. Where the keys sit with the same provider that holds the data and the logs, the three facts have collapsed into one party, which is the condition every exit plan assumes is false.

`sensitivity` is also a structure, not a scale. `criticality` measures availability — how much it matters if this stops working. Sensitivity measures confidentiality and integrity — how much it matters if this is seen, or changed. **They are different axes.** An archive of historical card data sits in no important business service, breaks nothing if unavailable, and is the thing an attacker most wants.

Firm-local scales are recorded, never standardised. `Asset.criticality`, `Dependency.criticality`, `Exception.materiality`, `system_tier` and `information_class` all carry a value on a scale the organisation defines. One firm runs four values, another five, and the model does not adjudicate between them, for the same reason it does not test whether an Interpretation is correct.

The value is held exactly as the organisation writes it. The scale itself is declared once per organisation in the mapping manifest, naming the registered Asset that defines it and its values in order. Because that Asset is versioned, principle 9 resolves a value recorded in March against the scale that was effective in March, even where the organisation re-tiered in June.

What this buys is interpretability, not comparability. A declared scale means a value can still be read back in two years. It does not make two organisations' scales equivalent; that would need a mapping between them, which the model does not attempt and does not claim.

And the purpose is catching an organisation contradicting itself. An asset hand-designated `crown_jewel` sitting in the lowest tier. A dependency marked critical feeding a system marked routine. Those are coherence failures, visible without anyone deciding whose scale is right, and coherence is in scope where correctness is not.

Field	Label	What it holds
<code>system_tier</code>	System tier	The organisation's own tiering, whatever scale it uses.
<code>information</code>	Information	Public, internal, confidential, and so on. The organisation's

_class	classification	scheme, not ours.
contains_pii	Contains personal data	
special_category	Special category data	Article 9 data, or the local equivalent.
regulatory_scope	Regulatory scope	PCI DSS, SOX, and similar designations.
crown_jewel	Crown jewel	A designation sitting above the others. Usually small, hand-curated, and frequently disagreeing with the tiering — which is itself a finding.

These registers already exist in most organisations and are usually better maintained than the dependency map. **The model holds what they already maintain rather than imposing a scheme**, which is design principle 8 applied.

Why this matters for reachability. "This agent can reach fourteen assets" is not a determination. *Two of the five assets outside its grant hold special category data* is one. A count becomes actionable only when the assets are graded.

7 · Dependency — *versioned*

A directed edge between two Assets. Lineage is this object. So is the dependency map, the boundary model and the concentration analysis.

Field	Label	What it holds
from_id	From	The Asset the edge runs from.
to_id	To	The Asset the edge runs to.
kind	Dependency type	See permitted values below.
criticality	Criticality	How much it matters if this dependency fails. The same availability axis as <code>Asset.criticality</code> , and on the same firm-local scale.
crosses_boundary	Crosses a boundary	Whether the edge leaves a controlled perimeter.
refresh_mode	How the copy is refreshed	For <code>derives</code> edges only. See below.
last_refreshed_observed	Last refreshed, as observed	The date the derived copy was last rebuilt, and the date that observation was made . True when recorded, and wrong a week later.
evidence_ref	Evidence	What proves the edge exists and, for <code>substitutes_for</code> , that it has been exercised.

Permitted values for `kind`:

Group	Terms
data_flow	<code>reads_from</code> · <code>writes_to</code> · <code>derives</code>
execution	<code>runs_on</code> · <code>invokes</code>
resilience	<code>backs_up</code> · <code>substitutes_for</code>
containment	<code>resides_in</code>

Most organisations hold the `data_flow` group as lineage and hold nothing in resilience.

``substitutes_for`` is an assertion until exercised.

``resides_in`` records which asset holds another, and runs from the contained asset to its container. An account sits in a table; a table sits in a system. The term is transitive and applies at whatever grain the organisation registers physical data elements, so the chain is the same term applied twice rather than a kind per level.

Two traversals depend on it. *Where does this sit* reaches a data element's residency, custody and jurisdiction by walking to the system that holds it. *What can this reach* walks the other way, down from whatever asset a grant names to everything inside it — and that is the number a CISO asks for, because entitlements are enforced at table grain while actions are recorded at column grain. Counting what an agent touched is a usage figure; counting what it could touch requires this edge.

The inheritance rule, and it must be written down. A `data_element_physical` carrying no `location{}` of its own takes the location of the asset it resides in. Where it carries its own and the two differ, that difference is recorded and is a finding: an element whose residency differs from its host means a copy exists that the dependency map is not showing. This is the same device object 4 already uses for ownership, and it is why location stays recorded once rather than copied onto every column.

Permitted values for ``refresh_mode``, on derives edges:

Term	What it means	Staleness
continuous	Rebuilds on source change	Minutes. Rare and expensive
scheduled	Nightly, weekly, monthly	Bounded, and therefore assessable
manual	Rebuilt when someone remembers	Unbounded. The honest answer is usually that nobody knows when it last ran
on_deployment	Built once, never touched	Whatever has happened since. Common where a proof of concept became production

Why this belongs on the edge rather than on either Asset. Refresh describes the *relationship* between a derived copy and its source, not a property of either end. Placing it here makes the useful question a traversal: which agents depend on an index last rebuilt before this interpretation changed?

And it generalises beyond retrieval stores. A materialised view, a nightly extract, a cached dataset and a replicated register are the same shape — a derived copy with a refresh contract that may or may not be honoured. Vector stores are the most recent and least governed example, not a special case.

8 · Action

An event: a named actor did something, to something, at a time. One object, three modes.

Field	Label	What it holds
mode	Which position this record asserts	See permitted values below.

actor_id	Who or what acted	Must resolve to a single actor — john.smith, batch-4471, agent-21094. Not a team, not a process family.
actor_type	Actor type	See permitted values below.
verb	What was done	See permitted values below.
target_asset_id	What it was done to	
destination_asset_id	Where the copy went	Required for extract, null otherwise. Resolves to an Asset, so its six location facts come with it.
retention	Whether the copy persisted	What happened to the copy after the act. A state after the act, not a different act. See permitted values below.
output_sensitivity	What the output carried	The class of what this action put into its destination, on the same scale as Asset.information_class. Null means the target asset's class stands.
authority_id	Under what authority	The grant the action was taken under. Null is a finding, not a blank.
preconditions	Preconditions	What had to hold before it could occur.
occurred_at	When it happened	

Permitted values for `retention`: discarded · held_for_task · persisted

Permitted values for `mode`: permitted · expected · performed

The three modes are the same act seen three ways, and they are the three positions an assessment reports on.

Mode	What the record asserts	How it is obtained
permitted	The agent may do this	Read from the grants, the configuration and the code. A live access policy is the configuration rather than a record describing it, so this is a fact about the system and not a claim about one
expected	The person accountable for this says the agent should do this	Asked, as a concrete scenario, and recorded in their own words
performed	The agent did this	Observed by running it

Whose expectation it is, without a new field. actor_id cannot carry it, because on an expected Action the actor is the agent. The position is a Decision: kind is set_expectation, decided_by the accountable person, subject_ref the grant or scenario, rationale their words, decided_at the date. The expected Action's derived_from resolves to that Decision. A position somebody took is an event with a person and a time attached, which is what object 10 is for.

`expected` is the only one of the three that comes from a person. The other two are read or observed. That is why an assessment's irreducible human input sits in exactly one place.

Three gaps follow from the three modes.

Gap	What it measures	What it needs
expected against permitted	Is what the agent may do what the accountable person wants	The grants, and a position
performed against permitted	Did the agent stay inside its grants	The grants, and a run
expected against performed	Did the agent do what the accountable person wanted	All three

The third is the question most often asked of an agent and it cannot be answered from any two of the three modes.

Permitted values for `actor\_type`: human · system · agent

Permitted values for `verb`, grouped by consequence:

Group	Terms	What separates it from the group above
read	access · read · infer	—
extract	extract	A copy leaves the organisation's control
propose	recommend · draft	—
change	update · submit	A regulated outcome changes
gate	approve · block · escalate	Authority is exercised

`output\_sensitivity` exists because a tool can reduce what it carries, and can be wrong about it. Without it, sensitivity is traversed from the target Asset, so every tool touching a raw store reports raw. That over-reports a genuine aggregate and — worse — cannot distinguish it from a tool that claims to be one and returns rows.

The comparison is the finding, and `mode` already carries it. The declared surface of a tool is an Action with mode: permitted; what it actually returns is mode: performed. Where the permitted record claims a lower class than the performed record carries, the tool is mislabelled and every control keyed on that label is being told the wrong thing. In the live case four separate policy layers — a dispatch gate, taint stamping, nested-call envelopes and an egress audit — all read one tag, and two of eight tools carried the wrong one.

Null is deny-by-default, not unknown. An untagged tool is a tool nobody has audited, not a safe one, so the target asset's class stands until someone states otherwise. Where nobody has established the answer, that is object 11's job and the evidence scale's UNKNOWN, as everywhere else.

Rejected: a controlled vocabulary of output classes. The live case used four — raw, masked, aggregate, meta — and it is a good scale. It is also one organisation's. D81 settled that a firm-local scale is held as the organisation writes it, declared on the manifest, interpretable rather than comparable. Importing these four would make them look standard and would be wrong the first time an organisation with five classes met them.

Three states of the same data, and the distinction is where the copy comes to rest.

Term	Definition
------	------------

access	Data was consulted and nothing was retained. A query returning a count; a lookup that produced a decision. No copy anywhere.
read	Data entered the actor's context or memory. A copy now exists <i>inside the actor</i> , for at least the duration of the task. Where that context runs is a property of the actor's Asset.
extract	A copy was written somewhere durable <i>outside both the source and the actor</i> . A file, a store, a training set, a development environment.

Why this is not over-specification. An organisation that cannot say whether its agent reads data into a model hosted in another jurisdiction cannot answer a question its regulator will ask — a third-party dependency with unknown jurisdiction under DORA, a transfer nobody assessed under data protection law. Collapsing the three into one verb to keep the vocabulary small would mean the model cannot express one of the largest risks it exists to record. **The finding is usually the not-knowing rather than the destination**, and the evidence scale carries UNKNOWN for exactly that.

Vocabulary note. `retrieve` was withdrawn. It means to fetch and bring back, which is movement — the opposite of what it was being used for, and unmappable against `extract` sitting beside it.

A caution worth stating. Whether a value entered a context window is an implementation fact, and an organisation assessing its own agent may not know. The distinction is right and frequently unanswerable. That is a reason for UNKNOWN, not a reason to collapse the terms.

Two boundaries now, not one. Between `read` and `extract` is where data leaves the organisation's control — the data owner's and the DPO's question. Between `propose` and `change` is where autonomy becomes consequential — the CISO's and the regulator's. Different risks, different owners, both visible in one vocabulary.

The boundary between `propose` and `change` is where autonomy becomes consequential. An agent permitted only `read` and `propose` cannot alter a regulated outcome. One permitted `change` can. One permitted gate is exercising authority.

The two modes are the point: `permitted` is the AI action space and change control; `performed` is the runtime record. Same object, so "was this action permitted?" is a join, not a project.

Agents require durable identities that survive redeployment, in the same identity space as people and systems. Without them the action log is a pile of unattributable events.

Action sits deliberately outside the accountability chain. **Actions can be delegated to machines; accountability cannot.**

9 · Exception

A divergence between expected and actual. This is "break", defined once.

Field	Label	What it holds
kind	Break type	See permitted values below.
control_id	Control affected	
asset_ids[]	Assets affected	
detected_by	Detected by	

detected_at	When detected	
occurred_at	When it happened	The interval between these two is what a supervisor asks about.
materiality	Materiality	Regulatory consequence. Not the same scale as <code>Asset.criticality</code> .
recurrence_key	Repeat identifier	What makes this the same break as a previous one. Without it, the third occurrence looks like the first.
root_cause_ref	Root cause	May reference an external classification — an ATT&CK technique, an OWASP category — rather than a term owned here.
status	Status	See permitted values below. Defaults to `recorded`.
remediation_id	Remediation	

Permitted values for `kind`, grouped by what failed, because each group has a different remediation owner:

Group	Term	Definition
specification	interpretation_mismatch	Implemented logic does not match the recorded interpretation
	config_defect	Configuration does not implement the intended rule
	transformation_defect	Data correctly sourced, incorrectly transformed
data	data_defect	An input value is wrong, missing, stale or duplicated
	population_drift	No individual record is wrong; the distribution has moved beyond tolerance
outcome	rejection	The receiving party rejected the submission
	late	Produced or submitted outside the required window
control	unauthorised_action	An action occurred without, or beyond, a valid Authority
	dependency_failure	A depended-upon asset became unavailable or unreliable
	boundary_violation	Execution, data or access crossed a boundary it was not permitted to cross

Vendors keep their own detection logic and declare a mapping into this list. Nobody gives up their taxonomy.

`population_drift` cannot be detected by record-level testing, and most issue registers cannot currently represent it.

Permitted values for `status`. Every value other than `recorded` requires something that resolves, and where that thing does not resolve the status is `recorded`.

Status	What it means	What it requires
recorded	The divergence is recorded and nobody has stated a position on it	Nothing. The default
accepted	A named person has accepted the divergence as it stands	An Owner, a date, and an expiry or an explicit statement that there is none

remediating	A remediation is under way	remediation_id resolving
withdrawn	On examination there was no divergence	What established that, as a pointer
closed	The divergence no longer holds	What established that, as a pointer, and a date

`recorded` is the default and requires nothing, which is the point of it. It is the commonest real state and the one every register in use skips. A vocabulary of open, in progress, accepted and closed forces a divergence into a bucket implying somebody is dealing with it. `recorded` says the true thing: it is written down and nobody has decided anything. A dated gap list with no owner is a record that the organisation knew, and this status says so without anyone having to write the sentence.

The other four require something that resolves, for the reason a `basis` vocabulary was rejected at object 5: a category invites a tick, where a pointer either resolves or it does not. `accepted` is the one that matters, because an acceptance with no named person and no date is the commonest way a divergence stops being tracked.

OSCAL's three dispositions are a mapping target, not a replacement. A risk adjustment and an operationally required risk both map to `accepted`, differing in the reason rather than the disposition, and a false positive maps to `withdrawn`. OSCAL has no term for the unexplained case. We point at their vocabulary on the same footing as `root_cause_ref` pointing at an ATT&CK technique, and we do not adopt it. See D112 and D114.

10 · Decision

A judgement made about an Exception, an Action, a Control or an Interpretation.

Field	Label	What it holds
subject_ref	What was decided about	
kind	Decision type	See permitted values below.
decided_by	Decided by	Who or what exercised the judgement — a person, a system or an agent.
answerable_for	Accountable for this decision	The person who answers for it. Always a person.
authority_id	Under what authority	
rationale	Reason for the decision	
evidence_refs[]	Evidence relied on	
decided_at	When decided	

Permitted values for `kind`:

Group	Terms
disposition	approve · reject
deferral	escalate · accept_risk
intervention	remediate · suspend · restart
standard	set_expectation

``set_expectation`` is how a position about what should happen is recorded, and it is the only value in this vocabulary that is not a disposition on something that already exists. Added at D118 because an Action in `expected` mode has to carry whose expectation it is, and `actor_id` cannot: on such a record the actor is the agent. So the position is a Decision, `decided_by` the accountable person, `subject_ref` the grant or the scenario, `rationale` their words, and the expected Action resolves to it through `derived_from`.

Two fields for people, deliberately. Usually `decided_by` and `answerable_for` are the same. When they diverge, that divergence is the finding: an agent decided, and the person answerable is whoever granted its Authority. Where nobody can be named, the model records that fact rather than refusing the row.

`answerable_for` is an Owner reference, **resolved as at ``decided_at``** under principle 9. It is point-in-time and event-scoped in a way that object 4 read as a whole is not: object 4 is standing accountability for a thing, and what answers for a decision is the Owner row that was effective when the decision was taken. Owners change; the dates on the object are what make this work. The person answerable is normally the Owner of the Control it was taken under, or their delegate — **and where it is neither, that is a governance gap the model should surface.**

11 · Evidence

A record asserting that something was true at a point in time. The manifest is a query over this object, not a document someone writes.

Field	Label	What it holds
<code>subject_ref</code>	What it evidences	
<code>kind</code>	Evidence type	See permitted values below.
<code>captured_at</code>	When captured	
<code>captured_by</code>	Captured by	Must differ from the actor that performed the Action. Self-verification is not evidence.
<code>retention_until</code>	Retain until	
<code>storage_location</code>	Where held	
<code>independent</code>	Held independently	Whether this evidence sits outside the system it evidences.
<code>integrity_hash</code>	Integrity check	

Permitted values for ``kind``, grouped by the question the evidence answers:

Group	Question	Terms
assurance	Was it tested?	<code>test_result</code> · <code>attestation</code>
authorisation	Was it allowed?	<code>approval</code> · <code>receipt</code>
occurrence	Did it happen?	<code>log</code> · <code>submission</code> · <code>report</code>
continuity	Does it survive?	<code>backup</code>

`independent` is the most consequential field in the model. Exit plans fail when the logs, the backups and the failed provider are the same entity — and that failure is invisible unless someone has written down a flag that says so.

12 · Permission — *versioned*

The position a data owner takes on an asset being used for a purpose. An agent may hold a valid Authority to update case records and still have no permission to move that data to a new location, or expose it to a third party.

Field	Label	What it holds
id	Permission ID	Stable, meaningless.
asset_ids[]	What it covers	The assets this position applies to.
granted_by	Granted by	The Owner of the asset. Not the owner of the agent.
purpose	Purpose	Why the data may be used. The field the model previously could not express.
permitted_destinations[]	Where it may go	The Assets the data may reach — systems, providers or stores. Jurisdictions are not held here; traverse to each destination's <code>location{}</code> , per principle 7.
onward_disclosure	Onward disclosure	Whether the data may leave the organisation at all. Which third parties is answered by <code>permitted_destinations[]</code> pointing at provider Assets.
conditions	Conditions	Constraints attached to the position.
expires	Expires	
assessment_refs[]	Assessments relied on	The privacy impact assessment and any equivalent, as Evidence.
retention_ref	Retention position	Where the retention decision is recorded.
source_ref	Source document	The agreement itself. Must resolve to a registered Asset , not free text.

Why this is an object and not fields on Authority. The cardinality settles it. One position may cover several agents; one agent may operate under several positions. That is many-to-many, and it cannot sit as attributes on either side without duplicating the same assessment reference into every grant that touches the data.

And why not an Interpretation. An Interpretation is a position about a *rule*. This is a position about an *asset*. It would need expiry, scope and purpose bolted on — three fields to make one object do two jobs.

Authority is granted to an actor. Permission is granted over an asset. Actor-side and asset-side. That distinction is the whole object.

Purpose is the point. Object 5 records what may be done and to what. It records nothing about why, and purpose is the entire basis of a data sharing position — data collected for one purpose being used for another is precisely what a privacy impact assessment exists to assess.

A caution that belongs in the object, not in a footnote. The model records *whether a position exists*, who took it, when it expires and what evidence supports it. **It does not record whether**

the position is correct. Whether a transfer is lawful is a legal judgement and stays with the organisation. Any implementation describing this as an assurance of lawfulness is misusing it.

Vocabulary note: "permission" also has an access-control sense — the thing RBAC grants. This model uses it only in the data sense. See the glossary.

Layer 2 — Where the controlled vocabularies live

Nine fields across the model are constrained to fixed sets of values. Every one is stated with its object above; this is the index, not a second source of truth.

Field	Terms	Grouped	Definitions written
Exception.kind	10	Yes — 4 groups	Yes
Asset.kind	11	Yes — 3 groups	Not yet
Action.verb	11	Yes — 5 groups	Yes
Dependency.kind	8	Yes — 4 groups	Not yet
Dependency.refresh_mode	4	Flat	Yes
Evidence.kind	8	Yes — 4 groups	Not yet
Decision.kind	7	Yes — 3 groups	Not yet
Control.kind	3	Flat	Not yet
Action.mode	2	Flat	Not yet
Action.actor_type	3	Flat	Not yet

Hierarchy is an assertion, not an identifier. Each term has a flat, stable identifier; group membership is stated separately and may change without breaking the identifier. This follows from design principle 2, and is why the group name never appears in the term. Groups exist so an organisation can report at the upper level and detect at the lower one.

Faults are not here. The simulation engine injects faults; that vocabulary constrains the engine, not these objects, and lives with the engine specification. A fault is something applied; an Exception is a divergence that occurred.

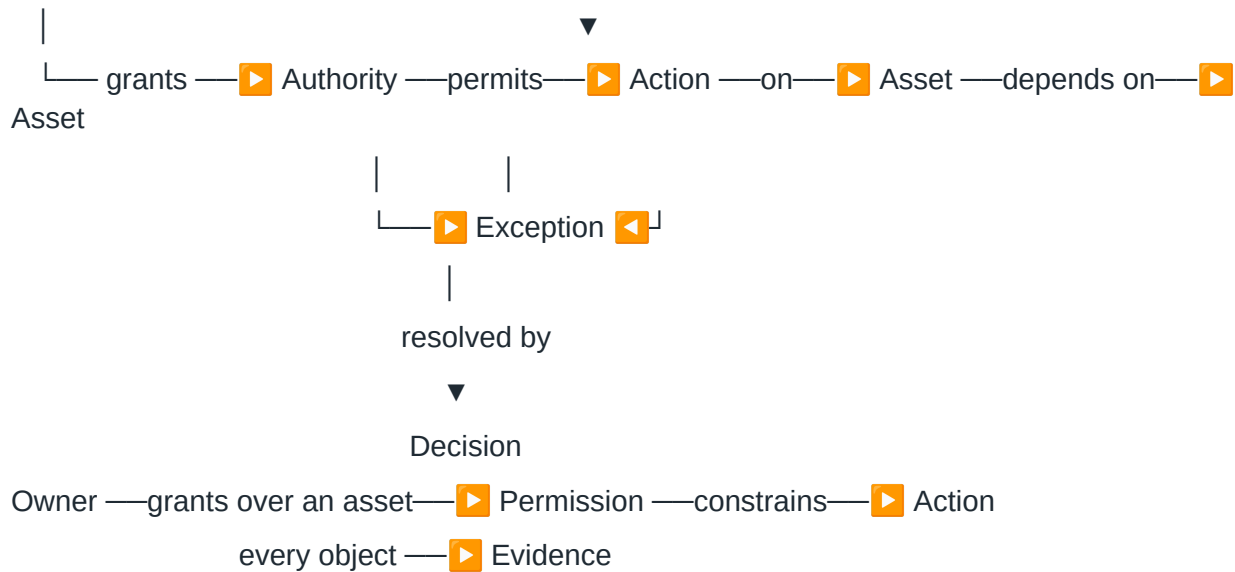
Definitions are the outstanding work. A term without a definition is an invitation to guess, and "vendors declare their mapping" is worthless if they are mapping into words whose meaning is assumed. Every term in every controlled vocabulary needs an owned, dated definition, and most do not have one. **The count is not stated here on purpose.** It has been wrong twice, it changes whenever a vocabulary is declared, and a number maintained by hand beside a list maintained by generation is the same defect this model exists to find. The generated vocabulary file is the count.

The spine

Obligation —interpreted by—  Interpretation —implemented as—  Control

|

Owner —accountable for—  Control, Asset | operates on



The accountability chain is 4 → 5 → 10. Owner holds standing accountability; Authority delegates; Decision answers. Object 8 is participation and sits outside it by design.

The evidence scale

State	Meaning
COMPLETE	Exists, is current, can be produced on request, and resolves to a maintained source outside the subject
PARTIAL	Exists but incomplete, unlinked, or not retrievable under pressure
SILENT	Checked, and the records say nothing
UNKNOWN	Nobody has established whether it exists
NOT ENGAGED	This object plays no part in this instance — a reason is required

SILENT and UNKNOWN must not be collapsed. One is a gap you can cost; the other is a gap you cannot yet size. The two words are deliberately unlike, because the pair they replaced were not: a reader skimming a table read *absent* and *unknown* as near synonyms, and the distinction they carry is the most consequential one on the scale.

The scale is a ladder and the words have to read like one. COMPLETE, PARTIAL and SILENT are a progression a reader follows without consulting a legend, which is the test a control vocabulary has to pass in front of a board. COMPLETE is measured against what this model expects for that object, which is why any report using the scale states that expectation beside the score rather than leaving it to the reader. It is not a statement that the record is correct: the model records whether a position exists, never whether it is right, and that holds here as everywhere else.

COMPLETE requires a derivation that resolves outward, and that is new at v0.9.6. A record carried inside the subject, with nothing it resolves to, scores at most PARTIAL, and the finding names it as carried rather than referenced. Without that condition the top of the scale was available to an obligation, an interpretation or an approval typed into the thing being assessed:

the record existed, was current and could be produced, so it scored COMPLETE, and the highest score on the scale was reachable by the architecture this model argues against. The condition is not a new idea here, only a general one. This specification already applies the same test to a single field, where a limit carrying a value and no `value_ref` is recorded as a number nobody maintains.

A carried record is not SILENT. SILENT means the records say nothing. A carried record says something, in the subject's own words, with no owner and no review date, and scoring it SILENT would lose the finding altogether.

NOT ENGAGED exists so that every table can carry every object without padding — a row absent from a table tells the reader nothing, and is how uncomfortable rows get quietly dropped.

The reference states

The evidence scale above records the condition of a record. This scale records something different and more specific: the relationship between a subject and an inventory. It is the scale an assessment of an agent reports against, and it is computed rather than judged, from `derived_from`, from `source_ref`, and from whether a reference resolves to an entry for the identifier cited.

State	Meaning	Reason required
Connected	The reference resolves to a register holding an entry for the identifier cited; that register governs; it has an owner and a review date; and the entry was not authored by the subject	No
Dangling	The reference resolves, and resolution buys nothing	Yes: <code>unowned</code> , <code>self_authored</code> or <code>contested</code>
Broken	The reference does not land. Either no register exists, or the register exists and holds no entry for the identifier cited	Yes: <code>no_register</code> or <code>no_entry</code>
Duplicated	The subject carries a copy and a record also exists outside it	Yes: <code>separate</code> or <code>generated</code>
Embedded	The subject carries the only record of it anywhere	No
Disconnected	Nothing in the subject, and the firm holds a record the subject cannot reach	No
Absent	No record anywhere	No

Where a subject both references outward and carries a copy, the state is `Duplicated` and it is settled from the artefacts alone, because two records exist and can be compared without asking anybody. The reason then does the work: `separate` where the external record was not made from the subject's copy, and `generated` where it was, which means there is one record and not two however many places it appears.

`Dangling / self\_authored` is the first consistency rule at D115 expressed as a state. The reference resolves, and it resolves to something the subject wrote about itself. That is not an unmaintained record belonging to somebody else; it is not an independent record at all.

`Dangling / contested` is the operational consequence of multiplicity. Several registers, none designated. The target may be owned, reviewed and the wrong one, so the reference cannot be scored any higher however well tended the thing it points at.

`Broken` is the state that cannot be seen without opening the register, and it is the hardest finding on the scale. A subject citing an identifier that no register holds is governed by something that does not exist. It is why reading a supplied register and looking up every identifier the subject cites is part of the work rather than an extension of it.

How the states score onto the evidence scale. Two scales, stated together so they are never conflated.

State	Evidence state
Connected	COMPLETE
Dangling, any reason	PARTIAL
Broken, any reason	SILENT
Duplicated, either reason	PARTIAL
Embedded	PARTIAL
Disconnected	PARTIAL
Absent	SILENT
Any state, unanswerable	UNKNOWN

`Broken` and `Absent` both score **SILENT** and are different findings, which is the clearest demonstration of why both scales exist. Under **Absent** nobody believes a record exists. Under **Broken** the subject is referencing one and the records say nothing. The evidence scale records the condition of the evidence; the reference state records what is actually happening.

The gap scale

The evidence scale grades a record. The gap scale grades **a statement an organisation makes about what its agent does**, against what the configuration shows. It is the instrument the tier 2 assessment reports in, and until v0.9.3 it was applied without being written down anywhere.

Grade	Meaning
1	It did what you said.
2	It did something you would accept, but not what the record says.
3	The records establish it would do something else — a reason is required .
4	The records cannot establish the answer either way — a reason is required .

Grade 4 reason codes. `not_declared` — no record addresses it. `not_recorded` — a record exists but nothing captures the event. `not_accessed` — the artefact that would settle it exists and was not provided.

Grade 3 reason codes, new at D97.

Code	What it means	Who closes it
<code>declaration_</code>	The record states something the configuration contradicts, and	The record's owner

wrong	the configuration is what the organisation intends	
declaration_incomplete	The record understates what the configuration permits or does	The record's owner
declaration_missing	The configuration does something no record mentions	Whoever owns the governance process
implementation_diverges	The record states the intent correctly and the configuration does otherwise	Whoever owns the system

The code attaches to the assessed statement and its source, not to the finding. One finding routinely produces several statements carrying different codes. A summary document describing a grant more narrowly than the policy enforces it is `declaration_wrong`; the declaration that summary was supposedly generated from, being narrower than the enforced policy, is `implementation_diverges`. Both were true of one wildcard, and they are closed by two different people.

The scale is not a score and does not aggregate. Counting grades across statements of unequal consequence produces a number that moves when the assessor chooses how finely to split a statement. Severity is recorded separately, per finding, and belongs to the organisation that owns the consequence.

What the scale deliberately does not carry. Whether a control is *durable* — whether a limit sits in a platform grant or in application code that changes without an approval board hearing about it — is not a grade. It is two Authority records with different `enforced_by` values, per D89, and the comparison between them is the finding. Pushing it into the grade was tried on a tier 2 dry run and mis-graded four statements out of forty-three, every one of them in the same direction. See D100.

Replay, disambiguated

Sense of "replay"	Query
Report replay	Evidence where <code>subject_ref</code> = the submission Action, plus the Obligation and Interpretation versions in force at <code>occurred_at</code>
Decision replay	Decision + its <code>evidence_refs</code> , plus the Control and Interpretation in force at <code>decided_at</code>
Agent action replay	Ordered Actions where <code>mode</code> = <code>performed</code> , joined to the Authority in force, plus any preceding Decisions
Recovery replay	Dependency traversal from the affected Asset, filtered to Evidence where <code>independent</code> = <code>true</code>

Four evidence sets, four retention regimes, one object model. An organisation that can run the first query and not the fourth now knows that about itself, which most do not.

Agent action replay is the one nobody can currently run, because it requires an actor identity that survives redeployment and an authority recorded in a form an action can be checked against. Neither exists in most estates. That absence is the finding, not a gap in the model.

Ten outputs, one model

Output	View over the model
Exception map	Exceptions grouped by Control × Asset × recurrence_key
Ownership model	Owner ↔ Control ↔ Asset, with Authority across the seams
Replay evidence pack	Evidence set for a subject_ref
Permitted action space	Actions where mode = permitted, grouped by Authority
Oversight model	Decisions by answerable_for, with the Actions they gate
Dependency map	Dependency traversal from Assets of criticality = important business service
Boundary model	Asset location{} plus Dependencies where crosses_boundary = true
Exit and recovery playbook	substitutes_for edges, plus Evidence where independent = true
Investment case	Exceptions and unmet Controls joined to Obligation effective_from — what is not covered, and by when it must be
Expected-outcome narrative	For a named Asset: the Authorities that reach it, the Dependencies and Actions that carry its contents onward, the Controls that would detect, the Exception route that would notify, the Evidence that would survive, and the Owner answerable. Each clause tagged with whether the records establish it, say nothing, or have not settled it

Ten outputs, one model. Each is a projection of a traversal, which is why they can be laid side by side rather than reconciled after the fact.

The tenth is the one people ask for in those words, and it is the only one that answers a question about something that has not happened. It predicts nothing about whether an attempt would succeed, which is a red team's work. It states what the organisation's own records say would happen next, which is usually what determines the consequence and which nobody can currently obtain except by asking people what they believe. Because every clause resolves to a record that exists or does not, the narrative carries the evidence scale on its face rather than in a footnote.

Two of these are agent-specific — the permitted action space and the oversight model. The other eight are not, and would be as useful in an organisation with no agents at all. That is the general-model, specific-problem shape described above.

Minimum viable version

If a table can only capture four things per object, capture these: **identifier**, **owner**, **effective_from**, and **the link to the next object in the spine**. Everything else can be added later. Those four cannot, because retrofitting identity across systems that already disagree is the expensive version of this work.

Annex A — Glossary

A glossary entry is warranted where two participants demonstrably use the same word for different things — **not** because a term appears in the model. Entries carry an owner and a date, and are revised, not overwritten.

Distinct from the controlled vocabularies at Layer 2: a vocabulary constrains *permitted values* for a field; a glossary disambiguates *words people already use* in conversation.

Term	Why it is contested
break / exception	Six definitions in use across the vendor set alone. Resolved by the Exception vocabulary at object 9.
material	Regulatory consequence, financial threshold, or client impact — used interchangeably
critical	Importance to a service, to the organisation, or designated status under a regime
control	An objective, an implementation, or a test result — used for all three
owner	Standing object accountability, service accountability, or whoever answered last time
replay	Four distinct operations. Resolved above.
evidence	A record, a pack assembled after the fact, or an assertion by someone senior
boundary	Legal, contractual, technical or organisational — rarely stated which
instrument	A tradeable financial instrument, or a legal instrument. This model uses it only in the financial sense.
permission	An access-control grant, or a data owner's position on use. This model uses it only in the second sense — the first is <code>Authority.permitted_actions</code> .
sandbox	A non-production environment for testing, or an isolated execution boundary around production code. Security sense here. The simulation engine is neither; it is a constructed environment.
blast radius	What an agent can reach if compromised, or what breaks downstream if it fails. Two different traversals — outward from the agent, and downstream to dependents. State which.
fault	Something the engine injects. Not an Exception, which is a divergence that occurred.
read	Colloquially, any consultation of data. In this model, specifically that a copy entered the actor's context or memory. Consultation leaving no copy is <code>access</code> . Pulling data into a hosted model's context is <code>read</code> , whatever it is called locally.
retrieval / RAG	An architecture, not an object. The store is an <code>Asset</code> of kind <code>index</code> ; what it was built from is a <code>derives</code> <code>Dependency</code> ; how current it is, is <code>refresh_mode</code> on that edge. The model is queried, never retrieved — a determination must be reproducible, and a retriever returning different chunks on different days is not.
model	This specification, or a machine learning model. The document means itself throughout , and <code>Asset.kind</code> carries <code>model</code> for the second sense. Say <i>the object model</i> for the first and <i>language model</i> or <i>the model endpoint</i> for the second. The entry exists because two participants used the one word for both inside a single conversation, which is this glossary's own test.
pipeline	A data pipeline, or a physical pipeline carrying oil, gas or water. The model uses `data_pipeline` so the question does not arise.
criticality vs sensitivity	Availability versus confidentiality and integrity. Different axes, both on <code>Asset</code> , frequently conflated.

Not the same scale, and frequently assumed to be: `Asset.criticality` measures importance to a business service; `Exception.materiality` measures regulatory consequence. A low-criticality asset can produce a highly material break.

Annex B — Extracts and mapping

Context comes from the container, and the container changes at each layer. In the object model the container is the object; in a database it is the table; in a flat file it is the filename.

Default: one file per object. `obligation.csv` with a column `source_ref`. Relationships get their own files.

Where a single wide file is genuinely needed — a vendor submission, a hand-off — qualify at the point of flattening and mechanically: `obligation_source_ref`, `exception_kind`. Generated by rule, never hand-typed, and reversible. **Qualification is a property of the projection, not of the definition.**

Every extract carries as-at metadata: `extracted_at`, `as_at`, `source_system`, `source_version`, `row_count`. A flat file is a snapshot and the model is time-aware. If you cannot say what moment the snapshot represents, you cannot reconstruct state as at a decision, and replay is gone. Most commonly skipped item in extract design; most reliably destroys the evidence chain.

Identity. Every extracted row carries `source_system` + `source_id` as a composite natural key, resolved to the canonical `id` by a cross-reference table. Extracts never mint canonical identifiers. Without this, `recurrence_key` is unimplementable.

Note which identifiers are industry-issued and which are internal. LEI, ISIN, UPI and UTI join across organisations. Obligations, interpretations, controls and authorities generally have internal identifiers or none, and must be minted before anything can be joined at all.

Mapping manifest. One row per source column: `source_system`, `source_file`, `source_column`, `target_object`, `target_field`, `transform`, `mapped_by`, `mapped_at`. Value-level mapping is a separate table and equally necessary — a vendor's break type 12 becoming `exception/transformation_defect` is a decision someone made. **The manifest is evidence, not plumbing.** When someone asks in 2028 why a break was classified as it was, the answer is in that file.

Three flat-file traps

- **Nulls.** A CSV cannot distinguish "not applicable", "unknown" and "not extracted". For `applicability` and especially `independent`, a blank is not false — reading it as false silently marks evidence as dependent when nobody has checked.
- **Arrays.** No flat-file representation. Use a child file, not a delimited field — delimiters get into the data eventually and the failure is silent.
- **Enumerations.** Files carry the canonical term, never the display label.

Annex C — Semantics

Scope, stated narrowly. Semantics here means *identity and declared meaning*: making it unambiguous and machine-checkable that one party's term and another's are the same term. It

does **not** mean inference — no class reasoning, no subsumption, no derivation of new facts. That is a research programme and nothing here requires it.

The constraint that keeps this affordable: the semantic layer must be addable **without changing the model**. A namespace, published definitions and a published schema achieve that. Anything requiring a triple store does not, and would break the claim that the minimum version runs in a spreadsheet.

D1 • Namespace and resolvable identifiers. Terms are not strings. Each resolves to a published definition under a stable base, versioned:

```
\
{base}/{version}/object/obligation
{base}/{version}/field/obligation.source_ref
{base}/{version}/term/exception/transformation_defect
{base}/{version}/group/exception/specification
\
```

Terms are flat within their enumeration. Group membership is asserted separately, never encoded in the path.

The base is declared. For this version it is `https://ceom.aiscale.ltd`, a namespace held by the publisher. So the Obligation object is identified as `https://ceom.aiscale.ltd/0.9.10/object/obligation` and its source field as `https://ceom.aiscale.ltd/0.9.10/field/obligation.source_ref`. **Who holds the base after v1.0 is a governance question and is still open**, recorded at open question 10.

D2 • Published definitions. Every enumerated term carries a definition, an owner and a date. Without these, "vendors declare their mapping" is worthless: they would be mapping into terms that mean whatever the reader assumes. The vocabularies at Layer 2 are this content.

D3 • A machine-readable serialisation. Objects, fields, types, cardinality, relationships and enumerations, published and versioned. **JSON Schema is sufficient.** JSON-LD if linked-data compatibility is wanted later — additive, need not be decided now.

D4 • The mapping manifest as the assertion layer. Annex B's manifest is the semantic device that does the work. It states that this vendor's column means this field, and this vendor's value means this term, **asserted by a named person on a date**. That provenance makes a mapping evidence rather than configuration, and allows it to be challenged.

Grounding. Obligations are abstract; data elements are concrete. What connects them is a business glossary with bindings — term, definition, owner, bound to logical and physical elements. Firms running one have most of layer 4 for their own estate; what they lack is the shared namespace making it joinable to anyone else's.

Open questions

Settled provisionally, and held that way until data contradicts them. The convention is to decide immediately, record the reasoning, and revise knowingly rather than drift. The decision log carries the reasoning.

1. Approval of an Interpretation — a Decision referencing it, or a workflow state on it? *Provisional: a Decision, so approval evidence lives in one place.*
2. Materiality — a common 1–5 scale with per-regime mapping, or regime-native throughout? *Settled at D81: regime-native. `Exception.materiality` holds the organisation's own value on its own scale, declared once in the manifest. A common scale was the earlier provisional answer and was reversed, because standardising it would adjudicate between organisations on a matter the model has no standing to decide.*
3. Does `location` belong on Asset or Dependency? *Provisional: both. An Asset has a home; a Dependency crosses.*
4. Provider concentration — derived over Dependency, or its own object? *Provisional: derived.*
5. Whether `crown_jewel` is derived from the other sensitivity facts or independently maintained. *Provisional: independently maintained — and where it disagrees with the tiering, that disagreement is itself a finding.*

Genuinely open, and best answered by contact with a real estate:

6. Retention — derived from the Obligation or from organisational policy? These conflict, particularly for agent action logs, where the obligation may be silent and the volume makes indefinite retention unaffordable.
7. Whether organisations' regulatory inventories cite the rule provision, the publication, or both — which determines whether `introduced_by` reflects practice or aspiration.
8. Whether a Permission is per agent, per asset, or per pairing of the two.
9. Whether the sensitivity facts as named match what organisations actually maintain, or whether the structure needs different fields.

Open, and answered only by adoption:

10. Who holds the namespace in the long run, and who owns revision of the controlled vocabularies after v1.0. *The base is declared at Annex C D1 as `https://ceom.aiscale.ltd`. What remains open is custody: a namespace is a claim of authority over the names inside it, and if this model is ever taken on by a standards body the base either moves, breaking every reference already made to it, or a published standard stays identified by a private company. Neither is settled here.*
11. **Whether the Exception vocabulary survives contact with the parties who must map into it.** This is the one that matters. Ten terms, each of which somebody's existing classification must fold into without loss.

Aiscale Ltd, September 2026. Deposited for open publication, 27 October 2026. Circulated for challenge, not adoption.

Since first drafted, the model has been instantiated against two real agent assessments, which produced the gap scale reason codes, the reference states and the third position on `Action.mode`. **The worked instances in this document remain constructed**, and deliberately so: a published example drawn from a client estate would carry that estate into

every reader's hands. What has not yet been done is a second assessor reaching the same determinations independently, which is the measurement this model still lacks.

Annex D — Conformance

A specification that cannot be tested can only be read. This annex states what it means for a set of records to conform, in a form a program can check. The reference validator implements every rule below and each rule carries the sentence from this document that it enforces.

D.1 The two kinds of failure, which must never be mixed

A defect means the records are malformed. The data does not express anything yet, so nothing can be concluded from it. A defect is fixed by correcting the record.

****A finding means the records are well formed and say something about the subject.**** A null approver is not a broken record. It is a statement that nobody approved the grant, and it is reportable as such.

Conformance is the absence of defects. It is not the absence of findings.

An estate that produces no findings has either been described incompletely or is in better condition than any estate observed so far.

D.2 Structural rules — a failure here is a defect

#	Rule	The sentence it enforces
C1	Every field present on a record is a field this model defines	Two spellings of one field make a count come out wrong and nothing else catches it
C2	Same-object references form a tree, never a cycle. A self-reference is a cycle of length one	Principle 10. Loading rejects a cycle rather than discovering it on the first report that never returns
C3	Every declared reference field names a record that exists	A reference that resolves to nothing is not a reference
C4	<code>source_ref</code> on Obligation, Interpretation, Authority and Permission resolves to a registered Asset , not free text	A filename someone remembers is not a source
C5	<code>Authority.value_ref</code> resolves to a registered Asset	A limit whose location nobody can state is not a limit
C6	<code>destination_asset_id</code> is present for the <code>extract</code> verb and null for every other verb	A copy leaves the organisation's control, so where it went is part of the act
C7	A reference to a versioned object resolves as at the referring record's own event time	Principle 9. Without a resolving date two implementers reach different rows and both believe they are right
C8	Every <code>Owner</code> record names a role	A role with nobody in it is a reportable state. A missing role is a defect

D.3 Independence rules — a failure here is a finding, not a defect

These are the rules that carry the model's purpose. Each one is satisfiable by a well formed record and each failure is a statement about the subject.

#	Rule	The sentence it enforces
C9	<code>Authority.approved_by</code> differs from <code>Authority.established_by</code>	Self-approval is not approval. Null is a finding, not a blank
C10	<code>Evidence.captured_by</code> differs from the actor that performed the Action	Self-verification is not evidence
C11	<code>derived_from</code> resolves outside the subject before COMPLETE may be claimed	A record cannot evidence the thing it was derived from
C12	<code>Control.enforced_by</code> is not the Asset the control governs	A control enforced by what it governs carries no independent statement
C13	<code>Action.authority_id</code> is present	A null authority is a finding, not a blank: it means the act had no recorded grant behind it

D.4 Reporting rules

#	Rule	Why
C14	SILENT and UNKNOWN are never collapsed into one value	SILENT means the source was consulted and says nothing. UNKNOWN means nobody looked. Treating them alike destroys the only distinction that tells a reader whether to go and look
C15	Gap grades are reported individually and never aggregated into a score	A single number invites comparison between organisations on a matter this model has no standing to decide
C16	Gap grade 4 does not arise against performed	Grade 4 is an absence of recorded intent, and an observed act is not an intent
C17	No conformance claim is made while a declared defect is open	A report issued over known malformed data misrepresents its own basis

D.5 What a conformance claim must state

A claim of conformance names four things. Without all four it is not a claim, because nobody can reproduce it.

1. The version of this model the records were validated against
2. Which objects are populated, and which inventories hold no register at all
3. The count of open defects, which must be nought
4. The reference states observed across the graph, because a graph of exclusively Connected references has either been constructed or corrected, and which one it is matters

D.6 Reference implementation

A validator implementing C1 to C13 is released under the Apache License 2.0 alongside this specification. It separates defects from findings in its output and prints the enforcing sentence with each result. It is a reference, not an authority: where the validator and this document disagree, this document governs and the validator is wrong.