Knowledge Foundry

The Reference Architecture — Complete Programme Manuscript

Knowledge Foundry Programme

Compiled 1 August 2026 · Consolidated pre-implementation draft

Table of Contents

About This Manuscript

This manuscript consolidates the complete Knowledge Foundry Reference Architecture into a single volume. It has been assembled from the individual programme documents following a full architecture review and editorial pass; the documents are reproduced in their canonical reading order.

How the programme is organised. Knowledge Foundry separates what is *always true* from *one way of acting on it*, across four inherited levels — **Principle** (the Canon), **Method** (the Methodology), **Architecture** (the Reference Architecture) and **Implementation**. A lower level may choose how to satisfy a higher one; it may never silently revise it.

Reading order of this volume.

- *Orientation* — KF-0000 Overview and KF-0001 Map give the whole programme on a page each.
- *Constitution* — KF-1000 Canon states the enduring principles.
- *Method* — KF-2001 Lifecycle, KF-2002 Engineering Practices, KF-2003 Materiality and Assurance.
- *Architecture* — KF-3000 Reference Architecture and its domains (KF-3001–3005), its behaviour (KF-3006 Integration), its assurance (KF-3007 Conformance), its institutional layer (KF-3008 Authority) and its synthesis (KF-3009 Crosswalks and Traceability).
- *Appendix* — KF-0002 Controlled Vocabulary.

Companion volume. The programme rests on the Literature Landscape (KF-L001), the evidence review from which the Canon is derived. The Landscape is maintained as a separate companion volume and is referenced throughout rather than reproduced here.

Status. Individual documents carry their own status lines. Those marked *Draft for Experiment* (KF-2003, KF-3007, KF-0002) state hypotheses to be validated by the first vertical implementation slice before the architecture is baselined; the remainder are stable drafts. This is a consolidated pre-implementation draft of the programme as a whole.

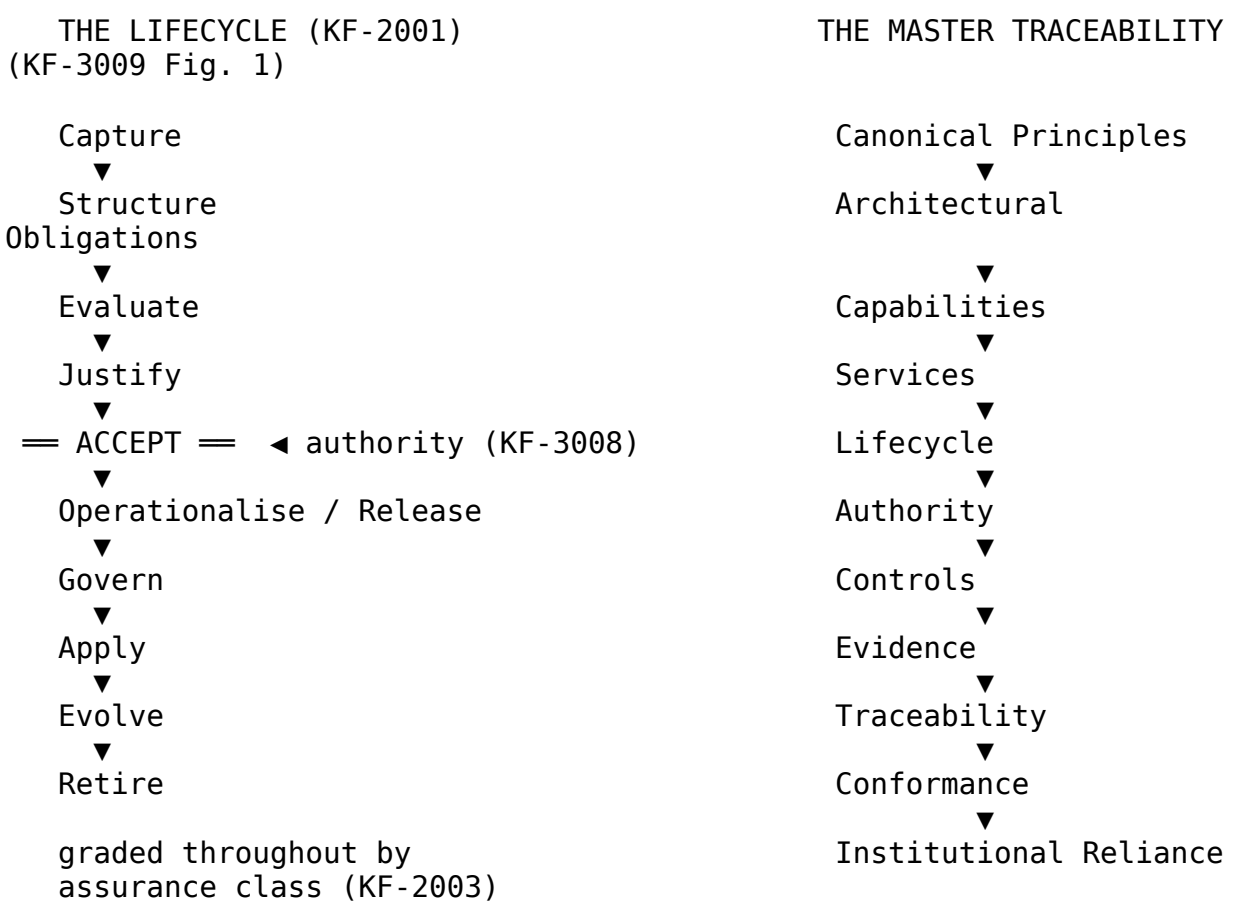
KF-0000 · Knowledge Foundry Reference Architecture — Overview

One page. The visual entry point to the whole programme.

Document Class: Programme-Wide Reference (Navigational) · Status: Draft · Version: 0.1 · No normative content — see home documents for every concept.

LAYERS	DOCUMENTS		OWNS
CONSTITUTION	KF-1000	Canon	principles, invariants
	KF-L001	Literature Landscape	evidence
METHOD	KF-2001	Lifecycle	stages, transitions
	KF-2002	Engineering Practices	practices
	KF-2003	Materiality & Assurance	classes, overlays
ARCHITECTURE	KF-3000	Reference Architecture	the worldview
	KF-3001	Principles & Invariants	AP · INV · IC
	KF-3002	Capability Model	capabilities
	KF-3003	Service Architecture	19 services
	KF-3004	Information & Record Model	records, evidence
	KF-3005	Lifecycle & Control	gates · controls
	KF-3006	Integration	how it behaves
	KF-3007	Conformance	how we know
	KF-3008	Authority	who may act
	KF-3009	Crosswalks & Traceability	how it fits
IMPLEMENTATION	KF-4000	Knowledge Operating System	a deployment
		↳ MacroLab (domain instantiation)	
NAVIGATION	KF-0000	this Overview · KF-0001 Map · KF-0002	

Dependency (top→bottom, one-way): a lower level chooses *how* to satisfy a higher one; it may never revise it.



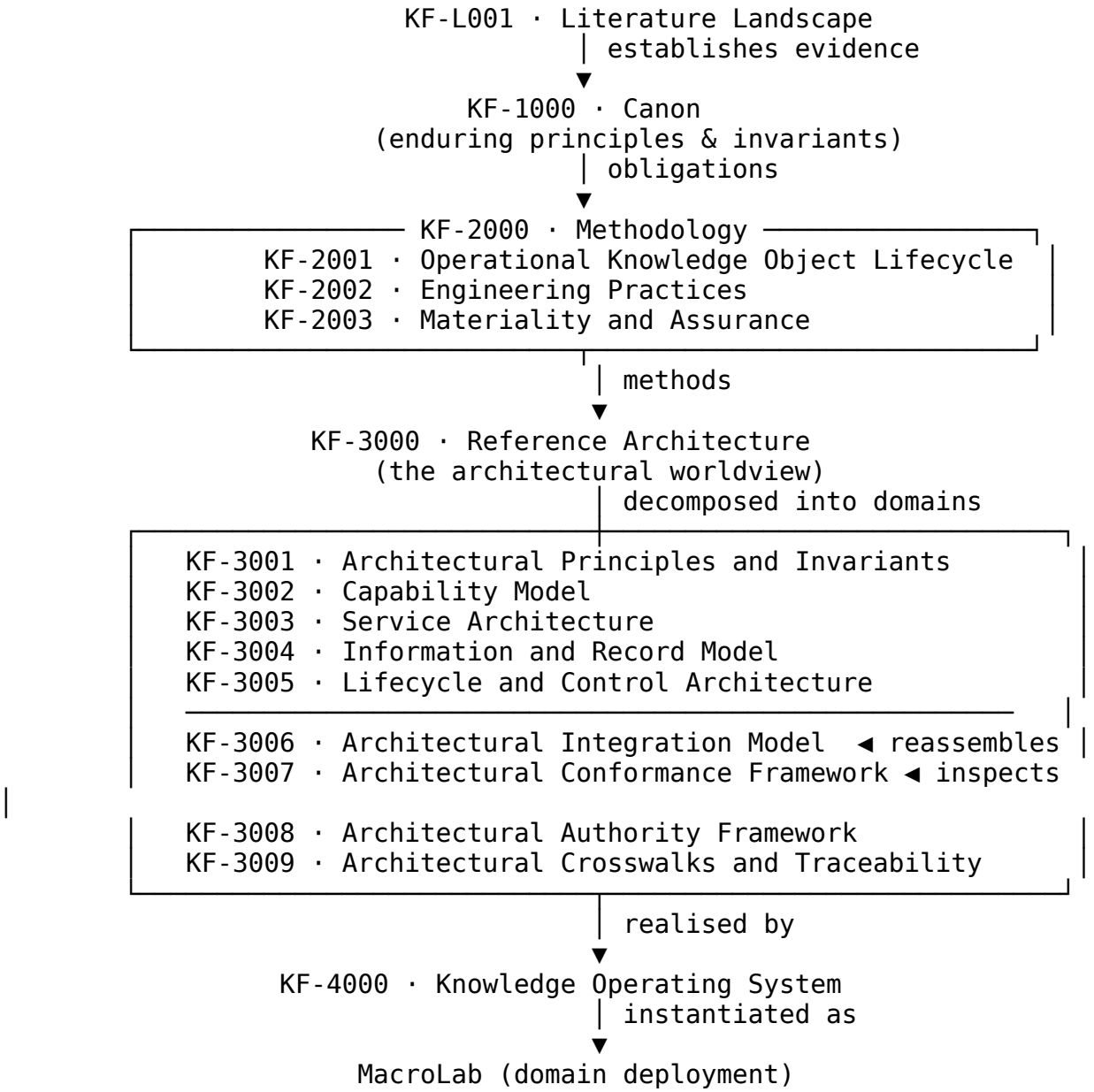
Three questions the concluding documents answer — behave (KF-3006) · know (KF-3007) · legitimate right (KF-3008). **One rule that keeps it coherent** — every concept has exactly one authoritative home (KF-3009 §4).

For the reading guide, concept ownership and all crosswalks, see KF-3009. For terminology, KF-0002. This page introduces nothing; it only orients.

KF-0001 · Architecture Map

One page. The dependency graph of the Knowledge Foundry programme.

Document Class: Programme-Wide Reference · Parent Programme: Knowledge Foundry · Status: Draft · Version: 0.1



Programme-wide references (cited throughout, not in the inheritance line):

KF-0001 · Architecture Map (this document) · KF-0002 · Controlled Vocabulary

KF-5000 · Exemplars & Case Studies (KF-5001 canonical slice · KF-5002+ cases)

Reading the graph. Inheritance runs top to bottom and is one-way: a lower level chooses *how* to satisfy a higher one and may never revise it. KF-3001–3005 decompose the reference architecture into domains; **KF-3006 reassembles them into a working system**; **KF-3007 asks whether that system conforms**; KF-3008 and KF-3009 supply the authority model and the exhaustive crosswalks. KF-2003 grades how deep every obligation below it must be discharged. This page is navigational only; each document governs its own content.

KF-5000 · Exemplars & Case Studies

Worked instantiations of the architecture — illustrative, not normative

Document Class: Series Note (Illustrative) · Parent Programme: Knowledge Foundry · Status: Draft · Version: 0.1

Purpose

The KF-5000 series shows the Knowledge Foundry architecture **at work on concrete Operational Knowledge Objects**. Its documents teach and demonstrate; they do not legislate. Like the KF-0000 navigation series, KF-5000 **introduces no new architectural concept** — every element it shows is defined in its home document (Canon, Lifecycle, Materiality, and the KF-3000 series) and is only *instantiated* here.

The series exists because an architecture is easier to trust once it has been seen to run. A reader who has followed one governed object from capture to retirement understands the whole shape of the system before studying any single part in depth.

The distinction the series preserves

- **KF-5001 · Canonical Vertical Slice** is *illustrative*: a domain-neutral object (a four-day-week Knowledge Brief) constructed to make every architectural stage visible. It proves the parts fit together and that the architecture is subject-independent. It does **not** validate the architecture — a constructed object cannot surprise its designer.
- **KF-5002+ · Case Studies** are *evaluative*: real domain instantiations — beginning with MacroLab's Singapore Business Impact Assessment — in which the architecture meets uncontrolled reality. These are where the architecture is stress-tested, and where the *Draft for Experiment* documents (KF-2003, KF-3007, KF-0002) are eventually validated before baselining.

The order is deliberate: **understand the architecture on the canonical slice; then evaluate it on real cases.** MacroLab is thereby positioned as *a* case study of Knowledge Foundry, not as its defining example.

Contents

ID	Title	Kind
KF-5001	Canonical Vertical Slice — the four-day-week Knowledge Brief	Illustrative exemplar
KF-5002+	Case Studies (MacroLab Singapore Business Impact Assessment, and others)	Evaluative case studies

Relationship to the rest of the programme

KF-5000 sits beside KF-4000 (Implementation): where KF-4000 defines how the architecture is *built* in a deployment, KF-5000 shows how it *behaves* on individual objects. Both are

downstream of the architecture and neither may revise it. For where every concept is defined, see KF-3009 (Crosswalks) and KF-0002 (Controlled Vocabulary).

KF-5001 · Canonical Vertical Slice

One governed knowledge object, walked from capture to retirement — the "Hello, World!" of Knowledge Foundry

Document Class: Exemplar (Illustrative) · Parent Programme: Knowledge Foundry · Parent Series: KF-5000 · Status: Draft · Version: 0.1

```
Instantiates, on one Operational Knowledge Object:
   Canon (KF-1000) · Lifecycle (KF-2001) · Materiality (KF-2003)
   · Reference Architecture and its domains (KF-3000–3005)
   · Integration (KF-3006) · Conformance (KF-3007) · Authority (KF-
3008)
```

Programming has a tradition: before the language reference, a tiny complete program — *"Hello, World!"* — proves the language, compiler and runtime work together, and lets a newcomer see the whole shape of a thing before learning its parts. This document is that program for Knowledge Foundry.

It takes **one** Operational Knowledge Object and follows it from the moment it enters the system to the moment it leaves, showing at every step which services act on it, which records it leaves behind, which gates and controls it must pass, who holds the authority to make it consequential, and how its conformance can later be reconstructed. The reader never has to swap objects out of mind.

> **This document teaches Knowledge Foundry, not the four-day week.** The subject is incidental. A reader should finish understanding *how governed knowledge works* — not holding a view on working hours. Like KF-3009, KF-5001 introduces no new architectural concept; it only instantiates concepts defined in their home documents, cited as it goes.

The example is a **Knowledge Brief** — a governed statement that answers one consequential question — because a brief is the simplest object that still exercises the whole architecture: several kinds of evidence, genuine counter-evidence, an authority that must accept reliance, and a life that continues after acceptance.

---

1 · The Object

> **Knowledge Brief KB-4DW-01 — "Should our organisation move to a four-day working week?"**

Attribute	Value
Identity	KB-4DW-01 (assigned once, immutable for life)
Type	Knowledge Brief — one consequential question, one governed answer
Intended reliance	informs the Executive Committee's decision

Attribute	Value
	on whether to run a trial
Assurance class	**A3 Consequential** (KF-2003) — it will materially influence a significant organisational decision
Overlays	internal · confidential
Steward	the People & Operations lead
Accepting authority	the Executive Committee

That single classification — **A3** — will quietly govern how deep every later step must go. Hold it in mind; §6 returns to it.

---

2 · The Guided Journey

The object advances one lifecycle stage at a time (KF-2001). At each stage below: what happens to the brief · the services that act (KF-3003) · the records it emits (KF-3004) · the gate and controls it passes (KF-3005) · the conformance dimension it exercises (KF-3007 §6). Every record named is **emitted by the act itself** — nobody stops to fill in a separate form (KF-3006 §9).

Capture — state: Candidate

The question is posed and candidate material is admitted: the Iceland public-sector trials, the UK 2022 pilot, two single-firm case studies, several peer-reviewed studies, and our own retention and output data. Capture is deliberately permissive — nothing is trusted yet — but one thing cannot be deferred: **origin**. Each source is recorded with where it came from. *Services:* Object Registry (SVC-01), Provenance (SVC-15). *Records:* Identity Record (KB-4DW-01), Intake Record, Provenance Seed. *Gate:* GATE-REG-01; *controls:* identity resolution and duplicate detection (CTRL-ID-01/02). *Dimension:* Structural. *Authority:* Creation (the Steward) — light, because nothing is yet relied upon.

Structure — state: Structured

The raw material is given form: the question, the candidate claims, the sources and their relationships are made inspectable. The object can now be reasoned about. *Service:* Representation (SVC-02). *Records:* Structured Representation, Metadata, Relationship Map. *Gate:* GATE-STR-01. *Dimension:* Structural.

Evaluate — state: Evaluated

Each piece of evidence is assessed for quality — and they are not equal. The Iceland and UK pilots are large and externally evaluated; the single-firm cases are vivid but weak; our internal data is primary but short-horizon. The **counter-evidence is admitted here, not buried**: reverted pilots, volunteer-firm self-selection, coverage costs in customer-facing roles. *Services:* Evaluation (SVC-04), Evidence (SVC-03). *Records:* Evidence Records, Quality Assessment, Evaluation Verdict, Counter-evidence Records. *Gate:* GATE-EVA-01; *controls:* evidence presence,

evaluation completion, counter-evidence visibility (CTRL-EVD-01/03, CTRL-EVA-01). *Dimension:* Evidence.

Justify — state: Justified

The reasoned case is assembled. Claims are linked to their supporting and defeating evidence; assumptions are written down (*"our role mix resembles the pilot firms; productivity means output-per-week; the window is ≥ 6 months"*); an epistemic status is set:

> *Warranted for a bounded trial among knowledge-worker teams under the stated assumptions — **not yet** warranted for permanent, organisation-wide adoption. Counter-evidence and the sector-dependence claim remain attached.*

Service: Justification (SVC-05). *Records:* Justification Record, Assumption Record, Rationale Record, Epistemic Status Record. *Gate:* GATE-JUS-01; *control:* justification completeness (CTRL-JUS-01). *Dimension:* Evidence. Because the object is **A3**, this stage also draws an **independent reviewer** (§6) — an A1 note would not have.

Accept — transition; state: Accepted

Now the decisive act, and the one the whole architecture is built around: **justification is not acceptance.** The brief is well-warranted, but warrant is not reliance. The **Executive Committee** — not the author, not the reviewer — decides that the organisation will rely on it, and fixes the *scope* of that reliance: *a six-month trial for knowledge-worker teams,* not permanent adoption. *Service:* Authority and Acceptance (SVC-06). *Records:* Acceptance Record, Scope-of-Reliance Record. *Gate:* GATE-ACC-01; *controls:* authority mandate and acceptance validity (CTRL-AUT-01, CTRL-ACC-01). *Dimension:* **Authority.** The acceptance record — naming the authority, the scope and the date — *is* the evidence that acceptance occurred (§4).

Operationalise — state: Operational

The accepted brief is turned into something the organisation can act on: a recommendation to **run a six-month four-day-week trial for knowledge-worker teams, measured on output-per-week and retention**, linked back to the warrant that justifies it. *Services:* Operationalisation (SVC-12), Release (SVC-10). *Records:* Actionable Form, Warrant Linkage, Release Record. *Gate:* GATE-REL-01. *Dimension:* Governance.

Govern — state: Governed

The object is given an owner and a review schedule so its legitimacy is *maintained*, not assumed. Currency conditions are set: a new major study or the trial's own results will trigger review. *Service:* Stewardship and Review (SVC-07). *Records:* Stewardship Assignment, Review Schedule. *Gate:* GATE-REV-01. *Dimension:* Governance.

Apply — state: in use → outcomes

The recommendation informs the actual trial decision. Its use is recorded, and the trial's results — output held steady, retention rose, two client-facing teams struggled — return as outcomes. *Service:* Application and Outcome (SVC-13). *Records:* Application Record, Explanation Record, Outcome Record, Feedback. *Gate:* GATE-APP-01. *Dimension:* Behavioural.

Evolve — state: Next Version

The trial's outcomes are new evidence. The brief re-enters at Evaluate, its claims are re-justified against real internal data, and — because a new version does not inherit warrant — it **passes through Acceptance again**. Version 2 recommends extending the trial for knowledge-worker teams and excluding client-facing teams pending a different approach. *Service:* Version and Lineage (SVC-09). *Records:* Version Record, Change Record, Lineage Record. *Gate:* GATE-EVO-01. *Dimension:* Lifecycle. The predecessor is not overwritten; v1 and v2 are linked.

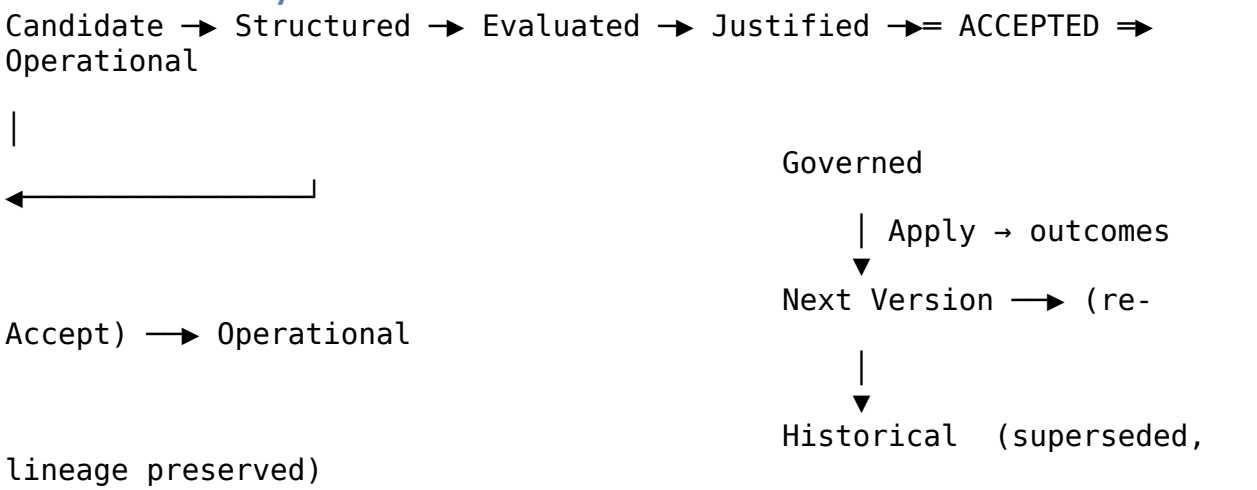
Supersede / Retire — state: Historical

A year on, the brief is folded into a broader **Flexible Working Framework** that governs four-day weeks, hybrid and compressed hours together. KB-4DW-01 is superseded — its successor linked, its history preserved append-only. Had the idea been abandoned instead, it would have been **retired** with the reason recorded. Either way, nothing is erased: the organisation can still reconstruct, years later, what it once decided and why. *Service:* Retirement and Preservation (SVC-14), with Version and Lineage (SVC-09). *Records:* Supersession Link (or Retirement Record), Historical Object Record. *Gate:* GATE-RET-01. *Dimension:* Traceability.

What only the whole journey shows. Every record above was emitted by *doing the step*, not by a separate filing exercise — the journey *is* the evidence trail. And the six lifecycle invariants held at every step: one identity throughout, every transition attributed and justified and inspectable, provenance present from Capture, lineage preserved through Evolve and Supersession.

---

3 · The Same Story as a State Machine



Every arrow is a governed transition: recorded, attributed, justified. There are no silent moves — that is the single difference between this and a document that merely changes status.

---

4 • Conformance, Demonstrated

Long after the decision, someone asks: *was this brief properly governed?* KF-3007 answers it against the object, through four tests:

- **Constitutional coverage** — every act traces to a Canonical Principle: the object was governed not as knowledge but as an OKO (I), engineered as a lifecycle (II), accepted by collective authority (III), justified before reliance (VIII).
- **Operational coverage** — every transition has a trigger, an authority, criteria and a recorded outcome; every gate evaluated explicit exit conditions.
- **Integrity coverage** — the six invariants held; counter-evidence stayed attached; evidence was proportionate to **A3**.
- **Reconstructability** — the material claims can be rebuilt from the record.

A worked Conformance Claim.

> *Claim:* "The acceptance of KB-4DW-01 held sufficient authority for A3 reliance." *Supported by:* the Acceptance Record (authority = Executive Committee; scope = six-month trial; date), the Authority Mandate Record confirming the Committee's remit for A3 decisions, and the independent-review record required at A3. *Assessment:* Constitutional + Operational + Integrity + Reconstructability all satisfied. *Finding:* **Conformant**, at **high confidence** — the evidence is complete, event-native and independently corroborated (KF-3007 §13.3).

Had the independent review been missing, the same reconstruction would have returned **Not Assessable** for that claim — never a silent pass.

---

5 • Authority, Demonstrated

The acceptance was made by the **Executive Committee**, and the architecture can resolve that authority all the way down (KF-3008 §5.1):

```
Decision: accept KB-4DW-01 for a six-month trial
     ▲ resolves to
Acceptance Authority
     ▲ carried by
Role: Executive Committee
     ▲ defined by
Governance Structure
     ▲ constituted by
The Institution
```

Because the object is **A3**, three authority conditions applied that a lower class would not have required: the accepting authority was **confirmed**, an **independent review** stood between justification and acceptance, and **separation** held — the People & Operations lead who *evaluated and justified* the brief was **not** the body that *accepted* it (KF-3008 §11). Authority here is not seniority: the Committee's right to accept rests on its mandate for consequential decisions, not on rank.

6 · Materiality, Demonstrated

Everything above was shaped by one value set at the very start: **A3 Consequential.** Materiality never changed the *path* — an A0 note and an A4 board paper cross the same gates — but it set the *depth* at each one (KF-2003; KF-3001 IC-23):

At this stage	Because it is A3	Had it been A1 (Working)
Justify	independent review; counter-evidence recorded; explicit assumptions	a named owner's own reasoning would suffice
Accept	confirmed authority; defined scope of reliance	no acceptance required at all
Govern	active monitoring; scheduled review	review on change only
Reconstruct	claims reconstructable end-to-end	shallow, on request

The six invariants held identically at both classes. **Only the depth above that floor scaled.** This is the whole of materiality in one object: same architecture, same guarantees, proportionate effort.

7 · The Point — Generality

Nothing in the journey depended on the four-day week. Swap the subject and every stage, service, record, gate, authority and conformance test is unchanged:

Domain	Example Operational Knowledge Object
Personal knowledge	a literature review on spaced repetition
Organisation	**the four-day-week brief** *(this slice)*
Research	a systematic-review conclusion
Healthcare	a clinical guidance note
Education	a curriculum standard
Engineering	a design decision record
Economics (MacroLab)	a Singapore Business Impact Assessment

> **Knowledge Foundry does not govern management, medicine, engineering or economics. It governs Operational Knowledge Objects. The subject matter changes; the architecture does not.**

That is what the slice exists to prove — and why the architecture, not the example, is the thing to remember.

---

8 • Using This Exemplar

Every architecture document can point to the same running object rather than inventing its own:

To see…	Read KF-5001…	Alongside
the lifecycle on a real object	§2	KF-2001
the elements collaborating	§2–§3	KF-3006 §6
how conformance is demonstrated	§4	KF-3007
who may make it consequential	§5	KF-3008
how reliance sets the depth	§6	KF-2003
that the architecture is domain-neutral	§7	the Canon's central claim

Status and Honest Limits

This is a **canonical illustration**, and it illustrates — it does not validate. A constructed object confirms the architecture on an object built to succeed; it cannot *surprise* us, and surprise is what an experiment is for. The documents still marked *Draft for Experiment* (KF-2003, KF-3007, KF-0002) therefore remain provisional until one **real, uncontrolled** object has been run and allowed to break something. KF-5001 makes the architecture legible; it does not close that experiment. That is the honest boundary of a "Hello, World!" — it proves the parts fit together, not that they survive the world.

KF-1000 · Knowledge Foundry Canon

The Foundational Principles of Governed Knowledge Systems

Document Class: Canon · Parent Programme: Knowledge Foundry · Status: Draft · Version: 0.1

```
Principle → Method → Architecture → Implementation
```

Why a Canon?

Engineering disciplines are organised around enduring principles that remain valid even as methods, technologies and implementations change. Mechanical engineering outlived the steam engine. Software engineering outlived the languages it was first written in. The principles endure because they describe obligations, not instruments.

Knowledge engineering adopts the same structure. Its tools — expert systems, ontologies, knowledge graphs, foundation models — have already turned over several times, and will again. What does not turn over is the set of obligations a system incurs the moment it holds knowledge on behalf of people who will act on it.

This document states those obligations. It is a Canon, not a manual. The Literature Landscape is where the discipline weighs evidence and persuades; the Canon is where it legislates. Everything that follows is written to be inherited unchanged by the methods, architectures and implementations built beneath it, and to remain true long after any of them is replaced.

The Purpose of the Discipline

Every mature discipline has a single sentence that explains why it exists — medicine's *first, do no harm*, accounting's *true and fair view*. Knowledge engineering has one too:

> **The purpose of knowledge engineering is to preserve, improve and govern the evolution of justified knowledge so that it remains trustworthy across time, institutions and computational modalities.**

Every principle in this Canon serves that sentence. A statement that does not serve it does not belong here.

Purpose

These principles define the intellectual foundations of **governed knowledge systems**. Knowledge Foundry is one methodology that adopts them; it is not their source, and they do not depend on it.

The Canon derives the principles that any governed knowledge system must satisfy — regardless of implementation technology, organisational setting, or computational modality. Subsequent methodology, architecture and implementation documents inherit them rather than redefining them.

A principle earns its place here only if it is true of the discipline, not merely of one methodology. The test is stated in *The Discipline and Its Four Levels* and applied to every principle that follows.

Relationship to the Literature Landscape

KF-L001 answered one question: *What does the literature establish?* The Canon answers a different one: *What design principles necessarily follow from that evidence?*

The Literature Landscape is the evidential authority; the Canon converts settled evidence into obligation. No new literature is introduced. Where a principle is grounded in the Landscape, its derivation is cited by chapter and by Landscape Proposition (L-P1–L-P6), Proposition B01, Proposition C01, the six Governance Primitives, or the requirements register (R1–R10). The Canon adds no evidence of its own.

The Discipline and Its Four Levels

A mature engineering discipline separates what is *always true* from what is *one way of acting on that truth*. Knowledge Foundry adopts four levels. Each inherits from the one above; none may masquerade as another.

Level	Answers	Stability	Home document
Principle	What is always true of a governed knowledge system?	Enduring	KF-1000 (this Canon)
Method	How do we act on that truth?	Revisable	KF-2000 (Methodology)
Architecture	With what structures and components?	Substitutable	KF-3000 (Reference Architecture)
Implementation	In what specific deployment?	Disposable	KF-4000 (Knowledge Operating System), and its domain instantiations

The worked example that fixes the boundaries:

Level	Statement about provenance
Principle	*A knowledge state is legitimate only if the transition that produced it is inspectable.*
Method	*Track each operational knowledge object with a provenance graph of its transitions.*
Architecture	*Represent that graph in a property-graph store with signed, append-only edges.*
Implementation	*A domain deployment — such as MacroLab — records provenance in its Knowledge Operating System.*

The Canon Admission Test

Every candidate principle is admitted only if it survives two questions:

1. **Would this still be true if Knowledge Foundry ceased to exist?**
2. **Would this still be true fifty years from now, even if today's AI models were obsolete?**

If either answer is *no* — if the statement presupposes a particular method, structure, product, deployment or generation of technology — it belongs below the line, in the Methodology, the Reference Architecture, or an Implementation. The two questions guard the two ways a statement fails to be constitutional: by belonging to one project, or by belonging to one era. This test is what separates a discipline from a framework. It is applied to each of the ten principles, every one of which closes with a *Below the line* note recording what it excludes.

Each principle is *derived primarily from* a chapter of the Landscape, but **no principle is a summary of a chapter**. A summary restates evidence; a principle states an obligation the evidence makes unavoidable. Several principles draw on the whole Landscape, and the later ones arise only when the chapters are read together.

How to Read This Canon

Part I defines the object and the system. Part II states the invariants that make principles necessary. Part III states the ten principles. Part IV draws their design consequences without descending into method. Part V is the bridge to KF-2000. Read in order, the Canon moves from *what a governed knowledge system is* to *what its existence obliges* — and stops where method begins.

Part I · Foundations

KF-1001 · The Nature of Governed Knowledge Systems

Purpose. Establish the central object of the discipline and define what a governed knowledge system is.

The central object

Knowledge systems never reach past their representations to touch knowledge itself (Proposition B01; L-P1). What they hold and change is the **Operational Knowledge Object**: a versioned representation together with its attached evidence, justification and acceptance-state. It is the concrete instance of the abstract **Conceptual Object** — `representation → claim → evidence → justification → acceptance` — that B01 defines.

The discipline takes the Operational Knowledge Object, not "knowledge," as its unit of design. This is precision, not modesty. A system cannot engineer truth, because truth is not inspectable. It engineers the object that stands in for knowledge and carries the warrant a community attaches to it.

Formal definition

> A **governed knowledge system** is a socio-technical system that holds operational knowledge objects and governs their transition from one justified state to another —

across human, symbolic, parametric and hybrid modalities — so that warranted claims remain available, revisable and trustworthy over time.

Four terms in this definition are canonical and are used throughout the programme:

- **Governed evolution** — the controlled transition of a knowledge system from one justified state to another; the discipline's characteristic operation, and the persistent challenge the Landscape identifies (L-P6).
- **Human–machine system** — the system boundary includes the people and institutions that create, authorise and depend on its knowledge. They are inside the system, not external to it.
- **Knowledge infrastructure** — knowledge treated as durable, shared, continuously maintained substrate rather than as a bounded artefact or a project deliverable.
- **Computational modality** — the cross-cutting axis (Human · Symbolic · Parametric · Hybrid) along which the system's operations are redistributed but never removed (L-P5).

Deliverable. The definition above is the discipline's fixed point. Every principle in Part III is a property of the system so defined; every method in KF-2000 is a way of realising one of those properties.

Part II · Constitutional Invariants

KF-1050 · What Never Changes

Technology changes. Methods change. Architectures change. Products change. Institutions change. Yet some obligations remain invariant across all of them, and it is these invariants that make principles necessary. The ten principles of Part III are not free postulates; each one operationalises an invariant named here. This chapter is the Canon's preamble: it states the properties that any governed knowledge system must exhibit in every era, before any principle prescribes how.

Five invariants hold.

Inspectability. Whatever a knowledge system holds must be capable of being examined. A system whose contents cannot be inspected cannot be justified, explained, audited or corrected. Inspectability is prior to all of these; it is the condition that makes governance possible at all.

Justification. A claim held as knowledge must carry the grounds on which it is held. Warrant, not mere assertion, is what distinguishes knowledge from information. A system may hold much that is unjustified, but it may not call that knowledge.

Provenance. The history of a claim — where it came from, how it changed, on whose authority — must be recoverable. A present state that cannot be traced to the transitions that produced it cannot be trusted, however plausible it appears.

Governed evolution. Knowledge changes, and the change must be controlled rather than incidental. A system in which states can shift without proposal, evidence, authority and release does not hold knowledge; it accumulates content.

Human accountability. Some identifiable person or body must answer for what a system accepts as knowledge. Operations may be distributed across people and machines, but accountability may not be discharged into a substrate. There is always someone who is answerable.

These five are not yet the principles. They are the properties whose loss would make a system cease to be a governed knowledge system. The principles that follow give them force: they state, for each invariant, the obligation it imposes on anyone who would build such a system. An invariant answers *what must always hold*; a principle answers *what that obligates us to do*.

Part III · The Canonical Principles

The ten principles are the constitutional core of the discipline. Each is stated as an obligation, derived from the Landscape, shown to survive both questions of the Admission Test, and closed with the line it must not cross. They are numbered I–X and carried under document IDs KF-1101–KF-1110 so that later documents can cite a principle in two characters.

KF-1101 · Principle I — Knowledge systems govern operational knowledge objects, never knowledge itself

Derived primarily from B01 (Proposition B01; L-P1; the Inspectable Objects Principle). Invariant: inspectability.

A knowledge system does not contain knowledge. It contains representations, the evidence and justification attached to them, and their acceptance-state under some authority. “Knowledge” is the name a community gives to an operational knowledge object that has cleared those tests. Truth and the tacit remainder lie permanently outside the system’s reach.

This principle fixes the object every other principle acts upon. It dissolves a standing confusion: whether a system “really contains knowledge” is mis-posed. No system does, and none needs to. What a system holds, warrants, governs and revises is the operational knowledge object.

Admission Test. True of every knowledge system ever built, from a 1970s rule base to a foundation model, and true of any yet to be built. Passes both questions.

Below the line (not Canon): how an operational knowledge object is represented is **method**; the store that holds it is **architecture**; a particular deployment is **implementation**. The Canon fixes only that the object — not knowledge — is what is governed.

KF-1102 · Principle II — A knowledge system is engineered as a lifecycle, not as an artefact

Derived primarily from B02 (L-P2; the expanding object of engineering). Invariant: governed evolution.

The object of knowledge engineering has widened, across five decades, from expert reasoning to whole hybrid systems. With that widening, the engineering task moved from constructing an artefact once to governing its continuous change. To engineer a knowledge system is to engineer its lifecycle: the unit of engineering is the *transition*, not the finished representation.

Continuous evolution is not a feature added to a built artefact. It is the thing being built. A method designed for one-off, up-front construction contradicts this principle.

Admission Test. Any system that holds knowledge over time faces change whether or not it is designed for it; the obligation to engineer that change is independent of any technology or project. Passes both questions.

Below the line (not Canon): which lifecycle stages to define, and how to move between them, is **method**; the pipelines and stores that realise them are **architecture**.

KF-1103 · Principle III — Knowledge becomes enduring capability only through collective authorisation

Derived primarily from B03 (L-P3). Invariant: human accountability.

Knowledge that endures is not a possession of an individual or a property of a machine. It is a capability of a collective, held in place by authorisation, legitimation and stewardship. Acceptance — the act by which a community with authority certifies a claim as settled — is what turns a well-justified representation into "what we know."

A governed knowledge system is socio-technical by necessity, not by preference. Its boundary includes the people and institutions that warrant its claims. A system that treats authorisation as external to itself has mislocated the source of the durability it depends on.

Admission Test. The collective and authorised character of enduring knowledge holds across organisations, disciplines and centuries. Passes both questions.

Below the line (not Canon): which roles hold which decision rights, and through what workflow authorisation flows, is **method** and **architecture**; who staffs those roles is **implementation**.

KF-1104 · Principle IV — Governance is constitutive of knowledge, not supervisory over it

Derived primarily from B04 (L-P4; the six Governance Primitives). Invariant: justification.

Governance is not a control layer placed on top of finished knowledge. It is what makes a representation count as knowledge at all. Strip away evidence, authority, provenance, validation, versioning and reversibility — the six Governance Primitives — and what remains is information, not knowledge. The primitives are the operations by which warrant is conferred; warrant is what distinguishes the two.

This inverts the intuition that governance constrains an otherwise-complete asset. Governance is the asset's condition of existence. Stated at full strength: a knowledge state is only as legitimate as the governance that produced it.

Admission Test. The dependence of knowledge-status on warrant, and of warrant on governance, is a claim about knowledge, not about any methodology or era. Passes both questions.

Below the line (not Canon): how each primitive is instantiated — the shape of a validation gate, the format of a provenance record — is **method** and **architecture**. The primitives themselves are constitutional.

KF-1105 · Principle V — Computational modality redistributes epistemic labour but never discharges epistemic obligation

Derived primarily from B05 (L-P5; "the substrate changes; the epistemic obligations do not"). Invariant: justification.

A new computational substrate — parametric, agentic, hybrid — changes *where* knowledge resides and *who or what* performs each operation. It does not change the obligations to justify, authorise, preserve and govern the resulting claims. Foundation models introduced a new representation, not a new epistemology; they relocated capability across the symbolic–parametric boundary rather than abolishing it.

This principle forbids the recurring error of granting a modality an epistemic discount — of treating a fluent generated answer as if it carried warrant it has not earned. Opacity of substrate is not a licence for opacity of obligation.

Admission Test. The principle is written about *any* substrate, not today's. It survives the obsolescence of current models precisely because it refuses to attach obligation to a technology. Passes both questions — and Question 2 is its point.

Below the line (not Canon): how to honour an obligation in a difficult modality — grounding generation in retrieved evidence, reconstructing provenance for a model weight — is **method**; the model and index that do so are **architecture**.

KF-1106 · Principle VI — A knowledge system is irreducibly engineered, collective and governed at once

Derived primarily from C01 (Proposition C01; the integrated reference model). Invariant: governed evolution.

Engineering, capability and governance are not three kinds of knowledge system. They are three operations that every knowledge system undergoes simultaneously on a single object. No system is only engineered, only collective, or only governed. To hold any one of these steady while working only on another — the division of labour the Landscape found across the literature — is to describe part of a system, never the whole.

The deficiency the Landscape identified was therefore one of **coordination**: the literature explains each operation in isolation and does not explain their coordination over the lifecycle of one governed object (Proposition C01). This principle makes coordination a first-class obligation.

Admission Test. The irreducibility of the three operations is a structural fact about knowledge systems, independent of who coordinates them or when. Passes both questions.

Below the line (not Canon): the coordination mechanism — how a single change propagates across engineering, capability and governance — is the central **method** of KF-2000. The Canon fixes only that the three are inseparable.

KF-1107 · Principle VII — Trust in a knowledge system is a property of its transitions over time, not of its state

Derived from B04 and C01 (L-P6; the records-continuum and provenance traditions). Invariant: provenance.

A knowledge system is not trustworthy because of what it holds at an instant. It is trustworthy because the history of how it came to hold it can itself be trusted. Trust is carried by transitions — by the appraisal, provenance, authenticity and controlled release that survive each change — not by any snapshot. Change is constitutive of a knowledge system; the question is whether trust survives it.

This is the temporal principle. It obliges a system to treat its own past as evidence: to retain, reconstruct and authenticate prior states, so that a present claim can be traced to the transitions that produced it.

Admission Test. That trustworthiness-over-time is a property of change rather than of state holds for any records system or knowledge system, in any era. Passes both questions.

Below the line (not Canon): retention windows, versioning schemes and authenticity mechanisms are **method** and **architecture**; how long an institution keeps a given state is **implementation**.

KF-1108 · Principle VIII — No knowledge state is legitimate without an inspectable justification for how it was reached

Derived from B01 and B04 (Proposition B01; Evidence/Justification and Provenance primitives; R2, R4, R6). Invariants: justification, inspectability.

Every transition of an operational knowledge object carries its justification: the evidence, reasoning and authority under which the new state was accepted. Acceptance without recorded, inspectable justification is assertion, not knowledge. A system can engineer justification even though it cannot engineer truth — justification is the richest thing it can govern — and so justification, not truth, is the standard to which every transition is held.

Principle VIII is the per-transition consequence of Principle IV: if governance is constitutive, each individual change must bear the warrant that governance confers. A knowledge state whose justification cannot be inspected has no standing, however plausible it appears.

Admission Test. The requirement that legitimate knowledge carry inspectable justification is epistemic, not methodological, and not tied to any generation of tools. Passes both questions.

Below the line (not Canon): the structure of a justification record and the interface that exposes it are **method** and **architecture**.

KF-1109 · Principle IX — A knowledge system must remain explainable irrespective of the modality in which its knowledge resides

Derived from B02, B05 and C01 (the explanation imperative; the parametric opacity findings). Invariants: inspectability, human accountability.

The demand that a system explain its conclusions is one of the discipline's oldest commitments, and it does not lapse when knowledge moves into an opaque substrate. Where the substrate cannot be read directly — as with parametric knowledge — explainability becomes an obligation the system meets *by design around* the substrate, not an excuse the substrate provides. Explainability is owed by the system, not by the component.

The principle sets a standard the hardest modality must still meet. A system may hold knowledge it cannot introspect; it may not therefore hold knowledge it cannot account for.

Admission Test. Explainability as an obligation of knowledge systems predates today's models and will outlast them; the techniques will change, the obligation will not. Passes both questions — and Question 2 is its point.

Below the line (not Canon): the techniques that deliver explanation — retrieval grounding, citation, surrogate inspection, neuro-symbolic structure — are **method** and **architecture**, and will change; the obligation will not.

KF-1110 · Principle X — A knowledge system is designed for governed evolution, not for preservation

Derived from the entire Landscape (L-P2, L-P6; Proposition C01). Invariant: governed evolution.

The default condition of knowledge is change. A governed knowledge system is designed, from the first, for governed evolution. Preservation is a special, bounded case of evolution — the decision to hold a state trustworthy across time — not the system's natural resting point. A design that treats stability as the norm and change as the exception has the discipline backwards.

This principle gathers the other nine into a posture. The object is governed (I, IV); its engineering is a lifecycle (II); its durability is collective (III); its obligations survive any substrate (V, IX); its coordination is inseparable (VI); its trust and its legitimacy live in its transitions (VII, VIII). What persists through every iteration is not knowledge itself, but the governed transition from one accepted state to the next.

Admission Test. That knowledge systems face continuous change, and are best designed for it, is true of the domain irrespective of any methodology or era. Passes both questions.

Below the line (not Canon): the mechanics of a knowledge release — quality gates, rollback, promotion between states — are the core **method** of KF-2000; their components are **architecture**; a running release cadence is **implementation**.

Part IV · Design Consequences

The principles are not methods, but they are not inert: they constrain the space of admissible methods, architectures and implementations. This part records those constraints without crossing into method. Each consequence is a boundary on KF-2000 and below.

KF-1201 · Engineering Consequences

The unit of engineering is the transition (II) and the object is the operational knowledge object (I). Governance cannot be bolted on after construction: versioning, provenance and justification are first-class from the first design increment (IV, VII, VIII). Representations must be inspectable, because a representation that cannot be inspected cannot be justified or explained (VIII, IX). Because the three operations are inseparable (VI), an engineering method that optimises one in isolation — a pipeline that builds without governing, or governs without accounting for modality — is inadmissible.

KF-1202 · Governance Consequences

All six Governance Primitives are present, or the system governs incompletely (IV). Authority is explicit and remains human, because legitimation is a collective act (III) and redistribution of labour across modalities does not redistribute accountability (V). Appraisal — deciding what to promote, retain and retire — is part of governance, not separate from it, because trust is a property of transitions over time (VII). Governance that only records change, without adjudicating whether a change should be promoted into an accepted state, satisfies none of these principles.

KF-1203 · AI Consequences

Parametric and agentic knowledge is subject to exactly the same obligations as symbolic knowledge (V). The substrate makes some obligations harder to honour — provenance and reversibility above all — but does not lift them. A generative core is engineered into a governed system, grounded in inspectable evidence, and made explainable by design around its opacity (VIII, IX). Autonomous agents may perform operations, but the authority to accept a claim into the knowledge state remains human (III, V). Any AI-native method that treats fluency as warrant, or model turnover as acceptable knowledge loss, violates the Canon.

KF-1204 · Institutional Consequences

Knowledge is infrastructure (KF-1001): an institution's knowledge is a shared, maintained, evolving asset, not a collection of documents or a set of model weights. The organisation and its decision rights are inside the system (III). Continuity of collective memory survives personnel change and model change alike (VII, X). Institutions hold identifiable authority for knowledge transitions, and do not allow knowledge that lives only in a substitutable substrate to become knowledge they cannot guarantee they still possess.

Part V · Transition to Methodology

KF-1301 · From Principles to Methods

The Canon ends where method begins. Its final task is to hand a clean set of obligations to KF-2000 and to fix, once, the rule that governs the handoff.

Each principle becomes a demand on method without dictating the method:

Principle	Obligation handed to KF-2000
I	Operate on operational knowledge objects as the unit of work
II	Engineer the lifecycle, not the artefact
III	Route authorisation through identified collective authority
IV	Instantiate all six Governance Primitives
V	Preserve every obligation across every modality
VI	Coordinate engineering, capability and governance on one object
VII	Preserve trust across transitions over time
VIII	Attach inspectable justification to every transition
IX	Deliver explanation irrespective of substrate
X	Default to governed evolution; treat preservation as its special case

KF-2000 proposes methods that discharge these obligations; KF-3000 specifies architectures that realise those methods; KF-4000 demonstrates them in a reference implementation. The direction of inheritance is strict and one-way: **a lower level may choose how to satisfy a higher one, but may never revise it.** A method that cannot be reconciled with a principle yields to the principle — or the principle is challenged openly, here in the Canon, on the evidence of the Landscape, and not quietly overridden in practice.

Relationship to Subsequent Documents

- **KF-L001 — Literature Landscape** establishes the evidence.
- **KF-1000 — Canon** establishes the principles.
- **KF-2000 — Methodology** defines the methods.
- **KF-3000 — Reference Architecture** specifies design patterns and system components.
- **KF-4000 — Knowledge Operating System** provides a reference implementation.
- **MacroLab** is a domain-specific instantiation of the Knowledge Operating System.

Each document inherits, rather than revises, the Canon. A change to a principle is a constitutional amendment, made against the Landscape's evidence and propagated downward;

a change to a method, architecture or implementation is ordinary work that leaves the Canon untouched.

Closing Statement

The Literature Landscape demonstrated that the constituent elements of governed knowledge systems already exist across multiple mature traditions. The Canon states the enduring principles that integrate those elements into a coherent discipline.

It is the constitutional foundation from which methodology, architecture and implementation derive. Its authority rests on a single discipline: every statement it contains would remain true if Knowledge Foundry itself were to disappear, and would remain true fifty years from now with today's technologies long replaced. That is what makes it a Canon rather than a framework — and what the work built upon it must never be allowed to erode.

KF-2001 · Operational Knowledge Object Lifecycle

The canonical engineering lifecycle of a governed knowledge object, from capture to retirement

Document Class: Methodology · Parent Programme: Knowledge Foundry · Parent Document: KF-2000 · Status: Draft · Version: 0.1

```
Principle (KF-1000)  →  ▸ Method (KF-2000)  →  Architecture (KF-3000)
→  Implementation (KF-4000)
```

Purpose

This document defines the canonical engineering lifecycle through which an Operational Knowledge Object evolves from initial capture to eventual retirement. It is the common engineering model beneath all Knowledge Foundry methods; the methods that follow in KF-2000 act on the stages defined here.

The lifecycle models the evolution of *justified knowledge*, not the storage of *information*. A document-management system versions files and advances a record whenever it is saved; this lifecycle advances an object only when the obligations of its current stage are discharged. The difference is the gate. In a storage system a transition is a save event; here a transition is an act of warrant, and an object that has not earned the warrant of its stage does not advance, however complete it looks.

How This Document Sits in the Programme

KF-2001 is a **Method-level** document. It says *how the discipline acts* on the principles fixed by the Canon (KF-1000); it does not restate or revise them. Consistent with the four-level separation, this document names no technologies, stores or products — those are **architecture** (KF-3000) — and no deployments — those are **implementation** (KF-4000). Where a stage discharges a Canonical Principle, the principle is cited in a single *Canon basis* line; the authoritative stage-to-principle mapping is KF-2002, and is not duplicated here.

Method, Not Workflow

This lifecycle specifies the engineering *obligations* associated with each stage — what must be true for an object to advance — not the *workflow* by which any particular organisation satisfies them. Method fixes the obligation; workflow fixes the order of work, the tools, the hand-offs and the cadence by which a given deployment meets it. Workflow therefore belongs to implementation (KF-4000), not here.

Conformance is not a matter of following a prescribed sequence of steps. It is whether, for every object, every obligation and every lifecycle invariant is met. Two organisations may run very different workflows — one manual and deliberative, another largely automated — and both conform to this lifecycle, provided no gate is passed unearned and no invariant is broken.

The Object and Its States

The object under management is an **Operational Knowledge Object** throughout its life. What the lifecycle changes is its *state*, not its identity. The stage outputs below name successive states

of one object — Candidate, Structured, Evaluated, Justified, Accepted, Operational, and so on — not distinct objects. An object becomes an Operational Knowledge Object in the full Canon sense once it carries representation, evidence, justification and an acceptance-state; the early stages assemble those attachments onto an object that already holds a stable identity.

> **Terminology note (flagged design decision).** "Operationalise" (Stage 5) makes an object's knowledge *actionable*; it is not the same word as "Operational Knowledge Object," which names the artefact across its whole life. Where the two could be confused, this document calls the Stage-5 result the object's *actionable form* and reserves "Operational Knowledge Object" for the artefact. The stage names from the source model are preserved unchanged.

Lifecycle Overview

> **Worked example.** KF-5001 walks a single Operational Knowledge Object — a four-day-week Knowledge Brief — through every stage below, from Capture to Supersession.

```
Capture
   ↓
Structure
   ↓
Evaluate
   ↓
Justify
   ↓
══ Accept ══   (an authority certifies the object — a named
transition, not a stage)
   ↓
Operationalise
   ↓
Govern
   ↓
Apply
   ↓
Evolve
   ↓
Retire
```

Each stage is a distinct engineering responsibility with a defined **entry condition** and an **exit gate**. An object progresses only when the exit gate of its current stage is satisfied; until then it remains in place or returns to an earlier stage. The stages are not all of one kind: Capture through Justify *construct* the object; **Accept** is a decision by authority, not engineering work, and is modelled as a transition; Operationalise *realises* it; Govern holds it in *custody*; Apply *uses* it; Evolve *changes* it; Retire *closes* it. Application is the one step that does not transform the object — it is an operational context in which the object is used and from which evidence returns.

The sequence above is the *principal path*, not the only path. As evidence changes, an object may return to an earlier stage, and from several steps it may pass directly to Retire. The lifecycle is a

loop, not a ladder — a property inherited from the Landscape's account of the knowledge object and made operational here through the return paths defined below.

#	Stage	Engineering responsibility	Output state	Exit gate (in one line)
1	Capture	Admit candidate information	Candidate	Origin recorded
2	Structure	Give the object identity and form	Structured	Object is inspectable
3	Evaluate	Assess quality before acceptance	Evaluated	Quality verdict recorded
4	Justify	Establish epistemic legitimacy	Justified	Justification is inspectable
—	**Accept** *(transition)*	Certify the object by authority	Accepted	Acceptance recorded by an authority
5	Operationalise	Make knowledge actionable	Operational (actionable)	Actionable form linked to justification
6	Govern	Maintain legitimacy over time	Governed	Ownership and review in force
7	Apply	Use the object in practice	(in use) → Outcomes	Outcomes captured as feedback
8	Evolve	Allow controlled improvement	Next Version	Change justified and versioned
9	Retire	Conclude operational life	Historical	Reason recorded, history preserved

The Three Dimensions of Every Transition

A workflow advances one property: a status. This lifecycle advances three at once. Every transition changes not only the **state** of the object, but the **evidence** attached to it and the **governance posture** it has achieved. These three dimensions are independent — they answer different questions — and reading the lifecycle as only a sequence of states hides two-thirds of what each transition does.

- **State** — what the object *is*: the label that says how far it has progressed.
- **Evidence** — what is *attached* to the object: the accumulating record that travels with it.

- **Governance posture** — what is *assured* about the object: the governance property each step secures.

Lifecycle step	State (what it is)	Evidence (what is attached)	Governance posture (what is assured)
Capture	Candidate	Origin	Identity
Structure	Structured	+ metadata, relationships	Inspectability
Evaluate	Evaluated	+ quality assessment	Quality
Justify	Justified	+ justification	Warrant
Accept	**Accepted**	**+ acceptance record**	**Authority**
Operationalise	Operational	+ actionable form	Operability
Govern	Governed	+ review history	Ownership
Apply	*(remains Governed)*	+ operational outcomes	Accountability in use
Evolve	Next Version	+ change record	Reversibility
Retire	Historical	+ retirement record	History

Two features of this table matter. First, the three columns advance at different times: Apply adds evidence and governance assurance without changing the object's *state*, which is precisely why application is a different kind of thing from the construction stages. Second, each column can be read on its own as a coherent progression — a reviewer auditing *evidence* need not trace *state*, and a reviewer auditing *governance* need not trace either. The lifecycle is richer than a pipeline because it is really three pipelines advancing on one object.

The Nine Stages

Stage 1 — Capture

Purpose. Admit candidate information into the system. Capture is deliberately permissive: the aim is to bring material in, not yet to judge it. Nothing captured is trusted, and nothing captured is knowledge; everything enters as a candidate whose status must be earned downstream.

The one obligation that cannot be deferred at capture is **origin**. Provenance can be recorded now but rarely reconstructed later: the moment a source is detached from its content, the object's history is compromised for the rest of its life. Capture therefore records *where the material came from* as a condition of entry, even while deferring every question of quality and warrant.

Typical inputs. Documents; research papers; interviews; AI outputs; meeting notes; regulations; datasets; observations.

Entry condition. Material is presented to the system. **Activities.** Intake; source identification; provenance seeding; assignment of a candidate identifier. **Exit gate.** The candidate is recorded together with its origin. **Output state.** Candidate. **Accountable role.** Contributor. **Key question.** *What has entered the system?* **Canon basis.** Provenance invariant; Principle I (the object, not knowledge, is what enters).

Stage 2 — Structure

Purpose. Transform raw, opaque input into a structured object that can be reasoned about. Structuring is what makes everything downstream possible: an object that cannot be inspected cannot be evaluated, justified, explained or governed. The stage confers a stable identity and an inspectable form.

Entry condition. A Candidate with recorded origin. **Activities.** Decomposition; metadata assignment; semantic typing; relationship identification; provenance recording. **Exit gate.** The object has a stable identity, a semantic type, its relationships, and a provenance record — that is, it is inspectable. **Output state.** Structured. **Accountable role.** Knowledge Engineer. **Key question.** *What exactly is this object?* **Canon basis.** Inspectability invariant; Principle I.

Stage 3 — Evaluate

Purpose. Assess quality before any question of belief is raised. Evaluation asks whether the object is *good enough to be worth justifying* — a different and prior question to *why it should be believed*. It is the first gate that can stop progression: an object may pass, be returned for restructuring, or be rejected outright.

Entry condition. A Structured object. **Activities.** Source assessment; evidence-quality assessment; confidence estimation; consistency and completeness checks. **Exit gate.** A quality verdict — pass, return, or reject — is recorded with its reasons. **Output state.** Evaluated. **Accountable role.** Evaluator. **Key question.** *Should this object be trusted?* **Canon basis.** Principle IV (the Validation/Quality primitive).

Stage 4 — Justify

Purpose. Establish epistemic legitimacy. This is the epistemic core of the lifecycle: the stage at which an object earns the standing to be called knowledge. Justification attaches the reasons for belief to the object and makes them inspectable. Counter-evidence is recorded as a matter of obligation, not option — an object justified only by the evidence that supports it has not been justified, it has been advocated.

A system can engineer justification even though it cannot engineer truth; justification, not truth, is the standard every transition is held to.

Entry condition. An Evaluated object that passed. **Activities.** Assembly of supporting evidence; assembly of counter-evidence; statement of rationale; statement of assumptions; end-to-end traceability. **Exit gate.** The object carries an inspectable justification: supporting and counter-evidence, rationale, assumptions, and traceability to source. **Output state.** Justified. **Accountable role.** Evaluator (with domain authority as required). **Key question.** *Why should this be believed?* **Canon basis.** Principle VIII; Principle IV; justification invariant.

The Acceptance Transition (Justify → Operationalise)

Acceptance is the act by which an identified authority certifies a justified object as fit to be relied upon. It is the single most consequential moment in the lifecycle, and it is deliberately **not** a stage: nothing is transformed, no work is performed on the object. It is a *decision about* the object — the point at which human authority admits a justified representation into operational use. For that reason it is modelled as a named transition, a gate, rather than an engineering stage. This mirrors the treatment of Application below: both are things that happen to a governed object without being engineering transformations of it, and naming them honestly keeps the nine stages purely about work performed on the object.

Acceptance is where the object acquires the **acceptance-state** that the Canon requires of a full Operational Knowledge Object (KF-1001). Until it is accepted, an object may be perfectly justified and still have no standing to be relied upon; justification establishes *why it may be believed*, acceptance establishes *that we will act on it*. The two are distinct, and conflating them is the failure this transition exists to prevent.

Who acts. An Authority — an identified person or body with the right to certify knowledge for the community that will use it. **Requires.** A Justified object with inspectable justification. **Produces.** An acceptance-state recorded against the object — accepted, provisionally accepted, or refused — attributed to the authority and dated. On acceptance the object proceeds to Operationalise; on refusal it returns to Justify or Evaluate with reasons. **Key question.** *On whose authority will we rely on this?* **Canon basis.** Principle III (knowledge becomes capability through collective authorisation); Principle IV (the Authority primitive); human-accountability invariant.

Stage 5 — Operationalise

Purpose. Make accepted knowledge usable. Operationalisation converts an accepted object into an *actionable form* — the shape in which it can be applied in decisions, reasoning or automation. The binding constraint is that the actionable form must remain linked to its justification. A playbook, rule or prompt that has been severed from the evidence and reasoning that warrant it is ungoverned knowledge in use, and is the failure mode this stage exists to prevent.

Entry condition. An Accepted object (see *The Acceptance Transition* above). **Activities.** Derivation of decision rules, playbooks, patterns, prompts, procedures or models; linkage of each to the underlying justification. **Exit gate.** An actionable form exists and is traceably linked to its justification. **Output state.** Operational (actionable). **Accountable role.** Knowledge Engineer. **Key question.** *How can this knowledge be used?* **Canon basis.** Principle II (lifecycle); Principle I.

Stage 6 — Govern

Purpose. Maintain legitimacy through time. Acceptance is not permanent, and Govern is the standing machinery that keeps an operational object legitimate as its context changes: ownership, review cycles, approvals, versioning, provenance upkeep and policy compliance.

Govern is a stage, but governance is not confined to it. The Governance Primitives operate throughout the lifecycle — provenance from Capture, validation at Evaluate, justification at Justify, authority at Accept. Stage 6 does not introduce governance; it makes it *continuous*, and it names the person or body answerable for the object from this point forward.

Entry condition. An Operational object. **Activities.** Assignment of ownership; scheduling of review cycles; approval workflows; version control; provenance maintenance; policy-compliance checks. **Exit gate.** Ownership is assigned and a review cycle is in force. (This gate is standing, not one-off: it must remain satisfied for the object to stay operational.) **Output state.** Governed. **Accountable role.** Steward (owner). **Key question.** *Who is accountable?* **Canon basis.** Principle IV; Principle VII; human-accountability invariant.

Stage 7 — Apply

Purpose. Use the governed object in the practice it was built to serve. Application is different in kind from every stage before it: it does not transform the object, it *uses* it. Its engineering value is that it is the lifecycle's principal source of new evidence — the outcomes of use are the truest test of whether the knowledge is fit — and it is therefore the principal trigger for evolution. The lifecycle's working rhythm is *use, observe, improve*, and Apply is the "use" and "observe" of that rhythm.

Because application does not change the object's state, an applied object remains Governed; what changes is the evidence attached to it (operational outcomes) and what is assured about it (its accountability in use). Outcomes feed forward into Evolve, or, where use shows the object is no longer needed, into Retire.

Entry condition. A Governed object. **Activities.** Use in human decisions, AI reasoning, automation and organisational workflows; use in teaching and advisory work; capture of outcomes. **Exit gate.** Outcomes are captured and attached to the object as feedback. **Output state.** Operational Outcomes (feedback into Evolve or Retire); the object remains Governed. **Accountable role.** User. **Key question.** *Did the knowledge produce value?* **Canon basis.** Principle IX (knowledge must remain explainable in use); Principle III.

Stage 8 — Evolve

Purpose. Allow controlled improvement. Change is the default condition of knowledge, and Evolve is where change is performed as a first-class, justified operation rather than an untracked edit. Every change produces a new version; the prior version is preserved, never overwritten. A change is itself a transition, and so carries its own justification and provenance.

Evolution is triggered principally by the outcomes of Application, and also by new external evidence, competing claims or a scheduled review — so an object may enter Evolve from Apply or directly from Govern. It takes several forms: amendment in the light of new evidence; adjudication of competing claims; supersession of one object by another; branching for

divergent contexts; and consolidation of several objects into one. Each returns the object to the appropriate earlier stage for re-evaluation, re-justification and re-acceptance before its new version can be relied upon.

Entry condition. A Governed object with a change trigger (application outcomes, new evidence, a competing claim, or a scheduled review). **Activities.** Amendment; incorporation of new evidence; adjudication of competing claims; supersession; branching; consolidation. **Exit gate.** A new version is produced under recorded justification and provenance, with the prior version retained. **Output state.** Next Version (which re-enters the lifecycle at Evaluate or Justify, and is re-accepted before returning to operational use). **Accountable role.** Steward, with the Evaluator and Authority as required. **Key question.** *What has changed?* **Canon basis.** Principle X; Principle II; governed-evolution invariant.

Stage 9 — Retire

Purpose. Conclude the operational life of the object. Retirement is a change of state, not a deletion: the object leaves active use but its record — content, justification, provenance and history — is preserved. An object is retired because it is obsolete, superseded, merged, archived or withdrawn; the reason is recorded, because the fact of retirement is itself part of the knowledge history a later reader may need.

Retirement is reversible in principle. If evidence changes such that a retired object regains relevance — for example, a superseding object is itself withdrawn — the historical object can be reactivated through Evolve, with its provenance intact.

Entry condition. A retirement reason holds for a Governed or applied object. **Reasons.** Obsolete; superseded; merged; archived; withdrawn. **Exit gate.** The object is moved to the historical state, its retirement reason recorded and its history preserved. **Output state.** Historical. **Accountable role.** Steward. **Key question.** *Why is this object no longer active?* **Canon basis.** Principle VII (trust across time); Principle X.

Transitions and Return Paths

The principal path runs Capture → Retire, but the lifecycle is governed as a loop. Three kinds of non-forward transition are canonical.

Returns send an object back to an earlier stage when a gate cannot be met. Evaluate may return an object to Structure or Capture when it is ill-formed or its origin is inadequate. Justify may return an object to Evaluate when its justification fails. **Acceptance may be refused**, returning a justified object to Justify or Evaluate with the authority's reasons — a refusal is as much a governed transition as an acceptance, and is recorded as one.

Evolution re-entry. Evolve re-enters the lifecycle at Evaluate or Justify, and its new version passes through the Acceptance transition again before returning to operational use. A new version re-earns its warrant and its authority; it does not inherit them.

Retirements may be reached from Govern, Apply or Evolve. An object that becomes obsolete, superseded or withdrawn passes directly to Retire without completing the forward path.

Every transition — forward, return, acceptance, refusal or retirement — is itself a governed act: it is recorded, attributed and justified. There are no silent transitions. This is what distinguishes the lifecycle from a workflow in which status simply advances: here, movement between states is the very thing being governed.

Engineering Events

States and transitions describe *what* the lifecycle does. Events describe *what causes it to act*. The lifecycle is, formally, a state machine with three distinct kinds of element:

- **States** — what an object is at a given moment (Candidate, Structured, Justified, Accepted, Governed, Historical …).
- **Transitions** — the governed moves between states (Accept, Return, Evolve, Retire …).
- **Events** — the occurrences that trigger a transition (new evidence, a scheduled review, a challenge …).

Events cause transitions; transitions produce new states:

```
State
   |
Engineering Event   (what happened)
   |
Transition          (the governed move it triggers)
   |
New State
```

Separating events from transitions matters for two reasons. First, it makes the model a clean state machine — the form over which later automated validation and orchestration operate. Second, it locates where automation is safe: an **event may be raised by a human or by an agent**, but the **transition it triggers remains governed** — attributable, justified, and, where authority is required, certified by a human. An AI recommendation is an event, not an acceptance; automation may propose, but the invariants still hold on whatever transition follows.

The canonical event types, and the transitions they typically trigger:

Engineering event	Typically triggers
New evidence	Evolve (or a Return to Justify)
Scheduled review	A governance review, then Evolve or Retire
Policy or regulatory change	Re-validation, then Evolve or Retire
Challenge to a claim	Return to Justify or Evaluate
AI recommendation	Capture of a new candidate, or an Evolve proposal
User feedback / application outcome	Evolve or Retire
Authority decision	The Acceptance transition (accept or refuse)

An event does not itself change an object; it obliges the system to perform — and govern — a transition. No event bypasses a gate.

Roles and Accountability

The lifecycle defines role *responsibilities*, not posts or systems; how they are staffed is an implementation concern (KF-4000). One person may hold several roles; a role may be held by several people. What the method fixes is that every stage has an accountable role, and that acceptance and stewardship are always held by an identifiable human authority — labour may be distributed across people and machines, but accountability may not be discharged into a substrate.

Role	Principal stages	Accountable for
Contributor	Capture	What enters the system, and its recorded origin
Knowledge Engineer	Structure, Operationalise	The object's form and its actionable expression
Evaluator	Evaluate, Justify	The quality verdict and the inspectable justification
Authority	the Acceptance transition	Certifying an object as fit to be relied upon
Steward (owner)	Govern, Evolve, Retire	The object's standing legitimacy over time
User	Apply	Using the object as governed, and returning outcomes

Lifecycle Invariants

The lifecycle principles below describe how the lifecycle *behaves*. The invariants here describe what must be true of an object *at every instant, regardless of stage*. An invariant is a continuously checkable property: an object that violates any invariant is non-conformant, whatever state it is in and however it reached it. These are the properties automated validation will assert, and the ground on which the Reference Architecture (KF-3000) is designed.

Six invariants hold across the whole lifecycle.

1. **Single identity.** Every Operational Knowledge Object has exactly one immutable identity for its entire life. Identity is assigned at Capture and is never reassigned, reused or mutated. Editing produces new *versions* under the same identity; branching, consolidation and supersession create *successor* objects with their own identities, linked to their predecessors. They do not split, merge or overwrite an existing identity. (Identity, version and lineage are distinct concerns, and KF-3000 realises them separately.)
2. **Attributable transitions.** Every state transition is attributable to an identified actor — a person, or an agent acting under a named human authority. No transition is anonymous.

3. **Justified transitions.** Every state transition carries a recorded justification. A transition without a recorded reason is not permitted, even when it is automatic.
4. **Inspectable transitions.** Every state transition is inspectable after the fact: what changed, why, by whom, and from which prior state can always be recovered.
5. **Provenance always present.** No operational state exists without a complete provenance chain back to origin. Provenance is a precondition of operational use, not an annotation added afterward.
6. **Lineage preserved.** Retirement, supersession and consolidation never destroy lineage. History is append-only; states are superseded, never erased.

Where an invariant appears to echo a principle — justification, for instance — the two do different work. The principle is the behavioural rule that a transition must *earn* its justification in order to proceed; the invariant is the checkable property that, once made, the transition *carries* that justification for all time. Principles govern progression; invariants govern conformance.

Invariants are ungraded. These six hold identically for every object, whatever its consequence or intended reliance. What *does* vary with reliance is the *depth* to which a gate's obligations must be discharged — the strength of evaluation, the independence of review, the sufficiency of evidence. That depth is set by the object's **assurance class**, defined in KF-2003 and attaching to the object's scope of reliance at acceptance and release. An Exploratory object and a Consequential one cross the same gates and satisfy the same six invariants; the Consequential object must satisfy each gate to a greater depth. No assurance class ever lowers the invariant floor.

Lifecycle Principles

Beyond the invariants above, the lifecycle is governed by six behavioural rules, each a direct consequence of the Canon.

1. **Knowledge always begins as a candidate.** Nothing is trusted on entry; status is earned stage by stage. (Principle I.)
2. **Every transition requires justification.** A gate is a justification checkpoint; an object advances only when it has earned the warrant of its stage. (Principle VIII.)
3. **Acceptance is an act of authority, distinct from justification.** Justification establishes why an object may be believed; acceptance establishes that an identified authority will rely on it. No object reaches operational use without both. (Principle III.)
4. **Governance accompanies every stage.** Govern is where legitimacy is *maintained*, but the Governance Primitives operate from Capture onward. Governance is a property of the whole lifecycle, not a single step within it. (Principles IV and VI.)
5. **Knowledge may return to earlier stages as evidence changes.** The lifecycle is a loop; returns, refusals and re-justification are normal operations, not exceptions. (Principle X.)
6. **Retirement preserves history rather than erasing it.** An object leaves active use with its record intact, because trust is a property of transitions over time. (Principle VII.)

Relationship to the Canon

Each lifecycle stage operationalises one or more Canonical Principles: the per-stage *Canon basis* lines above indicate the principal derivation. The lifecycle as a whole is the method by which the Canon's central operation — the controlled transition of a knowledge object from one justified state to another — is carried out in practice. The authoritative, complete stage-to-principle mapping is defined in **KF-2002** and is not reproduced here, so that a change to the mapping has a single home.

Relationship to the Reference Architecture

Each lifecycle stage is realised through one or more architectural services — for capture, structuring, evaluation, justification, acceptance, governance, versioning and retirement. This document fixes *what each stage must achieve and when it may advance*; it does not specify the services, stores or technologies that achieve it. That mapping is defined in **KF-3000**, which a given deployment (KF-4000, and domain instantiations such as MacroLab) then implements. KF-3000 must also enforce the lifecycle invariants as continuous conformance checks, and handle the engineering events as the triggers its services respond to.

KF-2002 · Engineering Practices

Operationalising the Canon through Engineering Practice

Document Class: Methodology · Parent Programme: Knowledge Foundry · Parent Document: KF-2000 · Status: Draft · Version: 0.1

```
Principle (KF-1000)
        ↓
Lifecycle (KF-2001)
        ↓
▸ Engineering Practices (KF-2002)
        ↓
Reference Architecture (KF-3000)
        ↓
Implementation (KF-4000)
```

Purpose

KF-2001 defines the lifecycle through which an Operational Knowledge Object evolves. KF-2002 defines the engineering practices that enable that lifecycle.

A lifecycle answers *what happens*. An engineering practice answers *how that responsibility is discharged*. Each practice operationalises one or more Canonical Principles while supporting one or more lifecycle stages, and each is specified as a set of obligations rather than a prescribed procedure. Consistent with KF-2001, this is method, not workflow: a practice fixes what must be true and what must be produced, not the order, tools or cadence by which a given organisation achieves it. Those belong to implementation (KF-4000), and the services that realise these practices belong to architecture (KF-3000). No technology choices are made here.

Relationship to KF-2001

The lifecycle defines engineering *states*; engineering practices define engineering *behaviour*. The relationship is many-to-many. Multiple practices may operate within a single lifecycle stage, and a single practice may span several stages. The lifecycle therefore specifies *when* engineering responsibilities arise; the practices specify *how* they are fulfilled.

The two documents share one spine. A lifecycle **gate** is satisfied by executing the practice that supports it; a lifecycle **invariant** is upheld by the obligations the practices carry; a lifecycle **event** is what calls a practice into action. Where KF-2001 states that a transition must be attributable, justified and inspectable, it is the practices below that produce the attribution, the justification and the inspectable record. The practices are, in effect, how the lifecycle keeps its promises.

The Engineering Practice Specification

Every practice is specified against the same nine fields, so that practices can be compared, audited and — later — validated uniformly. Immediately below each practice's title, a single line names the **accountable role** — the KF-2001 role answerable for the practice's discharge.

Accountability cannot be discharged into a computational substrate, so the answerable role is made visible on every practice rather than left to be inferred.

- **Canon principle** — the Canonical Principle(s) the practice operationalises.
- **Objective** — the purpose of the practice.
- **Engineering obligation** — what must always be true wherever the practice applies (an obligation is invariant-like: it holds at every instant, not only at a gate).
- **Supported lifecycle stages** — the KF-2001 stages or transitions that depend on the practice, distinguishing the practice's principal home from the stages it additionally supports.
- **Typical activities** — the engineering work normally performed.
- **Artefacts produced** — the mandatory outputs the practice must leave behind.
- **Governance checkpoints** — how compliance is verified, expressed as checks against gates, invariants and Governance Primitives.
- **Success criteria** — how success is recognised.
- **Failure modes** — the characteristic failures when the practice is absent or poorly executed.

On artefacts. *Engineering artefact* means an inspectable record produced or maintained by a practice as evidence that an obligation has been discharged. An artefact may be represented in several forms, but its required semantic content is fixed by the practice, not by any file, form or database row that carries it. This keeps KF-3000 free to choose representations while remaining bound to the content each artefact must hold.

On success criteria. Success criteria state the required condition. Quantitative thresholds, sampling rules and conformance measures are graded by the object's assurance class (**KF-2003 · Materiality and Assurance**) and may be further specified by a domain governance profile or implementation, not here — so the criteria below are written to be *observable* (a reviewer can confirm them) without prematurely fixing metrics or the depth at which each is required.

Practice Domains at a Glance

Domain	Focus	Practices
A	Identity & Representation	A1, A2
B	Evidence & Justification	B1, B2
C	Operational Governance	C1, C2, C3
D	Operational Use	D1, D2, D3

The four domains follow the arc of the lifecycle: A gives the object identity and form; B earns it justified trust; C governs its acceptance, custody and change; D puts it to use and concludes its life. Together the ten practices provide one principal practice for each of the nine lifecycle stages and the Acceptance transition; several also support neighbouring stages, as the crosswalk records.

Part I · Domain A — Identity & Representation

Purpose: ensure every Operational Knowledge Object possesses a stable identity, explicit structure and inspectable representation.

A1 · Operational Knowledge Object Identification

Accountable role. Contributor. **Canon principle.** Principle I. **Objective.** Assign and maintain the identity through which an object is known for its entire life, and bind its origin to it from the first moment. **Engineering obligation.** Every Operational Knowledge Object has one immutable identity. Editing produces new versions under the same identity; branching, consolidation and supersession create successor relationships — they do not erase, reuse or mutate the identities of the objects involved. Concretely: edits create versions under the same object identity; branching creates new object identities linked to a common predecessor; consolidation creates a new successor identity linked to all source objects; supersession changes status and lineage, not identity. No object exists without a recorded origin. (Lifecycle invariants: single identity; provenance always present. Identity, version and lineage are distinct concerns, separated in KF-3000.) **Supported lifecycle stages.** Capture (principal home); Structure (support). **Typical activities.** Intake registration; assignment of a durable identifier; recording of source and origin; disambiguation of a new object from existing ones (is this novel, a new version, a branch, or a consolidation of things already held?); carrying identity forward into structuring. **Artefacts produced.** An identity record (durable identifier bound to origin, distinct from version and lineage records); a provenance seed that later stages extend. **Governance checkpoints.** Uniqueness check — the identity resolves to exactly one object and no other object claims it; immutability check — no operation reassigns, reuses or mutates an existing identity; origin-recorded check — the Capture exit gate is met before the object advances. Exercises the Provenance/Traceability primitive. **Success criteria.** Every object resolves to exactly one identity with a recorded origin; a reviewer can retrieve any object by its identifier and trace it to its source; no two objects share an identifier, and no identifier has been reassigned across the object's life. **Failure modes.** Two identities for one object, or one identifier reused for two; identity reassigned or mutated on edit rather than versioned, silently breaking lineage; branching or consolidation that overwrites a predecessor's identity instead of creating a linked successor; origin unrecorded and later unreconstructable; a candidate admitted with no identifier at all.

A2 · Knowledge Structuring

Accountable role. Knowledge Engineer. **Canon principle.** Principles I and II. **Objective.** Transform raw captured material into an inspectable, structured object that downstream stages can reason about. **Engineering obligation.** The object is inspectable: its structure, semantic type and relationships are explicit and examinable, not implicit in unstructured content — and this holds identically whatever the computational modality of the underlying representation, symbolic or parametric (thereby also discharging Principle V within this practice). (Lifecycle invariant: inspectability, as the Structure exit gate.) **Supported lifecycle stages.** Structure (principal home). **Typical activities.** Decomposition into constituent claims; semantic typing; metadata assignment; identification of relationships to other objects; recording of provenance. **Artefacts produced.** A structured representation; a metadata record; a relationship map; an extended provenance record. **Governance checkpoints.** Inspectability

check — the object can be examined, typed and related; structure-complete check — the Structure exit gate (identity, type, relationships, provenance) is met. Exercises the Provenance/Traceability primitive. **Success criteria.** A reviewer can identify the object's type, constituent claims, relationships and provenance without reconstructing them from unstructured source material. **Failure modes.** Opaque objects that cannot be evaluated or justified downstream; structure left implicit in prose or weights; missing relationships that hide dependency; representations that resist inspection and therefore resist governance.

Part II · Domain B — Evidence & Justification

Purpose: ensure every knowledge object earns justified trust through explicit evaluation and inspectable reasoning.

B1 · Evidence Evaluation

Accountable role. Evaluator. **Canon principle.** Principle IV. **Objective.** Assess the quality of a structured object before any question of belief is raised — decide whether it is good enough to be worth justifying. **Engineering obligation.** No object advances toward justification without a recorded quality verdict and its reasons; evaluation is kept distinct from justification. (Evaluate exit gate; the distinction B1↛B2 is load-bearing.) **Supported lifecycle stages.** Evaluate (principal home). **Typical activities.** Source assessment; grading of evidence quality; confidence estimation; consistency and completeness checks; recording of a pass, return or reject verdict with reasons. **Artefacts produced.** A quality-assessment record: the verdict, the criteria applied, and the reasons. **Governance checkpoints.** Quality-verdict check — the Evaluate exit gate is met and the verdict is recorded and inspectable. Exercises the Validation/Quality primitive. **Success criteria.** For any object that advanced, a reviewer can retrieve the recorded verdict, the criteria applied and the reasons; objects that were returned or rejected carry the same record and did not proceed. **Failure modes.** Evaluation conflated with justification (asking *why believe it* before asking *is it worth the effort*); low-quality objects passed through; verdicts reached but not recorded; evaluation that advocates for an object rather than assessing it.

B2 · Knowledge Justification

Accountable role. Evaluator, with domain authority where required. **Canon principle.** Principles IV and VIII. **Objective.** Establish epistemic legitimacy by attaching an inspectable justification — supporting evidence, counter-evidence, rationale, assumptions and traceability — to the object. **Engineering obligation.** Every object that will enter operational use carries an inspectable justification, and counter-evidence is recorded as an obligation, not an option. Justification, not truth, is the standard, and it is required even where knowledge resides in parametric form, whose warrant must be reconstructed rather than read off an inference (again holding the obligation across modality, per Principle V). (Justify exit gate; lifecycle invariant: justified transitions; Principle VIII.) **Supported lifecycle stages.** Justify (principal home). **Typical activities.** Assembly of supporting and counter-evidence; statement of rationale; explicit statement of assumptions; construction of end-to-end traceability from claim to source. **Artefacts produced.** A justification record: supporting and counter-evidence,

rationale, assumptions, and a traceability chain. **Governance checkpoints.** Justification-inspectable check — any reviewer can recover why the object may be believed; the Justify exit gate is met. Exercises the Evidence/Justification primitive. **Success criteria.** An independent reviewer can trace each material claim to its supporting and defeating evidence, recover the stated assumptions, and reproduce the recorded rationale. **Failure modes.** Advocacy — only confirming evidence recorded; justification that cannot be inspected; truth conflated with justification; assumptions left hidden, so the object's warrant cannot be tested.

Part III · Domain C — Operational Governance

Purpose: maintain legitimacy, accountability and controlled evolution throughout the operational life of knowledge.

C1 · Knowledge Acceptance

Accountable role. Authority. **Canon principle.** Principle III. **Objective.** Have an identified authority certify a justified object as fit to be relied upon, producing the acceptance-state that admits it into operational use. **Engineering obligation.** No object reaches operational use without acceptance by an identified human authority, and every acceptance is attributable and recorded. Acceptance is distinct from justification: justification establishes *why an object may be believed*; acceptance establishes *that an authority will rely on it*. (Acceptance transition; lifecycle invariant: attributable transitions; the human-accountability invariant of the Canon. This practice also exercises the Authority primitive of Principle IV.) **Supported lifecycle stages.** The Acceptance transition, Justify → Operationalise (principal home). **Typical activities.** Review of the justification by the authority; decision to accept, provisionally accept or refuse; recording of the acceptance-state, attributed and dated, with the scope of reliance; on refusal, return to Justify or Evaluate with reasons. **Artefacts produced.** An acceptance record (decision, authority, date, scope); a refusal record with reasons where applicable. **Governance checkpoints.** Authority-certified check — the object bears an acceptance attributed to a named authority; attribution check — the transition is not anonymous. Exercises the Authority primitive. **Success criteria.** For any operational object, a reviewer can identify the accepting authority, the date, the scope of reliance and the acceptance-state; refusals carry the authority's recorded reasons. **Failure modes.** Acceptance conflated with justification; anonymous or implicit acceptance; acceptance discharged to a machine so that no human is answerable; fluent or plausible output relied upon with no acceptance at all.

C2 · Governance & Stewardship

Accountable role. Steward. **Canon principle.** Principles IV and VII. **Objective.** Keep an operational object legitimate as its context changes, through ownership, review, versioning, provenance upkeep and policy compliance. **Engineering obligation.** Every operational object has an identified steward and a review cycle in force, and its provenance is maintained for as long as it is operational. Governance is continuous, not a one-time step. (Govern standing gate; lifecycle invariant: provenance always present; Principle VII: trust across transitions over time.) **Supported lifecycle stages.** Govern (principal home); the whole operational life (standing support). **Typical activities.** Assignment of ownership; scheduling and running of

review cycles; approval workflows; version control; provenance maintenance; policy-compliance checks; response to scheduled-review and policy-change events. **Artefacts produced.** A stewardship assignment; a review schedule and its record; a governance and version log; a policy-compliance record. **Governance checkpoints.** Ownership-assigned and review-in-force check — the standing Govern gate remains satisfied; provenance-maintained check. Exercises the Versioning/Release and Provenance/Traceability primitives. **Success criteria.** For any operational object, a reviewer can identify its current steward, its review schedule and last review, its version history and its compliance status; none of these is absent. **Failure modes.** Ownerless operational knowledge; objects that drift out of review and go stale; provenance decay over time; governance that is recorded at acceptance and then never exercised — legitimacy on paper only.

C3 · Knowledge Evolution

Accountable role. Steward. **Canon principle.** Principles II and X. **Objective.** Perform change as a first-class, justified, versioned operation that improves the object while preserving its history and without leaving dependent knowledge silently inconsistent. **Engineering obligation.** Every change produces a new version under recorded justification and provenance; the prior version is retained; the new version re-earns its warrant by returning through evaluation, justification and acceptance; and the objects that depend on the changed one are identified and dispositioned. (Evolve gate; lifecycle invariants: lineage preserved, justified transitions; the governed-evolution invariant.) **Supported lifecycle stages.** Evolve (principal home); Evaluate, Justify and the Acceptance transition (support, via re-routing of the new version). **Typical activities.** Amendment in the light of new evidence; adjudication of competing claims; supersession; branching for divergent contexts; consolidation; impact analysis across dependent objects and actionable forms; routing the new version back to Evaluate/Justify and through re-acceptance. Evolution is called by events — new evidence, a challenge, a scheduled review, an application outcome. **Artefacts produced.** A new version; a change record (trigger, justification, provenance); lineage links for supersession, branch and merge; a dependency-impact record identifying objects, practices or applications requiring re-evaluation. **Governance checkpoints.** Change-justified-and-versioned check — the Evolve gate is met; prior-version-retained check; re-acceptance check — the new version passes the Acceptance transition before returning to use; dependency-impact check — affected dependencies have been identified and either revalidated, suspended or routed into Evolution. Exercises the Versioning/Release and Reversibility/Audit primitives. **Success criteria.** For any change, a reviewer can identify the trigger, the justification, the resulting version, the retained prior version, the affected dependencies and their disposition, and the re-acceptance of the new version before its return to use. **Failure modes.** Untracked edits; prior versions overwritten; evolution without re-justification or re-acceptance; lineage lost when objects are superseded, branched or merged; a changed object correctly versioned while its dependants are left silently inconsistent.

Part IV · Domain D — Operational Use

Purpose: ensure knowledge creates value while preserving explainability, feedback and historical integrity.

D1 · Operationalisation

Accountable role. Knowledge Engineer. **Canon principle.** Principles II and III. **Objective.** Convert an accepted object into an actionable form — the shape in which it can be used in decisions, reasoning or automation. **Engineering obligation.** The actionable form is traceably linked to the justification and acceptance beneath it; use never detaches from warrant. Only accepted objects are operationalised. (Operationalise exit gate.) **Supported lifecycle stages.** Operationalise (principal home). **Typical activities.** Derivation of decision rules, playbooks, patterns, prompts, procedures or models from the accepted object; linkage of each actionable form to the underlying justification and acceptance. **Artefacts produced.** One or more actionable forms; a linkage record binding each to its justification and acceptance. **Governance checkpoints.** Form-linked-to-justification check — the Operationalise exit gate is met and every actionable form traces back to warrant. Exercises the Versioning/Release primitive. **Success criteria.** For any actionable form, a reviewer can trace it to the object version, justification and acceptance-state from which it derives. **Failure modes.** Actionable forms severed from their justification — ungoverned knowledge in use; operationalisation of objects that were never accepted; forms that cannot be traced back to a source object.

D2 · Knowledge Application

Accountable role. User, under Steward oversight. **Canon principle.** Principle IX. **Objective.** Use governed knowledge in practice while preserving explainability, and capture the outcomes of use as new evidence. **Engineering obligation.** Knowledge remains explainable in use whatever the modality in which it resides, and the outcomes of application are captured and attached to the object. Application uses the object without transforming it; the object remains Governed. (Apply exit gate; Principle IX; explainability is owed by the system, not the substrate — and is therefore preserved across modality, per Principle V.) **Supported lifecycle stages.** Apply (principal home); Evolve and Retire (support, which application's outcome events call). **Typical activities.** Use in human decisions, AI reasoning, automation and organisational workflows; use in teaching and advisory work; capture of outcomes; raising of feedback events that call Evolution or Retirement. **Artefacts produced.** An outcome record attached to the object, feeding the Evidence dimension and the evolution/retirement decision. **Governance checkpoints.** Explainability-in-use check — an account of how the knowledge produced a given result can be given, regardless of substrate; outcome-captured check — the Apply exit gate is met. Exercises the Reversibility/Audit primitive. **Success criteria.** For every material application outcome, the system can identify the object version used, its actionable form, the governing acceptance-state and the recorded path from input to result. **Failure modes.** Black-box application that cannot be accounted for; outcomes not captured, breaking the feedback loop; application of ungoverned or unaccepted objects; use that has quietly detached from the justification it was meant to carry.

D3 · Knowledge Retirement

Accountable role. Steward or designated Authority. **Canon principle.** Principles VII and X. **Objective.** Conclude the operational life of an object without destroying its lineage, recording why it is no longer active. **Engineering obligation.** Retirement is a change of state, not a deletion; lineage is preserved; the reason is recorded; and retirement is reversible in principle. History is append-only. (Retire gate; lifecycle invariant: lineage preserved; Principle VII.)

Supported lifecycle stages. Retire (principal home). **Typical activities.** Determination of the retirement reason (obsolete, superseded, merged, archived, withdrawn); transition of the object to the historical state; recording of the reason; preservation of the full record; provision for reactivation through Evolution should evidence change. **Artefacts produced.** A retirement record (reason, date, authority); the preserved historical object with its lineage intact. **Governance checkpoints.** Reason-recorded and history-preserved check — the Retire gate is met; lineage-intact check — no history is erased and superseding links remain traceable. Exercises the Reversibility/Audit primitive. **Success criteria.** For any retired object, a reviewer can identify the retirement reason, date and authority, retrieve its full historical record and lineage, and determine the successor object where one exists. **Failure modes.** Deletion that destroys lineage; retirement with no recorded reason; silent removal; a superseding object that breaks the link back to the object it replaced.

Crosswalk

Each practice maps to the Canonical Principle(s) it operationalises, the role accountable for it, its principal lifecycle home, and the stages it additionally supports.

Practice	Canon	Accountable role	Principal home	Additional support
A1 · OKO Identification	Principle I	Contributor	Capture	Structure
A2 · Knowledge Structuring	Principles I, II	Knowledge Engineer	Structure	—
B1 · Evidence Evaluation	Principle IV	Evaluator	Evaluate	—
B2 · Knowledge Justification	Principles IV, VIII	Evaluator (+ domain authority)	Justify	—
C1 · Knowledge Acceptance	Principle III	Authority	Acceptance transition	—
C2 · Governance & Stewardship	Principles IV, VII	Steward	Govern	Operational life
C3 · Knowledge Evolution	Principles II, X	Steward	Evolve	Evaluate, Justify, Acceptance
D1 · Operationalisation	Principles II, III	Knowledge Engineer	Operationalise	—
D2 · Knowledge Application	Principle IX	User (under Steward)	Apply	Evolve, Retire
D3 · Knowledge Retirement	Principles VII, X	Steward / Authority	Retire	—

Constitutional coverage. The crosswalk's Canon column records each practice's *principal* principle. Because two principles are cross-cutting rather than stage-bound, coverage of all ten is stated explicitly here:

- **Principle I** — A1, A2 (the object, and its inspectable representation)
- **Principle II** — A2, C3, D1 (engineering as a lifecycle)
- **Principle III** — C1, D1, D2 (collective authorisation and accountable use)
- **Principle IV** — B1, B2, C1, C2 (governance as constitutive; the six primitives)
- **Principle V** — A2, B2, D2 (the obligation preserved across every computational modality)
- **Principle VI** — the methodology as a whole: the lifecycle (KF-2001) and these practices coordinate engineering, capability and governance on a single object; no one practice owns it, and the crosswalk itself is its evidence
- **Principle VII** — C2, D3 (trust preserved across transitions over time)
- **Principle VIII** — B2 (inspectable justification for every transition)
- **Principle IX** — D2 (explainability in use, across modalities)
- **Principle X** — C3, D3 (designed for governed evolution)

The crosswalk records each practice's principal Canon basis and principal lifecycle home. Additional stages, principles and Governance Primitives exercised by the practice are stated in its specification. Read down the principal-home column, the ten practices provide one primary practice for each of the nine lifecycle stages and the Acceptance transition. Read across the Canon column, every Canonical Principle must be discharged either through a principal mapping or an explicitly identified secondary obligation. Together, these two tests establish lifecycle coverage and constitutional coverage without assuming a permanent one-to-one correspondence between principles, stages and practices.

Relationship to KF-3000

The practices defined here specify engineering responsibilities — what must be done, what must be produced, and how compliance is verified. KF-3000 defines the architectural services that realise those responsibilities, and must in particular provide the services that produce each practice's mandatory artefacts and that enforce its governance checkpoints as continuous conformance checks. Because this document keeps identity, version and lineage distinct, and states each artefact by its required semantic content rather than its form, KF-3000 can justify each service by the practice obligation it fulfils, the artefact it maintains and the checkpoint it enforces — rather than inventing components one by one. No technology choices are made in this document; the practices are stated so that many architectures could satisfy them, and so that any that claims to must be measurable against the success criteria and failure modes recorded above.

KF-2003 · Materiality and Assurance

How reliance and consequence determine the depth to which a governed knowledge object must be assured

Document Class: Methodology · Parent Programme: Knowledge Foundry · Parent Document: KF-2000 · Status: Draft for Experiment · Version: 0.1

```
Principle (KF-1000)  →  ▸ Method (KF-2000)  →  Architecture (KF-3000)
→  Implementation (KF-4000)
```

Purpose

This document defines **materiality** in Knowledge Foundry and the **assurance classes** through which materiality is operationalised. It fixes, in one place, the depth to which a governed knowledge object's obligations must be discharged as a function of the reliance placed on it and the consequences of its being wrong.

It exists because the lifecycle (KF-2001), the practices (KF-2002) and the architecture (KF-3000 series) each state obligations without stating *how much* rigour each obligation demands for a given object. Uniform rigour is the wrong answer in both directions: it over-governs exploratory notes and under-governs consequential advice. Materiality supplies the missing dimension, and it supplies it once, so that gates, controls, reviews, evidence and traceability do not each invent their own interpretation.

A single rule governs everything below: **materiality grades the depth of assurance; it never grades the invariants.** The six lifecycle invariants of KF-2001 hold identically for every object at every class. What scales is the evidence, review, authority and reconstruction depth layered above that floor.

How This Document Sits in the Programme

KF-2003 is a **Method-level** document. It defines an obligation *shape* — proportionality — that the architecture then realises. Its architectural realisation is fixed by **KF-3001 IC-23 (Proportionate Discharge)**, **KF-3005** (gates and controls read the class), and the **Scope-of-Reliance Record (SVC-06)** to which class attaches. Its conformance consequences are carried by **KF-3007**. This document names no technologies and no deployments.

Assurance classes are one operational expression of materiality — the principal one — but they are not the whole of it. Materiality also informs which activities are recorded (KF-3004 §5.1), which decisions require independent review, and how deep a reconstruction must go (KF-3007). The classes make materiality actionable at the gate; the concept remains broader than the scale.

Part I · Materiality

1. Definition

> **Materiality** is the degree to which an Operational Knowledge Object, claim, decision or transition may affect consequential action, institutional reliance, legal or professional obligation, financial exposure, public representation, or the preservation of important knowledge.

Reliance is central to materiality but does not exhaust it. An object may carry high materiality because it will be relied upon, because being wrong would be costly, or because it stands for the institution publicly — often several at once. Materiality is therefore assessed across dimensions, not read off a single axis.

2. Dimensions of Materiality

A material assessment weighs, at minimum:

1. **Reliance** — will identified actors or systems act on this object, and how directly?
2. **Consequence** — what is the severity of a wrong, stale or misused conclusion?
3. **Audience and representation** — is the object internal, client-facing, or public; does it speak for the institution?
4. **Obligation** — does any legal, regulatory, contractual or professional duty attach to it?
5. **Exposure** — what financial, safety, reputational or operational loss could follow?
6. **Preservation value** — is this a claim the institution must be able to reconstruct and defend later?

No dimension alone fixes the class. The assessment takes the class that the highest-weighing dimensions justify.

3. Materiality Is Contextual and Version-Specific

Materiality attaches primarily to an object's **intended use and reliance context**, not permanently to the object itself. The same content may begin exploratory and become consequential when it enters a board paper, a policy submission or a client recommendation.

Accordingly, **assurance class attaches to the release / scope of reliance**, not to the immutable object identity (KF-3000 §2; SVC-06). A new intended use is a new reliance context and may carry a different class. This preserves the single-identity invariant: the object does not change identity when its class changes; a differently-classed reliance is a property of the release, recorded against it.

Part II · Assurance Classes

4. The Scale

Five classes grade the minimum rigour expected. They are ordinal: each inherits the expectations of those below it and adds to them.

Class	Name	Meaning
A0	Exploratory	Early inquiry, incomplete evidence, no authorised reliance
A1	Working	Active development and internal collaboration
A2	Routine *(provisional name — see note)*	Approved for defined routine use
A3	Consequential	Capable of materially influencing significant decisions or actions
A4	Critical	Failure, error or loss could create severe institutional, legal, financial, safety or public consequences

> **Naming note (A2).** "Operational" was rejected for A2 because the term is already load-bearing three ways in the programme — in "Operational Knowledge Object", as the Stage-5 lifecycle state, and as the KF-3000 §20 maturity profile. A2 is provisionally named **Routine** (alternatively **Standard**); the name may be settled at first review without affecting the scale.

5. Governance Overlays

Certain conditions are **not** degrees of reliance and must not be modelled as higher classes. Regulation, confidentiality, public exposure, client-facing status and legal privilege are **overlays**: orthogonal attributes that add specific obligations without moving the object up the scale. A low-consequence administrative record may be regulated; a high-consequence internal strategy may not be.

```yaml
assurance_class: A3              # sets the depth of assurance
governance_overlays:            # add specific, additional obligations
  - regulated
  - confidential
  - client-facing
```

The class determines depth; the overlays add duties. An object may be *Routine and regulated*, *Consequential and confidential*, *Critical and public*, or *Working and legally-privileged*.

6. Minimum Obligations by Class

These are floors, not ceilings. A domain governance profile may add obligations; it may never remove the invariant floor (§9) or fall below the class minimum.

Class	Typical minimum governance expectation
A0 Exploratory	provenance captured; uncertainty visible; no acceptance required

Class	Typical minimum governance expectation
A1 Working	named owner; versioning; basic evaluation; collaborative traceability
A2 Routine	defined scope of use; acceptance by an authorised role; monitored currency
A3 Consequential	independent review; explicit justification; counter-evidence recorded; stronger traceability; defined scope of reliance
A4 Critical	formal authority; segregation of duties where appropriate; comprehensive evidence; heightened monitoring, recovery and reconstruction depth

The obligations are stated as *outcomes a reviewer can confirm*, not as fixed metrics. Thresholds and cadences are set by the applicable domain profile (KF-4000), consistent with KF-2002's treatment of success criteria.

Part III · Assigning and Changing Class

7. Who Assigns the Class

The object **owner (Steward) proposes** the assurance class. For **A3 and A4**, the designated **governance authority confirms** it. Automated classification may *recommend* a class but is never the final authority for consequential or critical objects. Any downstream user whose intended reliance exceeds the assigned class **may request escalation**; unresolved escalation blocks that reliance until the class is confirmed.

Assignment is itself a governed, attributable transition: it records who proposed, who confirmed, when, and on what basis. There is no silent classification.

8. Raising and Lowering Class

Raising class is expected whenever reliance or consequence increases — a new audience, a new decision context, a new obligation. It is recorded as an attributable transition and re-tests the gates to the new depth.

Lowering class is permitted only through an attributable decision that records the changed scope of reliance — for example, when a Consequential object becomes a historical reference no longer relied upon operationally. **Historical evidence obligations are not retrospectively removed:** a lowered class governs only future reliance; the evidence already required at the higher class remains preserved (lineage invariant, KF-2001).

Part IV · Universality of Invariants

9. What Class Never Touches

The six lifecycle invariants — single identity, attributable transitions, justified transitions, inspectable transitions, provenance always present, lineage preserved — hold **identically at every class**, including A0. An Exploratory object still has one identity, still records origin, still transitions attributably and justifiably. The class governs the *depth of evidence and review*, never the *existence* of the invariant properties.

This is the boundary that keeps proportionality from becoming a loophole. Materiality answers "how deep"; it may never answer "whether at all" for any invariant.

Part V · Exceptions and Escalation

10. Exceptions

A time-limited, attributable **governance exception** may permit an object to operate one step below the class its use would otherwise require, provided: the gap is recorded; a named authority accepts the residual risk; the exception is visible until closed (KF-3001 IC-19); and **no invariant is waived**. An exception may reduce required depth for a bounded period; it may never convert an invariant violation into conformance.

11. Escalation

Where evidence of a mis-classification arises — an object relied upon beyond its class, or a class assigned below its dimensions — any actor may raise an escalation event (KF-2001 event model). Escalation obliges the authority to re-confirm or re-assign the class before the contested reliance proceeds.

Part VI · Relationship to Other Documents

- **KF-2001** — the invariants that class never grades, and the gates whose depth it sets.
- **KF-2002** — success criteria whose required depth is fixed here.
- **KF-3001 · IC-23** — the architectural constraint that discharges obligations to class depth; **IC-24** — event-native evidence, whose volume this document keeps proportionate.
- **KF-3004 · §5.1** — the material-activity definition this document governs.
- **KF-3005 · §28, §32** — gates and controls read the class.
- **KF-3008** — the authority model that confirms A3/A4 classes.
- **KF-3007** — the conformance model whose Integrity and Reconstructability tests are graded by class.

Status and Next Step

This document is **Draft for Experiment**. Its scale, minimum-obligation table and assignment rules are hypotheses to be tested by the first vertical slice (a MacroLab Singapore Business Impact Assessment, provisionally A3 Consequential). The slice is expected to show whether the class minimums are workable, whether the dimensions are sufficient, and whether A2's name should be settled. The document moves to *Validated with Exceptions*, then *Approved*, only after that evidence is in.

KF-3000 · Knowledge Foundry Reference Architecture

The logical architecture of a governed knowledge system

Document Class: Reference Architecture · Parent Programme: Knowledge Foundry · Status: Draft · Version: 0.3

> **Service-catalogue alignment (v0.3).** The service catalogue (§12) and every service reference in this document have been renumbered to the **definitive nineteen-service catalogue fixed by KF-3003**. The earlier provisional sixteen-service numbering — in which, for example, SVC-05 was Evaluation and SVC-15 was Conformance — has been superseded. KF-3003 is authoritative for the service catalogue; KF-3004 is authoritative for record ownership. KF-3000's *capability* enumeration (§6–§10) remains the provisional umbrella model and is refined definitively by KF-3002; a note at §12 and §23 identifies the three services (Release, Operating Condition and Exception, Temporal Continuity) whose capability grounding is established there rather than in this document's provisional capability list.

```
Principle (KF-1000)
        ↓
Lifecycle (KF-2001)
        ↓
Engineering Practices (KF-2002)
        ↓
▸ Reference Architecture (KF-3000)
        ↓
Implementation (KF-4000)
```

Purpose

This document defines the Knowledge Foundry Reference Architecture: the technology-neutral arrangement of capabilities, services, records and control relationships required to realise the Canon, lifecycle and engineering practices of a governed knowledge system.

KF-3000 does not prescribe a product, platform, vendor, storage model or deployment topology. It defines the logical responsibilities that any conforming architecture must discharge.

The Reference Architecture answers:

> *What must a governed knowledge system be architecturally capable of doing?*

KF-4000 answers:

> *How are those capabilities instantiated in a particular implementation?*

Architectural Derivation

Every architectural element must be traceable to an upstream obligation.

```
Canonical Principle
        ↓
```

```
Lifecycle stage, gate or invariant
        ↓
Engineering practice
        ↓
Mandatory artefact or governance checkpoint
        ↓
Architectural capability
        ↓
Architectural service
```

An architectural component is admitted only when it can be justified by this chain. The Reference Architecture therefore does not begin with technologies. It begins with the obligations the system must preserve.

Architectural Admission Test

A proposed architectural capability or service belongs in KF-3000 only if all of the following are true:

1. It discharges one or more obligations defined in KF-2001 or KF-2002.
2. Its responsibility can be stated without naming a specific technology or product.
3. Its inputs, outputs and governed records can be identified.
4. Its relationship to accountability and authority is explicit.
5. Its conformance can be inspected or tested.
6. Removing it would leave at least one lifecycle gate, invariant, practice obligation or Governance Primitive unsupported.

If a proposed element fails these tests, it belongs either in KF-4000 or outside the Knowledge Foundry architecture.

Part I · Architectural Foundations

1. Scope of the Reference Architecture

The Reference Architecture governs the operational life of Operational Knowledge Objects and the engineering artefacts attached to them.

Its scope includes:

- object identity and representation;
- evidence and justification;
- acceptance and authority;
- operationalisation and application;
- provenance and traceability;
- governance and stewardship;
- versioning, branching and lineage;
- change-impact analysis;

- outcome capture;
- retirement and historical preservation;
- identity and access governance;
- confidentiality and disclosure constraints;
- integrity protection;
- role- and authority-based permissions;
- retention and lawful disposition requirements;
- conformance and audit.

Its scope does not include:

- specific software products;
- physical infrastructure;
- deployment geography;
- organisational workflow design;
- user-interface design;
- model or database selection;
- implementation-specific security technologies and controls;
- performance thresholds.

Those concerns are defined in KF-4000 or in implementation-specific profiles. The Reference Architecture states the *logical* obligation to control access, disclosure and integrity; it does not name the encryption, identity-management or network technologies that realise it.

2. Architectural Unit

The primary architectural unit is the **Operational Knowledge Object**.

The architecture must treat the following as distinct but related concerns:

```
Object Identity
        ≠
Object Version
        ≠
Object Representation
        ≠
Object State
        ≠
Object Lineage
        ≠
Actionable Form
        ≠
Engineering Artefact
```

A conforming architecture must preserve these distinctions even when an implementation stores them together.

The three that are most often confused are fixed here precisely. An **object** is the enduring governed identity. A **version** is an immutable state of that object's represented content and attached warrant at a point in its history. A **release** is a version authorised for a defined scope of operational reliance. A new version does not automatically become a release, and a release does not create a new object identity.

Branching and consolidation create successor objects with new identities and explicit lineage links. They do not divide, merge or reuse the immutable identities of existing objects.

3. Architectural Invariants

The following properties must hold across the architecture:

1. **Stable identity** — each object has one immutable identity.
2. **Persistent provenance** — origin and subsequent lineage remain attached throughout the object's life.
3. **Inspectable state** — the current lifecycle state and the basis for entering it are recoverable.
4. **Attributable transition** — every state transition is assigned to an identifiable actor or authority.
5. **Justified transition** — every transition carries its recorded warrant.
6. **Version preservation** — prior versions are retained and recoverable.
7. **Separation of concerns** — identity, state, version, lineage, acceptance and representation remain logically distinct.
8. **Human accountability** — computational execution never removes identifiable human accountability.
9. **Traceable use** — every material application can be traced to the object version, actionable form and acceptance-state used.
10. **Historical continuity** — retirement preserves the object's record and lineage.
11. **Modality independence** — the obligations hold whether knowledge is represented symbolically, parametrically or through a combination of modalities.
12. **Continuous conformance** — governance conditions are maintained over time, not verified only once.

(KF-3001 separates these into three categories gathered here: architectural principles, system invariants, and service-interaction constraints.)

4. Architectural Perspectives

The same governed knowledge system must be inspectable through five complementary perspectives.

4.1 Object Perspective

What objects exist, how are they identified, structured, related and versioned?

4.2 Evidence Perspective

What evidence, counter-evidence, assumptions and justification warrant each object?

4.3 Lifecycle Perspective

What state is the object in, what event triggered its movement, and what gate authorised the transition?

4.4 Governance Perspective

Who owns, evaluates, accepts, reviews and retires the object?

4.5 Operational Perspective

How is the object made actionable, where is it used, and what outcomes arise from that use?

No one perspective is sufficient by itself. The architecture must preserve their relationships.

Part II · Capability Architecture

5. Capability Domains

The Reference Architecture is organised into four capability domains derived from KF-2002.

Domain	Architectural purpose	Principal upstream practices
A. Identity & Representation	Establish what the object is and make it inspectable	A1, A2
B. Evidence & Justification	Establish why the object may be trusted	B1, B2
C. Operational Governance	Control acceptance, custody and change	C1, C2, C3
D. Operational Use	Make knowledge actionable, observe its use and preserve its history	D1, D2, D3

These four domains are supported by a fifth, cross-cutting domain:

Domain	Architectural purpose	Upstream basis
E. Cross-Cutting: Provenance, Assurance & Access	Preserve traceability, verify continuing conformance, and govern access and integrity across every domain	Architectural invariants (§3); Governance Primitives; lifecycle gates; practice checkpoints and artefacts

The first four domains mirror the practice structure of KF-2002. The fifth gathers the three concerns that serve every domain rather than belonging to one: **provenance and traceability**, derived from practices A1, A2, B2, C2, C3 and D1; **assurance and conformance**, derived from

the architectural invariants and the gates and checkpoints it exists to keep honest; and **access and integrity**, derived from the human-accountability and attributable-transition invariants and the Authority and Provenance Governance Primitives.

Each capability below is specified against the same compact form: Purpose, Upstream basis, Required capabilities, Principal artefacts maintained, and a **Conformance condition** — a single high-level, testable statement of what an independent reviewer must be able to confirm. The detailed tests are deferred to KF-3002 and KF-3007; the conformance condition makes each capability testable in principle.

---

6. Domain A — Identity & Representation Capabilities

A-CAP-01 · Object Registration

Purpose. Admit a candidate object and assign its durable identity.

Upstream basis. KF-2002 A1.

Required capabilities.

- assign an immutable object identifier;
- distinguish a new object from a version, branch, consolidation or duplicate;
- record the object's origin;
- prevent identifier reuse;
- resolve an identifier to exactly one object.

Principal artefacts maintained.

- Identity Record
- Provenance Seed
- Intake Record

Conformance condition. For any object, an independent reviewer can resolve its identifier to exactly one object, retrieve its recorded origin, and confirm the identifier was never reused or reassigned.

A-CAP-02 · Structured Representation

Purpose. Represent an object in an inspectable form.

Upstream basis. KF-2002 A2.

Required capabilities.

- assign semantic type;
- represent constituent claims or elements;
- attach metadata;
- represent relationships to other objects;

- preserve the relationship between source material and structured form;
- accommodate multiple computational modalities while retaining inspectability.

Principal artefacts maintained.

- Structured Representation
- Metadata Record
- Relationship Map
- Extended Provenance Record

Conformance condition. For any object, a reviewer can retrieve its semantic type, constituent claims, metadata, relationships and provenance without reconstructing them from unstructured source material.

A-CAP-03 · Object Resolution and Discovery

Purpose. Enable authorised actors and services to locate and inspect objects and their related records.

Upstream basis. KF-2002 A1, A2 (the read side of identity and representation).

Required capabilities.

- resolve objects by identity;
- retrieve the current and historical states;
- traverse relationships and lineage;
- distinguish current, superseded, branched, consolidated, suspended and retired objects;
- expose sufficient context for downstream evaluation and governance.

Principal artefacts maintained.

- Resolution Index (a derived, rebuildable index over existing records; A-CAP-03 reads and indexes rather than originating primary records)

Conformance condition. For any identifier or relationship, a reviewer can resolve the object, retrieve its current and historical states, and traverse its lineage, distinguishing current, superseded, branched, consolidated, suspended and retired objects.

---

7. Domain B — Evidence & Justification Capabilities

B-CAP-01 · Evidence Registration

Purpose. Register supporting and defeating evidence as first-class governed records.

Upstream basis. KF-2002 B1 and B2.

Required capabilities.

- identify evidence independently of the claims it supports;

- classify evidence type and origin;
- attach evidence to one or more claims or objects;
- distinguish supporting, defeating and contextual evidence;
- preserve evidence provenance.

Principal artefacts maintained.

- Evidence Record
- Evidence-to-Claim Link
- Counter-evidence Record

Conformance condition. For any claim, a reviewer can retrieve the supporting, defeating and contextual evidence attached to it, each with its own provenance.

B-CAP-02 · Quality Evaluation

Purpose. Assess whether an object is suitable to proceed toward justification.

Upstream basis. KF-2002 B1.

Required capabilities.

- apply stated evaluation criteria;
- record pass, return or reject verdicts;
- record confidence, completeness and consistency assessments;
- retain rejected and returned evaluations;
- prevent progression where the quality gate is unmet.

Principal artefacts maintained.

- Quality Assessment
- Evaluation Verdict
- Evaluation Criteria Record

Conformance condition. For any object that advanced, a reviewer can retrieve the recorded verdict, the criteria applied and the reasons; returned and rejected objects carry the same record and did not proceed.

B-CAP-03 · Justification Construction

Purpose. Assemble and preserve the warrant for belief.

Upstream basis. KF-2002 B2.

Required capabilities.

- connect claims to supporting and counter-evidence;
- record assumptions and rationale;
- preserve traceability from claim to source;
- expose the justification to independent inspection;

- preserve an inspectable evidence, evaluation and rationale record sufficient to explain how parametrically represented knowledge or capability was admitted, bounded and relied upon.

Principal artefacts maintained.

- Justification Record
- Assumption Record
- Rationale Record
- Traceability Chain

Conformance condition. For any operational object, a reviewer can trace each material claim to its supporting and defeating evidence, recover the stated assumptions and reproduce the recorded rationale.

B-CAP-04 · Epistemic Status Management

Purpose. Represent the current epistemic standing of the object without conflating it with acceptance or lifecycle state.

Upstream basis. KF-2002 B2 (Principle VIII); the Landscape's separation of justification from acceptance.

Required capabilities.

- distinguish unassessed, quality-assessed, warrant-established, contested, warrant-weakened and insufficiently-supported conditions;
- represent confidence and unresolved challenge;
- preserve disagreement and competing claims;
- prevent epistemic status from being overwritten by operational acceptance.

Principal artefacts maintained.

- Epistemic Status Record
- Challenge Record

Conformance condition. For any object, a reviewer can determine its current epistemic status and any unresolved challenge, and confirm that acceptance has not overwritten it.

---

8. Domain C — Operational Governance Capabilities

C-CAP-01 · Authority and Acceptance

Purpose. Manage the governed transition from justified object to accepted operational object.

Upstream basis. KF-2002 C1.

Required capabilities.

- identify the accepting authority;
- record acceptance, provisional acceptance or refusal;
- define the scope and conditions of reliance;
- attribute and date the decision;
- return refused objects to the appropriate stage;
- prevent operational use without valid acceptance.

Principal artefacts maintained.

- Acceptance Record
- Refusal Record
- Scope-of-Reliance Record

Conformance condition. For any operational object, a reviewer can identify the accepting authority, the decision, the date and the scope of reliance; refusals carry the authority's recorded reasons.

C-CAP-02 · Stewardship and Ownership

Purpose. Maintain identifiable accountability throughout the operational life of the object.

Upstream basis. KF-2002 C2.

Required capabilities.

- assign and transfer stewardship;
- identify responsible roles;
- ensure no operational object is ownerless;
- preserve historical accountability;
- distinguish execution by machines from accountability held by people.

Principal artefacts maintained.

- Stewardship Assignment
- Responsibility Record
- Ownership History

Conformance condition. For any operational object, a reviewer can identify its current steward and its ownership history, and confirm that no operational object is ownerless.

C-CAP-03 · Review and Compliance

Purpose. Preserve legitimacy as time, context and policy change.

Upstream basis. KF-2002 C2.

Required capabilities.

- schedule and record reviews;
- detect overdue reviews;

- assess policy or governance compliance;
- suspend or flag objects whose standing gate is no longer met;
- preserve review and compliance history.

Principal artefacts maintained.

- Review Schedule
- Review Record
- Compliance Record
- Exception Record

Conformance condition. For any operational object, a reviewer can retrieve its review schedule, last review, compliance status and any exception, and the system can detect overdue reviews.

C-CAP-04 · Version and Release Management

Purpose. Preserve controlled change without loss of history.

Upstream basis. KF-2002 C2 and C3.

Required capabilities.

- create immutable versions;
- designate current, superseded and historical versions;
- preserve prior versions;
- distinguish draft, accepted and released forms;
- associate each version with its justification and acceptance-state.

Principal artefacts maintained.

- Version Record
- Release Record
- Supersession Link

Conformance condition. For any object, a reviewer can enumerate its versions, identify the current, superseded and historical versions and any releases, and confirm no prior version was overwritten.

C-CAP-05 · Lineage and Evolution

Purpose. Govern amendment, branching, consolidation, supersession and reactivation.

Upstream basis. KF-2002 C3.

Required capabilities.

- represent predecessor and successor relationships;
- preserve branch and merge lineage;
- record change triggers and rationales;

- return changed objects through evaluation, justification and acceptance;
- prevent silent mutation.

Principal artefacts maintained.

- Change Record
- Lineage Record
- Branch Record
- Consolidation Record

Conformance condition. For any change, a reviewer can recover the predecessor and successor relationships, the change trigger and rationale, and confirm the changed object returned through evaluation, justification and acceptance.

C-CAP-06 · Dependency Impact Management

Purpose. Identify and disposition objects and actionable forms affected by change.

Upstream basis. KF-2002 C3.

Required capabilities.

- identify downstream dependencies;
- determine which dependants require revalidation;
- recommend suspension, re-evaluation or evolution of affected objects;
- record their disposition;
- prevent changed knowledge from leaving dependants silently inconsistent.

Principal artefacts maintained.

- Dependency Impact Record
- Revalidation Record
- Suspension Recommendation

Conformance condition. For any change, a reviewer can identify the affected dependants and their disposition, and confirm none was left silently inconsistent.

C-CAP-07 · Event and Transition Management

Purpose. Govern movement between lifecycle states.

Upstream basis. KF-2001 (states, transitions, events, gates).

Required capabilities.

- register lifecycle events;
- evaluate transition eligibility;
- enforce exit gates;
- record forward, return, suspension, reactivation and retirement transitions;
- attribute each transition;

- preserve transition history.

Principal artefacts maintained.

- Event Record
- Transition Record
- Gate Decision Record
- Suspension Record
- Reactivation Record

Conformance condition. For any lifecycle transition, an independent reviewer can recover the triggering event, the prior and resulting states, the applicable gate, the gate decision, the accountable actor and the recorded warrant.

---

9. Domain D — Operational Use Capabilities

D-CAP-01 · Actionable Form Derivation

Purpose. Derive actionable forms from an accepted object — the forms usable in decisions, reasoning and automation.

Upstream basis. KF-2002 D1.

Required capabilities.

- derive decision rules, playbooks, prompts, procedures, models and other actionable forms;
- identify the object version from which each form derives;
- preserve links to justification and acceptance;
- distinguish the actionable form from the source knowledge object;
- withdraw or update actionable forms when the source object changes.

Principal artefacts maintained.

- Actionable Form
- Derivation Record
- Warrant Linkage Record

Conformance condition. For any actionable form, a reviewer can identify the object version, justification and acceptance-state from which it derives, and confirm it is withdrawn or updated when its source object changes.

D-CAP-02 · Controlled Application

Purpose. Enable governed use while preserving traceability and explainability.

Upstream basis. KF-2002 D2.

Required capabilities.

- identify the object version and actionable form used;
- verify that acceptance and governance remain valid;
- record material use — materiality criteria and assurance classes are defined by KF-2003 · Materiality and Assurance (a domain governance profile may set additional thresholds within it);
- preserve the path from input to result;
- provide an explanation appropriate to the modality, materiality and decision context of the application, without claiming forms of interpretability the underlying representation cannot support;
- prevent use of suspended, retired or unaccepted objects.

Principal artefacts maintained.

- Application Record
- Use Context Record
- Explanation Record

Conformance condition. For every material application, a reviewer can identify the object version and actionable form used, the governing acceptance-state and the recorded path from input to result, and confirm that suspended, retired or unaccepted objects were not used.

D-CAP-03 · Outcome and Feedback Capture

Purpose. Capture what happened when knowledge was applied and return it as lifecycle evidence.

Upstream basis. KF-2002 D2.

Required capabilities.

- attach outcomes to the relevant object version and application;
- distinguish observation from judgement;
- raise review, evolution or retirement events;
- aggregate outcomes without erasing individual provenance;
- preserve adverse and contradictory outcomes.

Principal artefacts maintained.

- Outcome Record
- Feedback Record
- Trigger Record

Conformance condition. For any application, a reviewer can retrieve the captured outcomes attached to the object version, distinguish observation from judgement, and trace any review, evolution or retirement event raised from them.

D-CAP-04 · Retirement and Historical Preservation

Purpose. Conclude operational use while preserving lineage and reversibility.

Upstream basis. KF-2002 D3.

Required capabilities.

- record retirement reason, date and authority;
- remove an object from active operational use;
- preserve the full historical record;
- maintain successor and supersession links;
- assemble the restoration package that permits governed reactivation through Evolution;
- prevent destructive deletion of governed history.

Principal artefacts maintained.

- Retirement Record
- Historical Object Record
- Restoration Package

Conformance condition. For any retired object, a reviewer can identify the retirement reason, date and authority, retrieve its full historical record and lineage, and confirm that reactivation remains possible and no history was destroyed.

---

10. Domain E — Cross-Cutting: Provenance, Assurance & Access

Domain E is cross-cutting. It gathers the three capabilities that serve every other domain rather than belonging to one. Its assurance members do not create or alter the substantive knowledge represented by an object, nor do they authorise lifecycle transitions; they evaluate the objects, records, controls and service actions through which the architecture demonstrates conformance. Its provenance and access members maintain records of their own, but likewise never authorise a transition or alter substantive content.

E-CAP-01 · Provenance and Traceability

Purpose. Preserve origin, derivation, custody and end-to-end traceability across every stage of the object lifecycle.

Upstream basis. Architectural invariants 2, 3, 4, 9 and 10; KF-2002 A1, A2, B2, C2, C3 and D1.

Required capabilities.

- preserve source origin;
- trace representations and claims to source material;
- trace versions to predecessor versions and change events;
- trace actionable forms to object versions, justification and acceptance;
- trace applications and outcomes to the forms and versions used;
- preserve custody and attribution across transitions;
- detect broken lineage or incomplete traceability.

Principal artefacts maintained.

- Provenance Seed
- Extended Provenance Record
- Traceability Chain
- Derivation and Custody Links

Conformance condition. For any object, version, actionable form, application or outcome, a reviewer can trace it to its origin and to every predecessor state, and any broken lineage or incomplete traceability is detectable.

E-CAP-02 · Invariant Monitoring

Purpose. Continuously verify that the architectural invariants of §3 hold across all objects.

Upstream basis. Architectural invariants (§3); lifecycle invariants (KF-2001).

Required capabilities.

- evaluate each architectural invariant against live and historical objects;
- detect and record violations;
- raise conformance exceptions;
- verify continuously rather than once.

Principal artefacts maintained.

- Invariant Test Register
- Conformance Exception Record

Conformance condition. For any architectural invariant, the system can demonstrate across the object population that it holds, and any violation raises a recorded conformance exception.

E-CAP-03 · Gate and Checkpoint Assurance

Purpose. Verify that every lifecycle gate and practice governance checkpoint was enforced and satisfied.

Upstream basis. Lifecycle gates (KF-2001); practice governance checkpoints (KF-2002).

Required capabilities.

- confirm each transition passed its exit gate;
- detect gate bypass;
- verify that the checkpoint artefacts required by KF-2002 exist for each transition;
- record verification.

Principal artefacts maintained.

- Gate Conformance Record
- Checkpoint Verification Record

Conformance condition. For any transition, the system can demonstrate that its exit gate and the required KF-2002 checkpoints were enforced and satisfied, and detect any bypass.

E-CAP-04 · Artefact Completeness Verification

Purpose. Verify that every mandatory artefact required by a practice exists, is inspectable and is retrievable.

Upstream basis. KF-2002 mandatory artefacts; the artefact definition of KF-2002.

Required capabilities.

- enumerate the required artefacts for each object and lifecycle state;
- detect missing or unretrievable artefacts;
- confirm semantic completeness — that an artefact holds its required content, not merely that a record exists.

Principal artefacts maintained.

- Artefact Completeness Record

Conformance condition. For any object and lifecycle state, the system can enumerate the required artefacts and demonstrate that each exists, is retrievable and holds its required content.

E-CAP-05 · Audit and Reconstruction

Purpose. Enable independent reconstruction of an object's full history for audit.

Upstream basis. Architectural invariants 3–5, 9–10; conformance evidence (§21).

Required capabilities.

- reconstruct the end-to-end lifecycle of any object — states, transitions, warrant, authority and use;
- produce inspectable audit trails;
- support external inspection without privileged reinterpretation.

Principal artefacts maintained.

- Audit Trail
- Reconstruction Record

Conformance condition. For any object, the system can reconstruct its end-to-end lifecycle — states, transitions, warrant, authority and use — as an inspectable audit trail, without privileged reinterpretation.

E-CAP-06 · Access and Integrity Assurance

Purpose. Ensure that governed objects, artefacts and transitions are accessed and altered only by authorised actors operating within their assigned roles.

Upstream basis. Human-accountability invariant; attributable-transition invariant; the Authority and Provenance Governance Primitives.

Required capabilities.

- distinguish authority, accountability and access permission;
- control who may view, create, modify, accept, release, apply, suspend and retire an object;
- protect governed records against unauthorised alteration;
- record access and material administrative action;
- enforce disclosure or use restrictions attached to an object;
- preserve access-control history as roles and ownership change.

Principal artefacts maintained.

- Access Policy Record
- Permission Assignment
- Disclosure Constraint Record
- Access and Administrative Action Log

Conformance condition. For any governed object, artefact or transition, a reviewer can determine which roles may view or alter it, confirm that access and material administrative actions were logged, and confirm that governed records were protected against unauthorised alteration.

A note on independence. Logical independence does not require a single implementation component. The assurance capabilities (E-CAP-02 to E-CAP-05) may be distributed across monitoring, policy-evaluation, audit and reporting components, provided assurance remains independent of the services whose conformance it evaluates.

Part III · Logical Service Architecture

11. From Capabilities to Services

An architectural service is a stable logical responsibility exposed to authorised actors or other services through defined inputs, outputs, records and controls. It may be realised by software, assigned human work, or a governed combination of both.

Capabilities state what the architecture must be able to do; services state the responsibilities through which those capabilities are made available. A service may realise several capabilities, and a capability may require several services: the relationship is many-to-many. Services are logical, not technological.

12. Core Architectural Services

This is the definitive **nineteen-service catalogue**, fixed here in alignment with KF-3003. It is grouped by domain: A (identity and representation); B (evidence, evaluation and justification);

C (governance, lifecycle, version, release and dependency); D (operationalisation, application and retirement); and E (five cross-cutting services — provenance, access, operating condition, conformance and time). Version, acceptance and release are three separate services, and the assurance concerns previously folded into one "Conformance and Audit" service are separated into provenance (SVC-15), operating condition (SVC-17), conformance and audit (SVC-18) and temporal continuity (SVC-19).

SVC-01 · Object Registry Service

Maintains durable identity, object resolution and basic status.

SVC-02 · Representation Service

Maintains structured representations, semantic types, metadata and object relationships.

SVC-03 · Evidence Service

Registers evidence, counter-evidence and evidence relationships.

SVC-04 · Evaluation Service

Supports quality assessment, verdicts and evaluation history.

SVC-05 · Justification Service

Maintains rationale, assumptions, warrant, epistemic status and claim-to-evidence traceability, holding epistemic status distinct from acceptance.

SVC-06 · Authority and Acceptance Service

Manages acceptance decisions, authority, scope and refusal.

SVC-07 · Stewardship and Review Service

Maintains ownership, review cycles and compliance status.

SVC-08 · Lifecycle and Transition Service

Manages object state, events, gates and governed transitions — including suspension and reactivation, which are lifecycle transitions. It holds the sole authority to write lifecycle state.

SVC-09 · Version and Lineage Service

Maintains immutable versions, branches, consolidations, supersession and historical lineage, kept strictly distinct from acceptance and release.

SVC-10 · Release Service

Designates an accepted version as eligible for a defined scope and period of operational reliance, and governs withdrawal, supersession and revalidation of that eligibility. Release is a distinct governed decision, separate from version creation (SVC-09) and acceptance (SVC-06).

SVC-11 · Dependency Service

Represents dependencies and performs change-impact analysis; recommends suspension where dependants are affected.

SVC-12 · Operationalisation Service

Creates and manages actionable forms and their warrant linkages.

SVC-13 · Application and Outcome Service

Records use, explanation, results and feedback.

SVC-14 · Retirement and Preservation Service

Manages retirement, historical state and restoration packages.

SVC-15 · Provenance and Traceability Service

Maintains origin, custody, derivation and end-to-end traceability. It is the principal service of the provenance capability (E-CAP-01) and maintains the provenance and traceability artefacts produced across Domains A, B, C and D.

SVC-16 · Access and Integrity Service

Governs who may view, create, modify, accept, release, apply, suspend and retire objects and artefacts; protects governed records against unauthorised alteration; enforces disclosure constraints; and logs access and material administrative action. It mediates access and permission; it does not itself authorise lifecycle transitions or hold substantive accountability, both of which remain with human roles.

SVC-17 · Operating Condition and Exception Service

Maintains service and transaction operating condition, determines the fail-posture when a required condition is unavailable or indeterminate, and holds operational exceptions until they are closed. *(Its capability basis — Operating Condition assurance — is fixed definitively in KF-3002; KF-3000's provisional capability model folded this concern into general assurance.)*

SVC-18 · Conformance and Audit Service

Verifies invariants, gates, artefact completeness and architectural conformance, and reconstructs object history for audit. It realises the assurance capabilities of Domain E and reads the records of every other service without altering them. Logical independence does not require a single component: SVC-18 may be distributed across monitoring, policy-evaluation, audit and reporting components, provided assurance remains independent of the services it evaluates.

SVC-19 · Temporal Continuity Service

Owns the common temporal model, historical sequencing, reconstruction and the preservation of recorded, effective and event time across governed records. *(Its capability basis — Temporal*

Continuity — is fixed definitively in KF-3002; it is not enumerated in KF-3000's provisional capability list.)

13. Service Interaction Principles

The services must interact according to the following rules:

1. No service may alter object identity after assignment.
2. No lifecycle transition may occur without a Transition Record.
3. No accepted operational object may lack a Justification Record and Acceptance Record.
4. No actionable form may exist without a traceable source object version.
5. No application may use an object whose acceptance or governance state is invalid.
6. No version change may overwrite a prior version.
7. No retirement operation may destroy lineage.
8. No automated service may become the final accountable authority.
9. All material service actions must be attributable and auditable.
10. Architectural services may automate work but may not silently bypass practice obligations.
11. The Conformance and Audit Service may read the governed records and control evidence of every service. It must not alter the records under examination, authorise the transitions it verifies, or rely solely on assertions produced by the service being assessed.
12. Access to governed records, and any change to permissions, is mediated by the Access and Integrity Service; no service grants itself access or alters its own permissions outside it.
13. Suspension and reactivation are lifecycle transitions owned by the Lifecycle and Transition Service; other services may recommend or trigger them but may not record the transition themselves.

Part IV · Architectural Records

14. Record Classes

The architecture distinguishes seven principal record classes, and a set of common properties every record must carry.

14.1 Object Records

Describe what the object is, and where it came from.

Examples: Identity Record, Intake Record, Structured Representation, Metadata Record, Relationship Map, Provenance Seed, Extended Provenance Record.

14.2 Epistemic Records

Describe why the object may be believed.

Examples: Evidence Record, Counter-evidence Record, Quality Assessment, Justification Record, Assumption Record, Rationale Record, Traceability Chain, Epistemic Status Record.

14.3 Governance Records

Describe who authorised, owns and reviews the object.

Examples: Acceptance Record, Refusal Record, Scope-of-Reliance Record, Stewardship Assignment, Review Record, Compliance Record.

14.4 Evolution Records

Describe how the object moved and changed through time.

Examples: Event Record, Transition Record, Gate Decision Record, Version Record, Change Record, Lineage Record, Dependency Impact Record, Suspension Record, Reactivation Record, Retirement Record.

14.5 Operational Records

Describe how the object was made usable and what happened when it was used.

Examples: Actionable Form, Derivation Record, Application Record, Outcome Record, Explanation Record.

14.6 Assurance Records

Describe whether the architecture is meeting its own obligations.

Examples: Invariant Test Register, Gate Conformance Record, Checkpoint Verification Record, Artefact Completeness Record, Audit Trail, Conformance Exception Record.

14.7 Access & Integrity Records

Describe who may access or alter an object, and what access occurred.

Examples: Access Policy Record, Permission Assignment, Disclosure Constraint Record, Access and Administrative Action Log.

14.8 Common Record Properties

Every governed record, of whatever class, must carry at minimum:

- a stable record identity;
- the object and version to which it relates;
- record type;
- creation time, and effective time where different;
- creator or originating service;
- accountable role where applicable;
- status;
- provenance;

- version or revision information;
- integrity status;
- retention or disposition condition;
- links to superseding or correcting records.

Where a record is corrected, the correction is appended or versioned; the prior record is not silently overwritten. The distinction between *creation time* and *effective time* is required: a policy change may be recorded on one date and take effect from another, and both must be recoverable.

15. Record Relationship Model

At minimum, the architecture must support the following logical relationships:

```
Operational Knowledge Object
     ├── has Identity Record
     ├── has one or more Versions
     ├── has current Lifecycle State
     ├── has Transition History
     ├── has Structured Representation
     ├── has Provenance
     ├── is supported or challenged by Evidence
     ├── has Quality Assessment
     ├── has Justification
     ├── has Epistemic Status
     ├── has Acceptance-State
     ├── has Steward
     ├── has Review History
     ├── is subject to Access and Disclosure Constraints
     ├── may derive Actionable Forms
     ├── may be used in Applications
     ├── may produce Outcomes
     ├── may depend upon or support other Objects
     ├── may branch, merge or supersede
     ├── may be Suspended
     ├── is subject to Conformance Verification
     └── may enter Historical State
```

The logical model fixes these relationships. It does not prescribe whether they are implemented in documents, relational tables, graphs, event streams, model context, or another substrate.

Part V · Control Architecture

16. Gate Enforcement

Each lifecycle gate must be represented as an inspectable architectural control.

Lifecycle point	Required architectural check
Capture exit	Identity and origin exist
Structure exit	Representation, type, relationships and provenance are inspectable
Evaluate exit	Quality verdict and reasons exist
Justify exit	Supporting evidence, counter-evidence, assumptions, rationale and traceability exist
Acceptance transition	Identified authority has recorded the acceptance-state and scope
Operationalise exit	Actionable form is linked to object version, justification and acceptance
Govern standing gate	Stewardship and review remain in force
Evolve exit	Change, version, lineage and dependency impacts are recorded
Apply exit	Application outcome and traceability are captured
Retire exit	Retirement reason, authority and historical preservation are recorded

Each check is enforced at execution by the Lifecycle and Transition Service (SVC-08) and verified after the fact by the Conformance and Audit Service (SVC-18): the first prevents an unearned transition, the second proves that none occurred.

17. Control Types

The architecture must support five classes of control.

17.1 Preventive Controls

Stop an invalid transition or use before it occurs.

17.2 Detective Controls

Identify missing, stale or contradictory conditions.

17.3 Corrective Controls

Return an object to an earlier stage, suspend it or trigger Evolution.

17.4 Evidentiary Controls

Produce inspectable proof that an obligation was discharged.

17.5 Historical Controls

Preserve prior states and enable reconstruction of what occurred.

18. Human Authority Boundaries

The architecture must distinguish:

- computation;
- recommendation;
- evaluation support;
- transition execution;
- accountable authorisation.

Automated systems may perform or assist the first four where permitted. They may not erase the requirement for an identifiable human authority where the Canon assigns acceptance, stewardship or accountability to a person or body.

Part VI · Conformance Architecture

19. Architectural Conformance

Conformance is assessed as **coverage, not one-to-one correspondence**, and to a depth **proportionate to assurance class** (KF-2003). A capability may be realised by several mechanisms and a mechanism may serve several capabilities; what must hold is that every obligation is covered, not that each maps to exactly one element. The nine clauses below are the structural integrity checklist; the four proportionate tests through which they are applied — Constitutional, Operational, Integrity and Reconstructability coverage — are defined authoritatively in **KF-3007 · Architectural Conformance Framework**. The six lifecycle invariants are demonstrated at full strength for every object regardless of class; the *depth* of evidence and reconstruction required of graded obligations scales with the object's assurance class.

An implementation conforms to KF-3000 when it demonstrates that:

1. every required capability is realised by at least one identified mechanism;
2. every mandatory artefact can be produced and retrieved;
3. every governance checkpoint can be enforced or inspected;
4. every architectural invariant is maintained, at full strength, for every object;
5. every lifecycle transition is attributable and reconstructable to the depth its assurance class requires;
6. every practice in KF-2002 is supported;
7. every Governance Primitive is instantiated;
8. access and integrity controls protect governed records and are themselves inspectable;
9. no implementation choice collapses distinctions the Reference Architecture requires to remain separate.

20. Progressive Implementation Profiles

The profiles below describe progressive implementation maturity. They **do not** constitute full conformance with KF-3000. An implementation is conformant only when it satisfies the complete requirements of §19, or an explicitly defined domain profile approved by the Knowledge Foundry governance authority.

These implementation-maturity profiles are **orthogonal to the per-object assurance classes** of KF-2003: a maturity profile describes how much of the architecture an implementation has stood up, while an assurance class describes how much rigour a *particular object* is governed to. A Foundational implementation and a Full-Conformance one may both hold Exploratory and Consequential objects; the class sets the depth, the profile sets the reach.

The profiles are cumulative:

```
Foundational
    ↓ adds
Governed
    ↓ adds
Operational
    ↓ adds
Full Conformance
```

Foundational

Supports identity, provenance, structure, evaluation, justification and acceptance.

Governed

Adds stewardship, reviews, versioning, lineage and controlled evolution.

Operational

Adds actionable forms, governed application, outcome capture and retirement.

Full Conformance

Adds continuous assurance, dependency impact, access and integrity governance, modality independence and complete lifecycle traceability — satisfying §19 in full.

21. Conformance Evidence

A conforming implementation must be able to produce:

- a capability-to-service mapping;
- a practice-to-capability mapping;
- an artefact implementation map;
- a lifecycle gate-control map;
- an authority, access and accountability map;
- an invariant test register;
- an access and integrity control register;

- a record-retention and lawful-disposition policy;
- evidence of representative end-to-end lifecycle execution.

Wherever possible this evidence is **event-native** — emitted by the governed acts themselves rather than authored separately (KF-3001 IC-24; KF-3004 §5.2) — and its required depth is **graded by assurance class** (KF-2003). The Conformance and Audit Service (SVC-18) is the architectural means by which this evidence is produced and kept current; the model that determines what must be reconstructable, and to what depth, is **KF-3007 · Architectural Conformance Framework**.

Part VII · Crosswalks

22. Practice-to-Capability Crosswalk

KF-2002 Practice	Principal architectural capabilities
A1 · OKO Identification	A-CAP-01, A-CAP-03
A2 · Knowledge Structuring	A-CAP-02, A-CAP-03
B1 · Evidence Evaluation	B-CAP-01, B-CAP-02
B2 · Knowledge Justification	B-CAP-01, B-CAP-03, B-CAP-04
C1 · Knowledge Acceptance	C-CAP-01, C-CAP-07
C2 · Governance & Stewardship	C-CAP-02, C-CAP-03, C-CAP-04
C3 · Knowledge Evolution	C-CAP-04, C-CAP-05, C-CAP-06, C-CAP-07
D1 · Operationalisation	D-CAP-01
D2 · Knowledge Application	D-CAP-02, D-CAP-03
D3 · Knowledge Retirement	D-CAP-04, C-CAP-05, C-CAP-07

Domain E is cross-cutting and does not map one-to-one to a practice. Its provenance capability (E-CAP-01) is derived from practices A1, A2, B2, C2, C3 and D1; its access capability (E-CAP-06) from the human-accountability and attributable-transition invariants and the Authority and Provenance Governance Primitives; and its assurance capabilities (E-CAP-02 to E-CAP-05) from the architectural invariants and the gates and checkpoints of KF-2001 and KF-2002, which they verify.

23. Capability-to-Service Crosswalk

Capability	Principal logical service
A-CAP-01	Object Registry Service
A-CAP-02	Representation Service
A-CAP-03	Object Registry and Representation Services
B-CAP-01	Evidence Service
B-CAP-02	Evaluation Service

Capability	Principal logical service
B-CAP-03	Justification Service
B-CAP-04	Justification and Lifecycle Services
C-CAP-01	Authority and Acceptance Service
C-CAP-02	Stewardship and Review Service
C-CAP-03	Stewardship and Review Service
C-CAP-04	Version and Lineage Service; Release Service
C-CAP-05	Version and Lineage Service
C-CAP-06	Dependency Service
C-CAP-07	Lifecycle and Transition Service
D-CAP-01	Operationalisation Service
D-CAP-02	Application and Outcome Service
D-CAP-03	Application and Outcome Service
D-CAP-04	Retirement and Preservation Service
E-CAP-01	Provenance and Traceability Service
E-CAP-02	Conformance and Audit Service
E-CAP-03	Conformance and Audit Service
E-CAP-04	Conformance and Audit Service
E-CAP-05	Conformance and Audit Service
E-CAP-06	Access and Integrity Service

Every capability in this document's provisional model has a principal service. Under the definitive nineteen-service catalogue (§12), three services are grounded in the definitive capability model of **KF-3002** rather than in KF-3000's provisional enumeration: **SVC-10 Release** (definitive capability C-CAP-05 Release), **SVC-17 Operating Condition and Exception** (E-CAP-04) and **SVC-19 Temporal Continuity** (E-CAP-05). Within KF-3000's provisional capabilities their responsibilities are folded respectively into version and release management (C-CAP-04), the assurance capabilities (E-CAP-02–E-CAP-05) and provenance (E-CAP-01). With provenance now seated at SVC-15 and access at SVC-16, and once the KF-3002 capability model is applied, no service is left without a capability basis.

24. Artefact-to-Service Crosswalk

This crosswalk identifies, for each mandatory artefact, the logical service that is its **authoritative maintainer** — the service holding primary architectural responsibility for creating and maintaining it. Other services may read an artefact; only human roles hold accountability in the normative sense reserved for Stewards and Authorities. Artefacts are grouped by the record classes of §14.

Object records

Artefact	Authoritative maintainer
Identity Record	Object Registry Service (SVC-01)
Intake Record	Object Registry Service (SVC-01)
Resolution Index	Object Registry Service (SVC-01), with SVC-02
Structured Representation	Representation Service (SVC-02)
Metadata Record	Representation Service (SVC-02)
Relationship Map	Representation Service (SVC-02)
Provenance Seed	Provenance and Traceability Service (SVC-15)
Extended Provenance Record	Provenance and Traceability Service (SVC-15)
Traceability Chain	Provenance and Traceability Service (SVC-15), with SVC-05
Derivation and Custody Links	Provenance and Traceability Service (SVC-15)

Epistemic records

Artefact	Authoritative maintainer
Evidence Record	Evidence Service (SVC-03)
Evidence-to-Claim Link	Evidence Service (SVC-03)
Counter-evidence Record	Evidence Service (SVC-03)
Quality Assessment	Evaluation Service (SVC-04)
Evaluation Verdict	Evaluation Service (SVC-04)
Evaluation Criteria Record	Evaluation Service (SVC-04)
Justification Record	Justification Service (SVC-05)
Assumption Record	Justification Service (SVC-05)
Rationale Record	Justification Service (SVC-05)
Epistemic Status Record	Justification Service (SVC-05)
Challenge Record	Justification Service (SVC-05), with SVC-08

Governance records

Artefact	Authoritative maintainer
Acceptance Record	Authority and Acceptance Service (SVC-06)
Refusal Record	Authority and Acceptance Service (SVC-06)
Scope-of-Reliance Record	Authority and Acceptance Service (SVC-06)
Stewardship Assignment	Stewardship and Review Service (SVC-07)

Artefact	Authoritative maintainer
Responsibility Record	Stewardship and Review Service (SVC-07)
Ownership History	Stewardship and Review Service (SVC-07)
Review Schedule	Stewardship and Review Service (SVC-07)
Review Record	Stewardship and Review Service (SVC-07)
Compliance Record	Stewardship and Review Service (SVC-07)
Exception Record	Stewardship and Review Service (SVC-07)

Evolution records

Artefact	Authoritative maintainer
Event Record	Lifecycle and Transition Service (SVC-08)
Transition Record	Lifecycle and Transition Service (SVC-08)
Gate Decision Record	Lifecycle and Transition Service (SVC-08)
Suspension Record	Lifecycle and Transition Service (SVC-08)
Reactivation Record	Lifecycle and Transition Service (SVC-08), with SVC-14
Version Record	Version and Lineage Service (SVC-09)
Release Record	Release Service (SVC-10)
Supersession Link	Version and Lineage Service (SVC-09); release supersession with SVC-10
Change Record	Version and Lineage Service (SVC-09)
Lineage Record	Version and Lineage Service (SVC-09)
Branch Record	Version and Lineage Service (SVC-09)
Consolidation Record	Version and Lineage Service (SVC-09)
Dependency Impact Record	Dependency Service (SVC-11)
Revalidation Record	Dependency Service (SVC-11)
Suspension Recommendation	Dependency Service (SVC-11); may also originate from SVC-07 or SVC-18

Operational records

Artefact	Authoritative maintainer
Actionable Form	Operationalisation Service (SVC-12)
Derivation Record	Operationalisation Service (SVC-12)

Artefact	Authoritative maintainer
Warrant Linkage Record	Operationalisation Service (SVC-12)
Application Record	Application and Outcome Service (SVC-13)
Use Context Record	Application and Outcome Service (SVC-13)
Explanation Record	Application and Outcome Service (SVC-13)
Outcome Record	Application and Outcome Service (SVC-13)
Feedback Record	Application and Outcome Service (SVC-13)
Trigger Record	Application and Outcome Service (SVC-13); raised as events to SVC-08
Retirement Record	Retirement and Preservation Service (SVC-14)
Historical Object Record	Retirement and Preservation Service (SVC-14)
Restoration Package	Retirement and Preservation Service (SVC-14)

Assurance records

Artefact	Authoritative maintainer
Invariant Test Register	Conformance and Audit Service (SVC-18)
Conformance Exception Record	Conformance and Audit Service (SVC-18)
Gate Conformance Record	Conformance and Audit Service (SVC-18)
Checkpoint Verification Record	Conformance and Audit Service (SVC-18)
Artefact Completeness Record	Conformance and Audit Service (SVC-18)
Audit Trail	Conformance and Audit Service (SVC-18)
Reconstruction Record	Conformance and Audit Service (SVC-18)

Access & integrity records

Artefact	Authoritative maintainer
Access Policy Record	Access and Integrity Service (SVC-16)
Permission Assignment	Access and Integrity Service (SVC-16)
Disclosure Constraint Record	Access and Integrity Service (SVC-16)
Access and Administrative Action Log	Access and Integrity Service (SVC-16)

Every mandatory artefact in this provisional set resolves to a single authoritative maintainer. Of the definitive nineteen services, seventeen maintain at least one artefact listed above; the two remaining cross-cutting services — **SVC-17 Operating Condition and Exception** and **SVC-19 Temporal Continuity** — maintain records (operating-condition status and operational exceptions; temporal, sequencing and reconstruction records) that are defined authoritatively in **KF-3004** and are not enumerated in this provisional artefact list. Two allocations are set deliberately: **suspension** and **reactivation** are lifecycle transitions owned

by SVC-08, while the originating service supplies the reason or recommendation; and the **restoration package** that supports reactivation is held by SVC-14, the maintainer of historical state.

Part VIII · Relationship to KF-4000

KF-3000 fixes the logical architecture. KF-4000 defines how that architecture is instantiated in a particular Knowledge Operating System, organisational environment or domain implementation.

An implementation may:

- combine several logical services into one component;
- distribute one service across several components;
- allocate responsibility across people and machines;
- use different storage and representation models;
- adopt different deployment and integration patterns.

It may not:

- omit a required capability;
- bypass a lifecycle gate;
- remove mandatory human accountability;
- destroy provenance, versions or lineage;
- collapse justification into acceptance;
- detach actionable forms from their warrant;
- weaken access, disclosure or integrity controls below the logical obligation;
- substitute technological sophistication for architectural conformance.

Subordinate Documents

KF-3000 is the umbrella architecture: it establishes the whole logical model and is not the place for further detail. Subordinate documents deepen specific parts without revising this one:

- **KF-3001 · Architectural Principles and Invariants** — separating the architectural principles, system invariants and service-interaction constraints presently gathered in §3 and §13.
- **KF-3002 · Capability Model**
- **KF-3003 · Logical Service Catalogue**
- **KF-3004 · Information and Record Model**
- **KF-3005 · Lifecycle and Control Architecture**
- **KF-3006 · Architectural Integration Model** — how the architectural elements collaborate to produce, govern, evolve and retire an Operational Knowledge Object; the integrated systems view that precedes conformance.

- **KF-3007 · Architectural Conformance Framework** — the four proportionate tests (Constitutional, Operational, Integrity, Reconstructability) through which §19 is applied, graded by assurance class.
- **KF-3008 · Architectural Authority Framework**
- **KF-3009 · Architectural Crosswalks and Traceability**

Two cross-programme documents are load-bearing for this architecture and are cited throughout: **KF-2003 · Materiality and Assurance** (the assurance-class scale and the depth to which graded obligations are discharged) and **KF-0002 · Controlled Vocabulary and Architectural Distinctions** (the authoritative terms and the user-facing projection of the architecture's states).

Closing Statement

The Knowledge Foundry Reference Architecture is not a diagram of software components. It is the logical arrangement of responsibilities required to preserve justified knowledge as it moves through identity, evaluation, acceptance, use, change and retirement.

Its purpose is to ensure that every implementation remains answerable to the same constitutional and engineering obligations, regardless of the technologies through which those obligations are realised.

KF-3001 · Architectural Principles and Invariants

Governing the Design and Behaviour of the Knowledge Foundry Architecture

Document Class: Reference Architecture · Parent Document: KF-3000 · Status: Draft · Version: 0.1

```
Canon (KF-1000)
        ↓
Lifecycle and Practices (KF-2001 / KF-2002)
        ↓
Reference Architecture (KF-3000)
        ↓
▸ Architectural Principles and Invariants (KF-3001)
        ↓
Capability, Service and Record Models (KF-3002–KF-3009)
        ↓
Implementation (KF-4000)
```

Purpose

KF-3000 defines the logical architecture required to realise the Knowledge Foundry methodology. KF-3001 defines the rules that govern the design and behaviour of that architecture.

It establishes:

- the **principles** that guide architectural decisions;
- the **invariants** that must remain true across every conforming implementation;
- the **constraints** that govern interactions among architectural services;
- the **mechanisms** — architectural decision and conformance — by which those rules are applied and demonstrated.

KF-3001 does not prescribe technologies, products, deployment topologies or implementation procedures. Its purpose is to ensure that different implementations may vary in form without departing from the constitutional and engineering obligations of the Knowledge Foundry.

It does not repeat the Canon, restate the lifecycle, or reproduce the capability catalogue. Its role is narrower and more technical: to translate constitutional commitment and system obligation into durable rules for architectural design.

The Organising Grammar

The document has three primary forms of architectural rule — principle, invariant and interaction constraint — and two mechanisms through which those rules are applied and demonstrated: architectural decision and conformance.

```
Principle    →  guides design
Invariant    →  defines what must remain true (whenever its scope
applies)
Constraint   →  governs interaction so invariants remain true
```

```
Decision     →  applies the rules to architectural choices
Conformance  →  demonstrates that the rules are satisfied
```

A principle permits several conforming designs; an invariant permits none that violate it within its scope; a constraint keeps an invariant true while the system operates; a decision chooses among designs that all satisfy the rules; and conformance shows that all of it holds. Keeping the three rule forms distinct — and separating them from the two mechanisms that apply and demonstrate them — is what stops the architecture drifting into either **component compliance** (possessing all the named services while letting them bypass one another) or **technology compliance** (using sophisticated tooling while failing to preserve warrant, authority, provenance or history).

Relationship to Upstream Documents

KF-3001 is derived from the Canonical Principles of KF-1000; the lifecycle stages, gates and invariants of KF-2001; the engineering obligations and governance checkpoints of KF-2002; and the architectural capabilities, services and controls of KF-3000.

The derivation chain is:

```
Canonical Principle
        ↓
Lifecycle or Practice Obligation
        ↓
Architectural Risk
        ↓
Architectural Principle
        ↓
System Invariant or Interaction Constraint
        ↓
Conformance Evidence
```

Admission test. A rule belongs in KF-3001 only where violating it could allow an implementation to reproduce the visible structure of KF-3000 while defeating its intended governance, epistemic or historical function. A rule that fails this test is a generic architecture slogan, a restated Canon principle, a data model, a security standard or an implementation guide — and belongs elsewhere.

Part I · Foundations

1. Architectural Rule Types

KF-3001 defines three forms of architectural rule.

1.1 Architectural Principle

A durable design rule that guides architectural choices. A principle constrains design judgement, permits more than one conforming solution, applies across capabilities and services, and is evaluated through the decisions it shapes.

1.2 System Invariant

A condition that must remain true whenever its stated scope and activation conditions apply. An invariant is not satisfied by a one-time check; it is independently testable; and it cannot be waived by implementation convenience. Some invariants are universally active; others apply only when a relevant condition exists (for example, counter-evidence, dependants, or a material application), which its Scope line states.

1.3 Service Interaction Constraint

A rule governing how logical services, actors and records may interact. A constraint protects one or more invariants, assigns or limits architectural responsibility, prevents bypass, silent mutation or collapse of required distinctions, and is enforced through service contracts and controls.

2. Rule Application and Demonstration

The three rule forms are applied to design and shown to hold through two mechanisms. These are not themselves rules.

2.1 Architectural Decision Rule

A rule for selecting between alternative conforming designs. A decision rule makes trade-offs explicit, records the obligations affected, requires justification and approval, and preserves reversibility where practical. Decision rules are specified in Part V.

2.2 Conformance

The demonstration that the principles, invariants and constraints are satisfied: through decisions, mappings and design evidence for principles; through preventive controls, detective checks and tests for invariants; and through service contracts and prohibited-path testing for constraints. Conformance is specified in Part VI. It applies the rules; it is not itself a rule.

3. Relationship Among the Rules

```
Principle
    guides
Architectural Decision
    produces
System Structure
    constrained by
Interaction Constraints
    preserving
System Invariants
```

Principles guide design. Invariants define the required condition. Interaction constraints preserve that condition during operation. Decision rules govern choices where several designs can satisfy the same obligations. Conformance demonstrates the result.

4. Rule Specification

Each **architectural principle** is specified through a Statement, a Rationale, an Upstream basis and its Architectural consequences; where misreading is likely or audit requires it, Prohibited interpretations and Conformance evidence are added. AP-01 is given in full as the exemplar of the complete template.

Each **system invariant** is specified through its Statement and a compact metadata line giving Scope, the obligation it Protects, its Violation condition, and the services that Uphold it. Required evidence is defined generically in Part VI.

Each **interaction constraint** is specified through its Statement and a compact metadata line giving the invariant it Protects, the participating services, and the required record or control. Permitted and prohibited interactions are stated within the constraint.

5. Precedence

Where architectural rules appear to conflict, the following precedence applies:

1. Canonical Principles;
2. human-accountability and formally assigned authority;
3. system invariants;
4. lifecycle gates and standing governance conditions;
5. service interaction constraints;
6. architectural principles;
7. local architectural preferences;
8. implementation convenience.

A lower-order rule may not be used to defeat a higher-order obligation.

Applicable law and binding regulation remain external constraints on the architecture and cannot be displaced by this precedence order. KF-3001 governs architectural precedence, not legal conformance.

Part II · Architectural Principles

Family A · Derivation and Sufficiency

AP-01 · Obligation-Derived Architecture

Statement. Every architectural element must be justified by an upstream Canonical, lifecycle, practice, artefact, checkpoint or conformance obligation.

Rationale. The architecture exists to realise the methodology, not to reproduce familiar technology categories.

Upstream basis. KF-1000; KF-2001; KF-2002; the KF-3000 Architectural Admission Test.

Architectural consequences.

- every capability and service carries an upstream basis;
- technology-led components are admitted only after their logical responsibility is established;
- unsupported services are excluded from the reference architecture;
- crosswalks are maintained from principle to implementation.

Prohibited interpretations.

- a technology belongs because it is commonly used;
- architectural completeness means including every enterprise platform category;
- a powerful component may replace an unfulfilled obligation.

Conformance evidence. Principle-to-practice, practice-to-capability and capability-to-service mappings, together with recorded architectural admission decisions.

AP-02 · Necessary and Proportionate Architecture

Statement. The architecture includes every responsibility necessary to discharge its obligations reliably and excludes responsibilities that cannot be justified by an upstream obligation, dependency or assurance need.

Rationale. Architectural economy means proportionate commitment, not minimum component count. A capability that makes several obligations reliable, scalable or independently verifiable is justified even where it maps to no single lifecycle step.

Upstream basis. The method–architecture–implementation separation of KF-1000 and KF-3000.

Architectural consequences.

- capabilities are logical and non-duplicative;
- an enabling or assurance capability is admitted where a dependency or assurance need justifies it;
- implementation concerns remain in KF-4000;
- services may be combined in implementation where their logical responsibilities remain distinguishable.

AP-03 · Conformance over Structural Resemblance

Statement. Architectural conformance is established by discharged obligations and preserved invariants, not by reproducing a particular component diagram.

Rationale. Different implementations may realise the same logical architecture through different arrangements of people, software and models.

Upstream basis. KF-3000 Conformance Architecture.

Architectural consequences.

- conformance is judged against §19 of KF-3000 and the invariants of this document, not against a reference topology;
- two structurally dissimilar implementations may both conform;
- resemblance to the service catalogue is neither necessary nor sufficient for conformance.

Family B · Separation of Concerns

AP-04 · Preserve Semantic Distinctions

Statement. Concepts that carry different governance or epistemic meanings remain logically distinct even where they share a physical representation. At minimum, the architecture distinguishes: object from representation; object from version; version from release; lifecycle state from epistemic status; justification from acceptance; authority from access permission; application from evolution; retirement from deletion; and service responsibility from human accountability.

Rationale. Collapsing these distinctions permits silent bypass of lifecycle and governance obligations.

Upstream basis. KF-3000 §2 (Architectural Unit) and the separation-of-concerns invariant.

Architectural consequences.

- a shared store may hold two concepts only if each remains independently addressable and changeable;
- no operation may alter one distinguished concept as an undocumented side effect of altering another;
- the information model of KF-3004 must preserve each distinction explicitly.

Prohibited interpretations.

- physical co-location is treated as logical identity;
- a single record type is allowed to stand for both a version and a release, or both justification and acceptance.

AP-05 · Separate Knowledge Content from Its Warrant

Statement. The represented content of an object remains distinguishable from the evidence, rationale, assumptions, evaluation and acceptance that warrant its use.

Rationale. Warrant and content change on different schedules and under different authority; fusing them lets a content edit inherit stale warrant, or a warrant change mutate content.

Upstream basis. KF-2002 B1, B2, C1.

Architectural consequences.

- a content change does not silently inherit prior warrant;
- a warrant change can be represented without mutating object identity;

- actionable forms retain traceable links to their warrant.

AP-06 · Separate Execution from Authority

Statement. The ability to perform an operation does not imply authority to approve it.

Rationale. Execution capacity is a technical property; authority is an institutional one. Conflating them lets whatever can act also decide.

Upstream basis. KF-2002 C1, C2; the human-accountability invariant.

Architectural consequences.

- automated execution may support evaluation, transition or release;
- accountable acceptance, stewardship and designated authorisations remain attributable to an authorised person or body;
- access permission does not itself constitute acceptance authority.

Prohibited interpretations.

- holding write permission to a record is read as holding authority to accept, release or retire it.

Family C · Governance and Accountability

AP-07 · Governance Is Continuous

Statement. Governance conditions remain in force throughout operational life rather than being treated as one-time admission checks.

Rationale. Legitimacy decays; a check passed once does not stay passed.

Upstream basis. KF-2002 C2; the Govern standing gate; Canon Principle IV.

Architectural consequences.

- stewardship, review, compliance and acceptance validity are monitored;
- objects may be suspended when standing conditions lapse;
- conformance checks operate continuously or at defined intervals.

AP-08 · Accountability Remains Human

Statement. Every material governance decision remains attributable to an identifiable human authority or accountable body.

Rationale. Computational systems may act, recommend and record, but they cannot absorb institutional accountability.

Definition. An **accountable body** is a formally constituted human decision-making body whose membership, mandate and decision record are recoverable. A system, an anonymous organisational function or an unminuted committee is not an accountable body.

Upstream basis. KF-1000 Principle III and the human-accountability invariant; KF-2002 C1.

Architectural consequences.

- acceptance, stewardship, release, suspension and retirement each resolve to a human authority or accountable body;
- automation may prepare and execute these decisions but is never their final locus of accountability.

Prohibited interpretations.

- an automated approval step is treated as discharging accountability;
- "the system decided" is accepted as an answer to "who is answerable".

AP-09 · Control Must Produce Evidence

Statement. Every material governance action and transition leaves an inspectable record sufficient to reconstruct what occurred, why it occurred and under whose authority.

Rationale. A control that leaves no evidence cannot be audited and therefore cannot be trusted to have operated.

Upstream basis. KF-2001 transition rules; KF-2002 artefact obligations.

Architectural consequences.

- controls are evidentiary as well as preventive;
- transitions without records are invalid;
- administrative actions affecting governed state are auditable.

Family D · Continuity, Change and History

AP-10 · Identity Persists Through Change

Statement. Changes to representation, state, warrant or operational form do not silently alter object identity.

Rationale. Identity is the thread on which history is strung; if it moves, lineage breaks.

Upstream basis. KF-2002 A1, C3.

Architectural consequences.

- versions inherit object identity;
- branches and consolidations create new identities with explicit lineage;
- identifiers are never reused.

AP-11 · Change Is Additive to History

Statement. Governed change adds to the historical record rather than replacing it.

Rationale. Trust in a knowledge system is a property of its transitions over time; overwriting the past destroys the evidence of trustworthiness.

Upstream basis. KF-2002 C3, D3; Canon Principle VII.

Architectural consequences.

- prior versions and decisions remain recoverable;
- corrections are appended or versioned;
- retirement preserves lineage;
- rollback does not erase the intervening history.

AP-12 · Dependency-Aware Evolution

Statement. Change to an object is assessed in relation to the objects, actionable forms and applications that depend upon it.

Rationale. A correctly versioned object can still leave its dependants silently inconsistent; provenance is the connective tissue that makes changed states, decisions and lineages reconstructable, and dependency is the reason a local change is never only local.

Upstream basis. KF-2002 C3; KF-L001 Literature Landscape (B04 · Governing Knowledge Change).

Architectural consequences.

- dependencies are represented explicitly;
- impact analysis accompanies material change;
- affected dependants are revalidated, suspended or evolved;
- semantic rollback is not treated as a purely local operation.

Family E · Modality, Implementation and Failure

AP-13 · Modality-Independent Obligation

Statement. Governance and epistemic obligations apply regardless of whether knowledge is represented, interpreted or applied through human judgement, symbolic structures, parametric models or hybrid systems.

Rationale. The computational substrate may change while the epistemic obligations remain (KF-1000 Principle V).

Upstream basis. KF-1000 Principle V (modality independence); KF-L001 (B05 · Foundation Models and the Knowledge Sciences).

Architectural consequences.

- the architecture does not assume all knowledge is explicitly inspectable in the same way;
- explanation and provenance requirements are adapted to modality without being discarded;

- opacity is treated as a constraint to be governed, not an exemption.

Prohibited interpretations.

- a modality that resists inspection is granted an exemption from justification, provenance or explanation;
- interpretability is claimed of a representation that cannot support it.

AP-14 · Logical before Physical Design

Statement. Logical responsibility is defined before selecting physical components, technologies or deployment patterns.

Rationale. Technology chosen ahead of responsibility tends to dictate — and quietly narrow — the responsibilities the architecture will admit.

Upstream basis. The logical-architecture boundary of KF-3000.

Architectural consequences.

- services are specified through responsibilities, inputs, outputs, records and controls;
- one logical service may be distributed across several components;
- several logical services may be combined in one component;
- physical co-location does not erase logical separation.

AP-15 · Humans and Machines Form One Governed System

Statement. Human and computational work are architected as one governed system rather than as separate formal and informal layers.

Rationale. An informal human layer outside the governed system is where provenance, attribution and accountability are lost.

Upstream basis. KF-1000 Principle III; the human–AI governance obligations of the Canon.

Architectural consequences.

- human decisions become governed records;
- machine actions remain attributable;
- hand-offs between people and services preserve provenance;
- informal human approval may not substitute for recorded acceptance.

AP-16 · Explicit Failure and Degraded State

Statement. The architecture represents failure, uncertainty and degraded governance explicitly rather than silently treating absence of evidence as evidence of conformance.

Rationale. A governed system must distinguish “condition satisfied” from “condition not checked”, “evidence unavailable” and “control failed”. These are different states, and collapsing them into an implicit pass is how silent failure enters a system that otherwise looks conformant.

Upstream basis. KF-2002 governance checkpoints; KF-3000 control architecture (detective and corrective controls); the continuous-conformance invariant.

Architectural consequences.

- gate outcomes include pass, fail, pending and indeterminate where appropriate;
- service unavailability does not silently produce approval;
- missing evidence is represented as missing, not as satisfied;
- degraded operation is visible and bounded;
- fail-open and fail-closed behaviour is explicitly decided and recorded.

Prohibited interpretations.

- an unreachable check is treated as a pass;
- absence of a violation record is treated as proof that no violation occurred.

Part III · System Invariants

6. Identity and Representation Invariants

INV-01 · Unique Object Identity

Every Operational Knowledge Object has exactly one immutable identity. *Scope:* all objects, all lifecycle states. *Protects:* AP-04, AP-10; Canon Principle I. *Violation:* an identity resolves to more than one object, an object holds more than one active identity, or an identifier is reused. *Upheld by:* SVC-01 (with SVC-03).

INV-02 · Representation Traceability

Every governed representation is traceable to the object and version it represents. *Scope:* all structured representations. *Protects:* AP-04. *Violation:* a representation cannot be resolved to a single object and version. *Upheld by:* SVC-02, SVC-03.

INV-03 · Logical Distinction

Identity, representation, version, release, state, lineage, justification and acceptance remain logically distinguishable. *Scope:* the whole architecture. *Protects:* AP-04, AP-05. *Violation:* two of these collapse such that one cannot be changed or inspected without the other. *Upheld by:* SVC-01, SVC-02, SVC-06, SVC-07, SVC-09, SVC-10.

7. Evidence and Justification Invariants

INV-04 · Provenance Persistence

Every object retains recorded origin and subsequent provenance throughout its lifecycle. *Scope:* capture to historical state. *Protects:* AP-11, AP-12; the provenance invariant. *Violation:* origin or subsequent provenance is lost or unrecoverable. *Upheld by:* SVC-03.

INV-05 · Justification before Operational Reliance

No object enters operational reliance without an inspectable justification meeting the applicable gate. *Scope:* the Justify → Acceptance → Operationalise path. *Protects:* AP-05; Canon Principle VIII. *Violation:* an object is relied upon operationally without an inspectable justification meeting its gate. *Upheld by:* SVC-06, SVC-07, SVC-09.

INV-06 · Counter-Evidence Visibility

Material counter-evidence and unresolved challenge remain attached to the object's epistemic record. *Scope:* objects for which material counter-evidence or unresolved challenge exists. *Protects:* AP-05. *Violation:* material counter-evidence or unresolved challenge is detached, hidden or overwritten. *Upheld by:* SVC-04, SVC-06.

INV-07 · Acceptance Does Not Rewrite Epistemic Status

Operational acceptance may define permitted reliance but may not erase uncertainty, disagreement or evidential weakness. *Scope:* the Acceptance transition and operational life. *Protects:* AP-05; IC-09. *Violation:* acceptance overwrites, hides or resets recorded epistemic status. *Upheld by:* SVC-06, SVC-07.

8. Governance and Accountability Invariants

INV-08 · Attributable Transition

Every lifecycle transition identifies both the service or actor that executed it and, where a governance decision is involved, the human authority or accountable body under whose authority it occurred. *Scope:* all transitions. *Protects:* AP-08, AP-09; Canon Principle III. *Violation:* a transition lacks an identified executor, or a governance-bearing transition lacks an identified human authority or accountable body. *Upheld by:* SVC-09.

INV-09 · Human Accountability

Every material acceptance, stewardship, release, suspension and retirement decision resolves to an identifiable human authority or a formally constituted accountable body. *Scope:* all material governance decisions. *Protects:* AP-08; Canon Principle III and the human-accountability invariant. *Violation:* such a decision lacks an identifiable human authority or accountable body. *Upheld by:* SVC-07, SVC-08, SVC-09, SVC-10 and SVC-14, with SVC-16 governing authority and permission assignments.

INV-10 · Standing Governance

Every operational object has valid stewardship, review status and applicable governance conditions in force. *Scope:* all operational objects. *Protects:* AP-07; Canon Principles IV and VII. *Violation:* an operational object is ownerless, unreviewed or out of compliance. *Upheld by:* SVC-08.

INV-11 · Authorised Change

No governed record or state is materially altered outside authorised service and role boundaries. *Scope:* all governed records and states. *Protects:* AP-06, AP-09. *Violation:* a governed record or state is altered outside its authorised service and role. *Upheld by:* SVC-16 (with every service).

9. Change and Historical Invariants

INV-12 · Version Preservation

Every prior governed version remains recoverable after change. *Scope:* all versioned objects. *Protects:* AP-11; the version-preservation invariant. *Violation:* a prior version becomes unrecoverable. *Upheld by:* SVC-10.

INV-13 · Lineage Preservation

Branching, consolidation, supersession, retirement and reactivation preserve explicit predecessor and successor relationships. *Scope:* all change and closure operations. *Protects:* AP-10, AP-11. *Violation:* a predecessor or successor link is lost or made ambiguous. *Upheld by:* SVC-10 (with SVC-14).

INV-14 · No Silent Mutation

Every material change produces a change, version or correction record. *Scope:* all governed records and states. *Protects:* AP-09, AP-11. *Violation:* a material change leaves no change, version or correction record. *Upheld by:* SVC-10 (with every service).

INV-15 · Dependency Disposition

No materially changed object returns to operational use until affected dependencies have been identified and dispositioned. *Scope:* evolution of objects that have dependants. *Protects:* AP-12. *Violation:* a materially changed object returns to use with dependants unidentified or undispositioned. *Upheld by:* SVC-11 (with SVC-09).

INV-16 · Retirement Preserves History

Retirement removes an object from active operational use without destroying its identity, records or lineage. *Scope:* all retirements. *Protects:* AP-11; Canon Principle VII. *Violation:* retirement destroys identity, records or lineage. *Upheld by:* SVC-14.

10. Operational Invariants

INV-17 · Warranted Actionable Form

Every actionable form traces to a specific object version, justification and valid acceptance-state. *Scope:* all actionable forms eligible for release or use. *Protects:* AP-05; IC-12. *Violation:* a releasable or usable actionable form has no traceable link to version, justification and acceptance. *Upheld by:* SVC-12.

INV-18 · Governed Application

No suspended, retired, unaccepted or governance-invalid object is used in a governed application. *Scope:* all governed applications. *Protects:* AP-07, AP-16. *Violation:* such an object is used in a governed application. *Upheld by:* SVC-13. The architecture distinguishes the ways this can fail — prevention failure, detection failure and deliberate override — so that a violation is diagnosable rather than merely flagged.

INV-19 · Outcome Traceability

Every material application outcome can be traced to the object version, actionable form and use context that produced it. *Scope:* all material applications. *Protects:* AP-09; the traceable-use invariant. *Violation:* a material outcome cannot be so traced. *Upheld by:* SVC-13.

INV-20 · Modality-Appropriate Explainability

Every material application supports an account appropriate to its modality, risk and decision context. *Scope:* all material applications. *Protects:* AP-13; Canon Principle IX. *Violation:* a material application supports no account appropriate to its modality, risk and context. *Upheld by:* SVC-13.

11. Assurance Invariants

INV-21 · Inspectable Conformance

The architecture can produce evidence showing whether each required capability, gate and invariant is satisfied. *Scope:* the whole architecture. *Protects:* AP-09. *Violation:* the architecture cannot show whether a required capability, gate or invariant holds. *Upheld by:* SVC-15.

INV-22 · Assurance Independence

A service may not be the sole judge of the validity of the records or controls it produces. *Scope:* all conformance judgements. *Protects:* IC-18. *Violation:* a service certifies its own records or controls with no independent check. *Upheld by:* SVC-15.

INV-23 · Audit Non-Interference

Assurance and audit operations do not alter the governed records under examination. *Scope:* all assurance and audit operations. *Protects:* IC-17. *Violation:* an assurance or audit operation alters a governed record under examination. *Upheld by:* SVC-15.

INV-24 · No Silent Success

An unmet, unevaluated or indeterminate required condition is never represented as satisfied. *Scope:* all required conditions, gates and checks. *Protects:* AP-16, AP-09. *Violation:* an unmet, unevaluated or indeterminate condition is represented as satisfied — for example, service unavailability yields approval, or missing evidence is read as a pass. *Upheld by:* SVC-15 (with SVC-09 for gate outcomes).

Part IV · Service Interaction Constraints

12. State and Transition Constraints

IC-01 · Lifecycle State Authority

Only the Lifecycle and Transition Service may alter the recorded lifecycle state of an object. Other services may request, recommend or trigger a transition but may not change state directly. *Protects:* INV-08, INV-11. *Participants:* SVC-09; all requesting services. *Required control:* every state change carries a Transition Record attributable to SVC-09.

IC-02 · Gate Enforcement

A requested transition may execute only where the applicable gate has been evaluated and the gate decision is recorded as part of the same governed transition operation. *Protects:* INV-05, INV-08, INV-24. *Participants:* SVC-09 (with SVC-15 assurance). *Required control:* the Gate Decision Record and Transition Record are transactionally bound, share the same transition identifier, and cannot produce a completed transition independently of one another.

IC-03 · Suspension Ownership

The Lifecycle and Transition Service owns the suspension transition and record. Triggering services provide the reason and evidence. *Protects:* INV-08, INV-10. *Participants:* SVC-09; triggers from SVC-08, SVC-11, SVC-15. *Required control:* a Suspension Record (SVC-09) references its Suspension Recommendation or reason (originating service).

IC-04 · Reactivation through Evolution

A retired or suspended object may return to operational use only through the applicable Evolution, evaluation, justification and acceptance path. *Protects:* INV-05, INV-16. *Participants:* SVC-09, SVC-14; via SVC-10, SVC-05, SVC-06, SVC-07. *Required control:* reactivation produces a Transition Record and re-passes the Acceptance transition; SVC-14 supplies the Restoration Package.

13. Identity, Version and Lineage Constraints

IC-05 · Identity Immutability

No service may modify or reuse an assigned object identifier. *Protects:* INV-01. *Participants:* SVC-01; all services. *Required control:* identifier assignment is write-once and append-only.

IC-06 · Version Immutability

No service may overwrite a released or governed version. Corrections create a new version or an append-only correcting record. *Protects:* INV-12, INV-14. *Participants:* SVC-10; all services. *Required control:* version records are immutable; corrections are versioned or appended.

IC-07 · Branch and Consolidation Identity

Branching or consolidation creates successor object identities and explicit lineage links; it does not mutate predecessor identity. *Protects:* INV-01, INV-13. *Participants:* SVC-10 (with SVC-01). *Required control:* a Branch or Consolidation Record with predecessor and successor links.

IC-08 · Release Separation

Creating a version does not automatically release it. Release requires the applicable acceptance and governance conditions. *Protects:* INV-05, INV-17; AP-04. *Participants:* SVC-10, SVC-07. *Required control:* a Release Record distinct from the Version Record, linked to an Acceptance Record.

14. Evidence, Acceptance and Use Constraints

IC-09 · Justification–Acceptance Separation

The Justification Service establishes and records warrant. The Authority and Acceptance Service records authorised reliance. Neither may silently substitute for the other. *Protects:* INV-05, INV-07; AP-05. *Participants:* SVC-06 and SVC-07 (co-responsible); SVC-15 (assurance). *Required control:* Justification Record and Acceptance Record are distinct and separately attributable; the boundary is verified by assurance, not enforced by either peer against the other.

IC-10 · Operationalisation Eligibility

The Operationalisation Service may prepare a provisional actionable form from a non-operational object only where it is clearly marked non-releasable. It may release a form, or make it eligible for use, only when the form is linked to an accepted object version. *Protects:* INV-17. *Participants:* SVC-12 (checks SVC-07). *Required control:* a Warrant Linkage Record binds any releasable or usable form to an accepted version; provisional forms are marked non-releasable.

IC-11 · Application Eligibility

The Application and Outcome Service must verify acceptance, release and governance status before material use. *Protects:* INV-18. *Participants:* SVC-13 (checks SVC-07, SVC-08, SVC-09). *Required control:* an eligibility check recorded in the Application Record.

IC-12 · Warrant Link Preservation

No service may detach an actionable form or application record from the object version and warrant on which it depends. *Protects:* INV-17, INV-19; AP-05. *Participants:* SVC-12 and SVC-13 (co-responsible), with SVC-03 maintaining traceability and SVC-06 the warrant. *Required control:* warrant links are non-nullable and versioned with the form.

15. Governance and Access Constraints

IC-13 · Authority Is Not Permission

Possession of technical permission does not confer evaluation, acceptance, release, stewardship or retirement authority. *Protects:* INV-09, INV-11; AP-06. *Participants:* SVC-16, SVC-07. *Required control:* permission assignments and authority assignments are separate records.

IC-14 · Least Necessary Alteration

A service may alter only the records and states assigned to its logical responsibility. *Protects:* INV-11. *Participants:* all services. *Required control:* each record class names its authoritative maintainer (KF-3000 §24); alteration outside it is rejected and logged.

IC-15 · Material Administrative Action

Changes to authority, permissions, stewardship, retention, release or governance state are recorded and attributable. *Protects:* INV-09, INV-11. *Participants:* SVC-16 (with SVC-08). *Required control:* an Access and Administrative Action Log entry per material change.

IC-16 · Automated Authority Boundary

Automated services may recommend and execute authorised operations but may not become the final accountable authority where human authority is required. *Protects:* INV-09; AP-08. *Participants:* SVC-07, SVC-08; all automated services. *Required control:* the accountable authority on a material decision resolves to a human role or accountable body.

16. Assurance Constraints

IC-17 · Read-Only Assurance Boundary

The Conformance and Audit Service may inspect governed records and control evidence but may not alter the records it evaluates or authorise the transitions it verifies. *Protects:* INV-23. *Participants:* SVC-15. *Required control:* assurance operates through read-only access to the records under examination.

IC-18 · No Sole Self-Certification

Conformance may not rely solely on assertions produced by the service or actor being assessed. Each material conformance judgement must include evidence generated independently of the assessed service's own conformance assertion — such as record inspection, cross-service reconciliation, control testing or external review. *Protects:* INV-22. *Participants:* SVC-15; all assessed services. *Required control:* each conformance judgement cites at least one independently generated source of the kinds named above.

IC-19 · Exception Visibility

A failed control, waived requirement or unresolved exception remains visible until corrected, formally accepted as risk, or rendered irrelevant by retirement. *Protects:* INV-21, INV-24. *Participants:* SVC-15 (with SVC-08). *Required control:* an Exception Register entry persists until one of the three closure conditions is recorded.

IC-23 · Proportionate Discharge

The rigour to which a material obligation is discharged — gate exit depth, control strength, required evidence and reconstruction depth — is a function of the object's assurance class (KF-2003), applied at the release / scope-of-reliance to which the class attaches. Higher assurance classes require deeper discharge; lower classes require less. **The lifecycle invariants are not graded:** single identity, attributable transitions, justified transitions, inspectable transitions, provenance and preserved lineage hold identically at every class. Only the depth of evidence and review scales. *Protects:* AP-02 (Necessary and Proportionate Architecture); the six lifecycle invariants (ungraded). *Participants:* SVC-08, SVC-18, and every gate-executing service. *Required control:* each gate and control resolves the object's assurance class and applies the minimum obligations KF-2003 fixes for that class, never less than the ungraded-invariant floor.

17. Provenance and Record Constraints

IC-20 · Provenance Continuity

Every material hand-off between actors or services preserves origin, derivation and attribution links. *Protects:* INV-04, INV-19. *Participants:* SVC-03; all services at hand-off. *Required control:* provenance links are carried through, not regenerated, at each hand-off.

IC-21 · Append-Only Correction

A material governed record may be corrected, but the original record and the correction relationship remain recoverable. *Protects:* INV-12, INV-14. *Participants:* all services maintaining records; SVC-10 for versioned corrections. *Required control:* corrections append; they do not overwrite.

IC-22 · Time Semantics Preservation

Every governed record carries recorded time and, where relevant, effective time; where the underlying event occurred at a third distinct time, event time is also preserved. (*Recorded time* is when the system received the fact; *effective time* is when it is valid in the represented domain; *event time* is when the real-world event occurred.) *Protects:* INV-14; the Common Record Properties of KF-3000 §14.8; aligns with KF-3004. *Participants:* all services. *Required control:* recorded time is mandatory on every governed record; effective and event time are recorded where they differ.

IC-24 · Event-Native Evidence

Conformance evidence is emitted by the governed act, not authored as a separate artefact. The record a governed act produces is its conformance evidence: an acceptance record evidences acceptance, a transition record is the audit event, a validation is the test result, a provenance link evidences derivation, a supersession link evidences lineage. A distinct evidence artefact is admitted only where the act's own record cannot carry the required proof. This constraint bounds the cost of assurance and prevents the accumulation of evidence-about-evidence. *Protects:* INV-22; the conformance-evidence obligation (CTRL-CNF-01, KF-3005). *Participants:* all services producing governed records; SVC-18 reads without re-authoring. *Required control:* the governed act's own record is registered as the conformance evidence for that act; a separate

artefact is produced only against a recorded justification that the act's record is insufficient. Aligns with KF-3004 §5.2.

Part V · Architectural Decision Rules

18. Decision Record Requirement

Every material architectural decision records: the question being decided; the upstream obligations affected; alternatives considered; the invariants and constraints involved; trade-offs accepted; the accountable decision-maker; the effective date; and the review or reversal condition.

19. Decision Tests

A proposed architectural decision must pass the following tests.

- **19.1 Obligation test.** Which upstream obligation does the decision support?
- **19.2 Invariant test.** Could the decision cause any invariant to cease to hold within its scope?
- **19.3 Separation test.** Does the decision collapse concepts that KF-3000 requires to remain distinct?
- **19.4 Accountability test.** Does it obscure who is answerable?
- **19.5 Traceability test.** Can the resulting actions and records be reconstructed?
- **19.6 Reversibility test.** Can the decision be revised without destroying history or lineage?
- **19.7 Modality test.** Does the design remain valid across the computational modalities it claims to support?
- **19.8 Failure test.** Does the design represent failure, unavailability and indeterminacy explicitly rather than as silent success?
- **19.9 Sufficiency test.** Does the design discharge the obligation without introducing unnecessary architectural commitment?

20. Variations, Exceptions and Non-Conformance

Three cases are distinguished, and they are not interchangeable.

20.1 Architectural Variation

A design that differs from the reference form but continues to satisfy all principles, invariants and constraints is a conforming variation. It requires an Architectural Decision Record, not an exception.

20.2 Principle Exception

A time-bounded departure from an architectural principle may be permitted where: the affected principle is identified; no Canonical Principle or system invariant is violated; the rationale and risk are recorded; the exception is approved by an identified authority; its scope

and duration are bounded; compensating controls are stated; and a review or expiry condition is defined.

20.3 Non-Conformance

A design or operation that violates a Canonical Principle, a system invariant or a mandatory interaction constraint is non-conforming. It cannot be converted into conformity merely by recording an exception. A proposed exception that cannot preserve an invariant is non-conformance, not an architectural variation.

Part VI · Conformance

21. Principle Conformance

Principle conformance is demonstrated through architectural decisions, mappings and design evidence showing that the principle shaped the architecture.

22. Invariant Conformance

Invariant conformance is demonstrated through preventive controls; detective checks; representative test cases; record inspection; exception and violation reporting; and evidence that the invariant remains true over time within its scope. This is the *Required evidence* referenced by each invariant in Part III.

23. Interaction Conformance

Interaction conformance is demonstrated through service responsibility matrices; logical service contracts; transition and authorisation tests; audit records; prohibited-path testing; and evidence that services cannot silently bypass the required interaction constraints.

24. Required Conformance Artefacts

A conforming architecture maintains: an Architectural Principle Register; an Invariant Register; a Service Interaction Constraint Register; an Assurance Class Register (recording each release's assigned class and overlays, per KF-2003); an Architectural Decision Record Register; an Exception Register; an Invariant Test Register; a Principle-to-Capability crosswalk; an Invariant-to-Control crosswalk; and an Interaction Constraint-to-Service crosswalk.

Part VII · Crosswalks

25. Principle-to-Upstream Crosswalk

Principle	Principal upstream basis
AP-01	KF-1000; KF-2001; KF-2002; KF-3000 Admission Test
AP-02	Method–architecture–

Principle	Principal upstream basis
	implementation separation
AP-03	KF-3000 Conformance Architecture
AP-04	KF-3000 Architectural Unit (§2)
AP-05	KF-2002 B1, B2, C1
AP-06	KF-2002 C1, C2; human-accountability invariant
AP-07	KF-2002 C2; Govern standing gate
AP-08	KF-1000 Principle III; KF-2002 C1
AP-09	KF-2001 transition rules; KF-2002 artefact obligations
AP-10	KF-2002 A1, C3
AP-11	KF-2002 C3, D3; Canon Principle VII
AP-12	KF-2002 C3; KF-L001 (B04)
AP-13	KF-1000 Principle V; KF-L001 (B05)
AP-14	KF-3000 logical-architecture boundary
AP-15	KF-1000 Principle III; human–AI governance obligations
AP-16	KF-2002 governance checkpoints; KF-3000 control architecture

26. Invariant-to-Service Crosswalk

Invariant family	Principal services
Identity and representation (INV-01–03)	SVC-01, SVC-02, SVC-03
Evidence and justification (INV-04–07)	SVC-04, SVC-05, SVC-06
Governance and accountability (INV-08–11)	SVC-07, SVC-08, SVC-09, SVC-10, SVC-14, SVC-16
Change and history (INV-12–16)	SVC-09, SVC-10, SVC-11, SVC-14
Operational use (INV-17–20)	SVC-12, SVC-13
Assurance (INV-21–24)	SVC-15 (with SVC-09 for INV-24)

27. Interaction-Constraint-to-Service Crosswalk

Each constraint names the service that principally enforces it and the services that participate. (The detailed service contracts that carry these constraints are specified in KF-3003.)

Constraint	Principal enforcing service	Participating services
IC-01 · Lifecycle State Authority	SVC-09	all requesting services
IC-02 · Gate Enforcement	SVC-09	SVC-15 (assurance)

Constraint	Principal enforcing service	Participating services
IC-03 · Suspension Ownership	SVC-09	SVC-08, SVC-11, SVC-15
IC-04 · Reactivation through Evolution	SVC-09	SVC-14, SVC-10, SVC-05, SVC-06, SVC-07
IC-05 · Identity Immutability	SVC-01	all services
IC-06 · Version Immutability	SVC-10	all services
IC-07 · Branch and Consolidation Identity	SVC-10	SVC-01
IC-08 · Release Separation	SVC-10	SVC-07
IC-09 · Justification–Acceptance Separation	SVC-06 and SVC-07 (co-responsible)	SVC-15 (assurance)
IC-10 · Operationalisation Eligibility	SVC-12	SVC-07
IC-11 · Application Eligibility	SVC-13	SVC-07, SVC-08, SVC-09
IC-12 · Warrant Link Preservation	SVC-12 and SVC-13 (co-responsible)	SVC-03, SVC-06
IC-13 · Authority Is Not Permission	SVC-16	SVC-07
IC-14 · Least Necessary Alteration	SVC-16	all services
IC-15 · Material Administrative Action	SVC-16	SVC-08
IC-16 · Automated Authority Boundary	SVC-07	SVC-08; all automated services
IC-17 · Read-Only Assurance Boundary	SVC-15	—
IC-18 · No Sole Self-Certification	SVC-15	all assessed services
IC-19 · Exception Visibility	SVC-15	SVC-08
IC-20 · Provenance Continuity	SVC-03	all services at hand-off
IC-21 · Append-Only Correction	SVC-10	all record-maintaining services
IC-22 · Time Semantics Preservation	all services	—

Relationship to Subsequent Documents

KF-3001 governs the documents that follow.

- **KF-3002** must derive capabilities consistent with these principles and sufficient to preserve all system invariants and enforce all interaction constraints.

- **KF-3003** must define service responsibilities and contracts that enforce the interaction constraints, and complete the constraint-to-contract detail behind §27.
- **KF-3004** must define records capable of proving invariant and transition conformance, and fix the recorded/effective/event time semantics of IC-22.
- **KF-3005** must encode gates and lifecycle controls without permitting bypass, including the transactional binding of gate decision and transition (IC-02).
- **KF-3008** must define authority and accountability, kept distinct from the technical permission assigned by SVC-16.
- **KF-3007** must define tests and evidence for principle, invariant and interaction conformance.
- **KF-3009** must preserve end-to-end traceability across all mappings.

KF-4000 may choose technologies and physical designs freely within these boundaries.

Closing Statement

Architectural freedom exists within constitutional constraint. The Knowledge Foundry Reference Architecture does not require every implementation to look the same. It requires every implementation to preserve the same essential distinctions, responsibilities, histories and lines of accountability.

KF-3001 defines those non-negotiable conditions, and fixes the grammar — principle, invariant, constraint; applied by decision, shown by conformance — in which the remainder of the architecture series is written.

KF-3002 · Capability Model

The Required Abilities of a Governed Knowledge Architecture

Document Class: Reference Architecture · Parent Document: KF-3000 · Governed By: KF-3001 · Status: Draft · Version: 0.2

```
Canon and Method
KF-1000 / KF-2001 / KF-2002
        ↓
Reference Architecture
KF-3000
        ↓
Architectural Rules
KF-3001
        ↓
▸ Capability Model
KF-3002
        ↓
Logical Services, Records and Controls
KF-3003–KF-3009
        ↓
Implementation
KF-4000
```

Purpose

KF-3002 defines the architectural capabilities required to realise the Knowledge Foundry methodology and preserve the principles, invariants and interaction constraints established in KF-3001.

A capability is an enduring ability the architecture must possess, independent of organisational workflow, accountable role, logical service arrangement, information-storage model, technology, vendor, deployment topology or computational modality.

KF-3002 answers: *What must a conforming Knowledge Foundry architecture be able to do?* KF-3003 answers: *Through which logical services are those abilities exposed and coordinated?* KF-3004 answers: *Through which records and relationships are those abilities represented and evidenced?* KF-3005 answers: *Through which lifecycle controls are those abilities constrained?*

Capability Derivation

Every capability must be traceable to an upstream obligation.

```
Canonical Principle
        ↓
Lifecycle or Practice Obligation
        ↓
Architectural Principle, Invariant or Constraint
        ↓
Required Architectural Ability
```

```
        ↓
Capability
        ↓
Logical Service
        ↓
Implementation
```

A capability belongs in KF-3002 only when its absence would leave a Canonical Principle, a lifecycle stage/gate/invariant, an engineering-practice obligation, a mandatory engineering artefact, a KF-3001 invariant or interaction constraint, or a conformance requirement unsupported.

Capability Admission Test

A proposed capability is admitted only where it expresses an enduring architectural ability; is independent of any specific technology or organisational procedure; has an identifiable upstream basis; has observable required behaviours; has identifiable information objects and relationships; would create a defined architectural risk on failure; has demonstrable conformance; and is not already fully contained within another capability.

Capability Boundaries

A capability is not a lifecycle stage, an engineering practice, a logical service, a human role, an artefact, a control, an application feature or a technology component. The relationships are:

```
Practice   establishes an engineering obligation
Capability states the ability required to discharge it
Service    exposes and coordinates that ability
Record     preserves its governed information and evidence
Control    limits or verifies its exercise
Role       holds authority or accountability where required
```

Part I · Capability Model Foundations

1. Capability Domains

The model contains five capability domains and twenty-four constituent capabilities.

Domain	Purpose	Capabilities	Principal upstream basis
A · Identity & Representation	Establish and preserve what the object is and how it can be inspected	3	KF-2002 Domain A
B · Evidence & Justification	Establish and preserve the warrant for belief	4	KF-2002 Domain B

Domain	Purpose	Capabilities	Principal upstream basis
C · Operational Governance	Govern authority, custody, versioning, release, change and transition	8	KF-2002 Domain C
D · Operational Use	Make knowledge actionable and learn from its use	4	KF-2002 Domain D
E · Assurance, Protection & Continuity	Preserve provenance, integrity, conformance and explicit operating condition across all domains	5	KF-3000 and KF-3001

Domains A–D describe the abilities that act upon the Operational Knowledge Object and its operational life. Domain E is cross-cutting: **it does not author, evaluate or alter the substantive content of an Operational Knowledge Object, nor does it independently authorise lifecycle transitions. It preserves and evaluates the provenance, integrity, temporal condition, protection and conformance evidence surrounding that content.**

2. Capability Levels

KF-3002 uses two levels. A **domain capability** is a broad architectural ability grouping related subordinate capabilities (for example, *Operational Governance*). A **constituent capability** is a bounded, independently testable architectural ability (for example, *Authority and Acceptance Management*). KF-3002 does not decompose below constituent capability level; detailed service operations belong in KF-3003 and implementation functions in KF-4000.

3. Capability Specification

Each constituent capability is specified through eleven fields:

- **Capability statement** — the enduring ability;
- **Purpose** — why the ability is required;
- **Upstream derivation** — practices, principles, invariants and constraints;
- **Scope** — what falls inside and outside the capability;
- **Required behaviours** — what the architecture must be able to do;
- **Principal information objects** — the logical information structures acted upon or produced;
- **Capability relationships** — typed relationships to other capabilities;
- **Protected invariants** — KF-3001 invariants preserved;
- **Operating criticality** — the consequence of the capability's inability or degradation;
- **Failure conditions** — what inability or misuse looks like;
- **Conformance condition** — observable proof that the capability exists.

A **principal information object** is a logical information structure created, consumed or maintained by a capability. It may be an engineering artefact, governed record, representation, decision, status, linkage or result. It is not necessarily an Operational Knowledge Object; the OKO is the governed subject, while information objects are the structures through which capabilities act upon and evidence it.

Operating criticality classifies the consequence of a capability being unavailable or degraded, using the categories: *conformance-critical*, *governance-critical*, *operationally critical*, *recoverable with bounded manual control*, and *advisory*. A capability may combine a category with a default failure posture (fail-open or fail-closed). This classification does not set implementation availability targets; it lets KF-3005 and KF-3007 determine the consequence of unavailability.

4. Capability Relationship Types

Capabilities relate through five typed relationships, replacing any single undifferentiated notion of dependency:

- **Prerequisite** — an ability that must already exist or produce an input before this one can operate;
- **Collaborating capability** — an ability participating with this one in the same governed operation;
- **Constraining capability** — an ability that limits or validates this one's execution;
- **Triggered capability** — an ability invoked as a consequence of this one;
- **Evidence provider / consumer** — an ability supplying, or receiving, conformance evidence.

Typing these relationships removes false circularity: two capabilities that *collaborate* in one operation are not mutual prerequisites. Relationships do not create authority; authority is assigned through KF-3008.

5. Common Capability Requirements

Every capability must:

1. preserve the relevant KF-3001 invariants;
2. operate within the applicable interaction constraints;
3. produce or maintain inspectable evidence of material actions;
4. expose failure, pending and indeterminate conditions explicitly;
5. preserve attribution between human and computational action;
6. operate across supported computational modalities;
7. preserve recorded time and effective time where relevant;
8. preserve the authoritative-maintainer boundary assigned to each information object in KF-3003 and KF-3004, altering such information only through a governed interaction;
9. remain logically distinguishable even where combined physically;
10. support conformance inspection.

Part II · Domain A — Identity & Representation

Domain Statement

The architecture must be able to establish what an Operational Knowledge Object is, preserve its identity and make its representation and context inspectable.

A-CAP-01 · Object Registration

Capability statement. The architecture can register a candidate Operational Knowledge Object and assign it one durable, immutable identity, and can determine how an intake relates to existing objects.

Purpose. Establish the object as a governable entity from the moment of capture, and route the intake correctly without itself performing version or lineage operations.

Upstream derivation. KF-2002 A1; KF-3001 AP-04, AP-10; INV-01; IC-05.

Scope. Includes candidate registration; identity assignment; initial origin capture; duplicate and prior-object assessment; classification of intake as a new object, version candidate, branch candidate or consolidation candidate. Excludes detailed semantic structuring; acceptance; release; the creation of the resulting version, branch or consolidated successor. **A-CAP-01 determines whether an intake should create a new object identity or be routed to an existing object's version or lineage path; it does not itself create the resulting version, branch or consolidated successor** (those are C-CAP-04 and C-CAP-06). This preserves the distinction: identity resolution ≠ version creation ≠ lineage operation.

Required behaviours.

- assign an immutable object identifier;
- prevent identifier reuse;
- distinguish object identity from record identity;
- determine whether intake represents a new object or relates to an existing one;
- bind origin and intake context to the identity;
- route version, branch and consolidation candidates to the responsible capability;
- expose ambiguous or unresolved identity conditions;
- prevent progression where identity is absent or conflicting.

Principal information objects. Object Identity Record; Intake Record; Provenance Seed; Identity Resolution Decision.

Capability relationships. - *Constrained by:* E-CAP-03, E-CAP-04 - *Evidence providers:* E-CAP-01 - *May trigger:* C-CAP-04, C-CAP-06

Protected invariants. INV-01; INV-04; INV-11; INV-24.

Operating criticality. Governance-critical; normally fail-closed — no progression without a resolved, unique identity.

Failure conditions. Duplicate identity; reused identifier; unidentified object; origin absent; unresolved ambiguity represented as resolved; version or branch incorrectly registered as an unrelated object.

Conformance condition. For any registered object, an independent reviewer can recover its identity, origin, intake decision and, where applicable, its relationship to predecessor or source objects.

A-CAP-02 · Structured Representation

Capability statement. The architecture can create and maintain an inspectable representation of an object's semantic content, type, constituent elements and relationships.

Purpose. Enable evaluation, justification, governance, discovery and explanation to operate on explicit representations.

Upstream derivation. KF-2002 A2; KF-3001 AP-04, AP-05, AP-13; INV-02; INV-03.

Scope. Includes semantic typing; constituent claim or element representation; metadata; object relationships; source-to-representation linkage; multiple representations of one object version. Excludes evidence evaluation; justification; operational acceptance; physical representation technology.

Required behaviours.

- assign and maintain semantic type;
- represent constituent claims or elements;
- represent relationships among elements and objects;
- preserve the distinction between content and warrant;
- support human, symbolic, parametric and hybrid representations;
- indicate representation limitations and opacity;
- link every representation to the object and version represented.

Principal information objects. Structured Representation; Semantic Type Record; Metadata Record; Object Relationship Record; Representation Limitation Record.

Capability relationships. - *Prerequisites:* A-CAP-01 - *Constrained by:* E-CAP-03 - *Evidence providers:* E-CAP-01 - *May trigger:* C-CAP-04

Protected invariants. INV-02; INV-03; INV-04; INV-20.

Operating criticality. Governance-critical — inspectability is a precondition of evaluation, justification and governance.

Failure conditions. Representation not linked to object or version; structure implicit and unrecoverable; type absent or contradictory; semantic relationship lost; modality limitations concealed; content and justification collapsed.

Conformance condition. A reviewer can identify what the representation expresses, which object and version it represents, how its material elements relate and what limitations apply to its modality.

A-CAP-03 · Object Resolution and Contextual Discovery

Capability statement. The architecture can locate and retrieve an object together with the governed context required to interpret its identity, state, warrant, relationships and history.

Purpose. Ensure that retrieval does not detach content from its governance and epistemic context.

Upstream derivation. KF-2002 A1, A2; KF-3001 AP-05, AP-09; INV-02, INV-03, INV-04; IC-12.

Scope. Includes discovery by object identity, version, semantic type, relationship, lifecycle state, epistemic status, steward, lineage, and acceptance or release condition. Excludes ranking algorithms; user-interface design; physical indexes; implementation-specific search technology.

Required behaviours.

- resolve an identifier to exactly one object;
- retrieve current and historical versions;
- expose lifecycle and epistemic status separately;
- traverse object relationships and lineage;
- retrieve relevant warrant and governance context;
- distinguish current, superseded, branched, consolidated, suspended and retired objects;
- prevent context-free retrieval from being represented as governed knowledge.

Principal information objects. Resolution Result; Governed Context Package; Relationship Path; Lineage Path; Retrieval Provenance Record.

Capability relationships. - *Prerequisites:* A-CAP-01, A-CAP-02 - *Constrained by:* E-CAP-03 - *Evidence providers:* B-CAP-04, C-CAP-08, E-CAP-01

Protected invariants. INV-01; INV-02; INV-03; INV-04; INV-17.

Operating criticality. Recoverable with bounded manual control — discovery may degrade for a bounded period without blocking governed transitions, provided degradation is visible.

Failure conditions. Ambiguous resolution; stale version presented as current; content retrieved without its relevant acceptance or warrant; retired or suspended status concealed; lineage or relationship path broken.

Conformance condition. Given a valid retrieval request, the architecture can return the correct object and enough governed context to determine what it is, which version is being inspected, its present standing and the basis on which it may be relied upon.

Part III · Domain B — Evidence & Justification

Domain Statement

The architecture must be able to represent, evaluate and preserve the grounds on which an object may be believed without conflating evidential quality, justification and operational acceptance.

B-CAP-01 · Evidence Registration and Relationship

Capability statement. The architecture can register evidence and counter-evidence as governed records and relate them explicitly to claims, objects and assessments.

Purpose. Make evidential support, challenge and origin independently inspectable.

Upstream derivation. KF-2002 B1, B2; KF-3001 AP-05; INV-04; INV-06; IC-20.

Scope. Includes evidence and counter-evidence registration; evidence-to-claim and evidence-to-object linkage; evidence classification (support, challenge, context); evidence provenance; evidence availability status. Excludes quality evaluation (B-CAP-02); justification assembly (B-CAP-03); acceptance; adjudication of the truth of the evidence.

Required behaviours.

- assign evidence-record identity;
- capture evidence origin and type;
- distinguish support, challenge and context;
- relate evidence to specific claims or objects;
- preserve evidence independently from the conclusion drawn from it;
- retain conflicting evidence;
- identify missing or unavailable evidence explicitly.

Principal information objects. Evidence Record; Counter-Evidence Record; Evidence-to-Claim Link; Evidence Classification; Evidence Availability Status.

Capability relationships. - *Prerequisites:* A-CAP-01, A-CAP-02 - *Constrained by:* E-CAP-03 - *Evidence providers:* E-CAP-01

Protected invariants. INV-04; INV-06; INV-24.

Operating criticality. Governance-critical — warrant depends on inspectable evidence.

Failure conditions. Evidence embedded but not identifiable; evidence detached from its source; counter-evidence hidden or deleted; support relationship stated without basis; unavailable evidence represented as favourable evidence.

Conformance condition. For each material claim, the architecture can recover the evidence and counter-evidence related to it, their origins, roles and present availability.

B-CAP-02 · Quality Evaluation

Capability statement. The architecture can assess the quality and fitness of a structured object or evidence item against explicit criteria and record the resulting verdict.

Purpose. Determine whether an object is suitable to proceed toward justification.

Upstream derivation. KF-2002 B1; KF-3001 AP-09, AP-16; INV-21; INV-24.

Scope. Includes application of evaluation criteria; assessment of quality, consistency, completeness and confidence; recording of pass, fail, return, pending and indeterminate verdicts; retention of verdicts and reasons. Excludes justification construction (B-CAP-03); acceptance (C-CAP-01); the definition of organisational thresholds, which belong to the applicable governance or implementation profile.

Required behaviours.

- apply defined evaluation criteria;
- preserve criteria and applicable thresholds;
- evaluate quality, consistency, completeness and confidence;
- distinguish pass, fail, return, pending and indeterminate;
- retain all verdicts and reasons;
- route failed or incomplete assessments appropriately;
- prevent unevaluated conditions from appearing as passed.

Principal information objects. Evaluation Criteria Record; Quality Assessment; Evaluation Verdict; Evaluation Exception; Confidence Record.

Capability relationships. - *Prerequisites:* A-CAP-02, B-CAP-01 - *Constrained by:* E-CAP-04 - *Evidence providers:* E-CAP-05 - *Provides evidence to:* E-CAP-02

Protected invariants. INV-05; INV-21; INV-24.

Operating criticality. Governance-critical — the quality gate precedes justification.

Failure conditions. Undocumented criteria; verdict without reasons; pending assessment treated as passed; low-quality object advanced; evaluation altered after the fact without history.

Conformance condition. For any evaluation, a reviewer can recover the criteria, inputs, evaluator, result, confidence, reasons and disposition.

B-CAP-03 · Justification Construction and Inspection

Capability statement. The architecture can assemble, preserve and expose an inspectable warrant connecting claims to evidence, counter-evidence, rationale and assumptions.

Purpose. Establish why an object may be believed without claiming that justification guarantees truth.

Upstream derivation. KF-2002 B2; KF-3001 AP-05; INV-05; INV-06; IC-09.

Scope. Includes claim-to-evidence linkage; assumptions and rationale; the end-to-end traceability chain; the admission and reliance rationale for parametrically represented capability; preservation of unresolved challenges. Excludes acceptance (C-CAP-01); evidence registration (B-CAP-01); quality evaluation (B-CAP-02); any claim that justification guarantees truth.

Required behaviours.

- relate claims to supporting and defeating evidence;
- record assumptions and rationale;
- preserve unresolved challenges;
- create an end-to-end traceability chain;
- support independent inspection;
- preserve an inspectable admission and reliance rationale for parametric capability;
- prevent acceptance from substituting for justification.

Principal information objects. Justification Record; Assumption Record; Rationale Record; Traceability Chain; Challenge Record.

Capability relationships. - *Prerequisites:* B-CAP-01, B-CAP-02 - *Constrained by:* E-CAP-03 - *Evidence providers:* E-CAP-01 - *May trigger:* C-CAP-01

Protected invariants. INV-04; INV-05; INV-06; INV-07.

Operating criticality. Governance-critical — justification is the warrant on which reliance rests.

Failure conditions. Rationale unsupported; assumptions hidden; counter-evidence omitted; traceability incomplete; accepted status treated as epistemic proof; modality opacity used as an exemption from warrant.

Conformance condition. An independent reviewer can reconstruct the material reasons for and against belief, identify the assumptions and trace each material claim to its evidential basis.

B-CAP-04 · Epistemic Status Management

Capability statement. The architecture can represent and maintain the epistemic standing of an object independently from its lifecycle, release and acceptance states.

Purpose. Preserve uncertainty, dispute and evidential weakness even where operational reliance is permitted.

Upstream derivation. KF-2002 B1, B2, C1; KF-3001 AP-04, AP-05; INV-03; INV-07; IC-09.

Scope. Includes representation of epistemic conditions from unassessed through insufficiently supported; confidence and uncertainty; contestation; attributable status-change events; prior epistemic states. Excludes lifecycle state (C-CAP-08); acceptance (C-CAP-01); release (C-CAP-05).

Required behaviours.

- represent unassessed, quality-assessed, warrant-established, contested, warrant-weakened and insufficiently supported conditions;
- preserve confidence and uncertainty;
- record unresolved challenges;
- distinguish epistemic status from acceptance;
- update status only through attributable evidence or assessment events;
- preserve prior epistemic states.

Principal information objects. Epistemic Status Record; Confidence Record; Contestation Record; Epistemic Status Change Record.

Capability relationships. - *Prerequisites:* B-CAP-01, B-CAP-02, B-CAP-03 - *Collaborates with:* C-CAP-01 - *Evidence providers:* E-CAP-01

Protected invariants. INV-03; INV-06; INV-07; INV-14.

Operating criticality. Governance-critical — protects the separation of epistemic standing from acceptance.

Failure conditions. Acceptance overwrites evidential weakness; uncertainty reduced without evidence; dispute removed without disposition; lifecycle state used as proxy for epistemic status.

Conformance condition. The architecture can show an object's present and historical epistemic standing separately from whether it is accepted, released or operationally active.

Part IV · Domain C — Operational Governance

Domain Statement

The architecture must be able to control authority, stewardship, review, versioning, release, change, dependency and transition throughout the operational life of an object. Version, acceptance and release are kept as distinct capabilities because they occur at different points and are governed by different obligations.

C-CAP-01 · Authority and Acceptance Management

Capability statement. The architecture can record and govern the decision by which an identified authority permits reliance upon a justified object.

Purpose. Separate epistemic warrant from institutional permission to rely.

Upstream derivation. KF-2002 C1; KF-3001 AP-06, AP-08; INV-05, INV-08, INV-09, INV-17; IC-09, IC-13.

Scope. Includes acceptance, provisional-acceptance and refusal decisions; authority identification and scope verification; the scope, conditions and duration of reliance; decision rationale. Excludes justification construction (B-CAP-03); permission assignment (E-CAP-03); version creation (C-CAP-04); release designation (C-CAP-05); stewardship (C-CAP-02).

Required behaviours.

- identify the accepting authority or accountable body;
- verify that the authority's mandate, scope and effective period apply to the decision;
- record acceptance, provisional acceptance or refusal;
- define scope, conditions and duration of reliance;
- preserve decision rationale;
- prevent acceptance where mandatory prerequisites are absent;
- route refusal or conditional acceptance appropriately.

Principal information objects. Acceptance Record; Refusal Record; Scope-of-Reliance Record; Authority Assignment Reference; Acceptance Condition Record.

Capability relationships. - *Prerequisites:* B-CAP-03, B-CAP-04, E-CAP-03 - *Collaborates with:* C-CAP-08 - *Evidence providers:* E-CAP-01 - *May trigger:* C-CAP-05, C-CAP-08

Protected invariants. INV-05; INV-08; INV-09; INV-17; INV-24.

Operating criticality. Governance-critical; normally fail-closed — no acceptance without an applicable authority.

Failure conditions. Acceptance without an identified applicable authority; acceptance substituting for justification; technical permission mistaken for acceptance authority; acceptance where mandatory prerequisites are absent; refusal neither routed nor recorded.

Conformance condition. Every accepted object carries a decision attributable to an authority or accountable body whose mandate, scope and effective period were applicable, together with the reliance scope, conditions, date and supporting justification.

C-CAP-02 · Stewardship and Responsibility Management

Capability statement. The architecture can assign, maintain and transfer stewardship and responsibility for an operational object.

Purpose. Ensure sustained human accountability throughout operational life.

Upstream derivation. KF-2002 C2; KF-3001 AP-08; INV-09; INV-10; INV-11.

Scope. Includes steward assignment and transfer; identification of accountable roles; stewardship history; the mandate and membership of an accountable body where used. Excludes acceptance decisions (C-CAP-01); review scheduling (C-CAP-03); access-permission assignment (E-CAP-03).

Required behaviours.

- assign a steward before operational release;
- identify accountable roles;
- transfer stewardship without loss of history;
- prevent ownerless operational state;
- distinguish service execution from human accountability;
- preserve mandate and membership where an accountable body is used.

Principal information objects. Stewardship Assignment; Responsibility Record; Stewardship History; Accountable Body Record.

Capability relationships. - *Prerequisites:* A-CAP-01 - *Collaborates with:* C-CAP-08 - *Constrained by:* E-CAP-03 - *Evidence providers:* E-CAP-01

Protected invariants. INV-09; INV-10; INV-11.

Operating criticality. Governance-critical — no operational object may be ownerless.

Failure conditions. Ownerless operational object; stewardship transfer that loses history; service execution mistaken for accountability; accountable body without recoverable mandate or membership.

Conformance condition. For every operational object and material governance decision, the architecture can identify the responsible steward and applicable human authority or accountable body.

C-CAP-03 · Review and Compliance Management

Capability statement. The architecture can schedule, execute and record reviews and determine whether standing governance conditions remain satisfied.

Purpose. Preserve legitimacy as evidence, policy and context change.

Review determines whether the object's warrant, utility and governance standing remain adequate. Compliance determines whether applicable formal obligations are satisfied. A review may include compliance evaluation, but one does not substitute for the other.

Upstream derivation. KF-2002 C2; KF-3001 AP-07, AP-16; INV-10, INV-18, INV-21, INV-24.

Scope. Includes review scheduling; due and overdue detection; evaluation of applicable policy and governance requirements; recording of review outcomes, compliance findings and exceptions; triggering of suspension, evolution or retirement. Excludes the transition execution itself (C-CAP-08); acceptance (C-CAP-01); the definition of external policy.

Required behaviours.

- establish review cycles;
- detect due and overdue reviews;
- evaluate applicable policy and governance requirements;
- record review outcomes, compliance assessments and exceptions;
- trigger suspension, evolution or retirement;
- prevent expired governance conditions from remaining silently valid.

Principal information objects. Review Schedule; Review Record; Compliance Assessment; Compliance Finding; Governance Exception; Review Trigger.

Capability relationships. - *Prerequisites:* C-CAP-02 - *Collaborates with:* C-CAP-08 - *Constrained by:* E-CAP-04 - *May trigger:* C-CAP-06, C-CAP-08 - *Provides evidence to:* E-CAP-02

Protected invariants. INV-10; INV-18; INV-21; INV-24.

Operating criticality. Governance-critical; recoverable within a defined period — a late review is bounded and must be visible as overdue.

Failure conditions. Expired governance condition remaining silently valid; overdue review undetected; compliance breach unrecorded; review outcome without disposition.

Conformance condition. The architecture can establish whether every operational object has a valid steward, review status and applicable governance conditions in force.

C-CAP-04 · Version Management

Capability statement. The architecture can create, preserve and resolve immutable versions of an Operational Knowledge Object without altering its enduring identity.

Purpose. Preserve every state of represented content and warrant as an immutable, recoverable version, distinct from acceptance and release. A version must exist before an object can be evaluated, justified, accepted or released; version creation is therefore prior to, not consequent on, acceptance.

Upstream derivation. KF-2002 C2, C3; KF-3001 AP-04, AP-11; INV-03, INV-12, INV-14; IC-06.

Scope. Includes creation of candidate and governed versions; preservation of prior versions; the version-to-identity relation; predecessor and successor identification; correction and supersession relationships; the distinction of version state from acceptance and release state. Excludes the acceptance decision (C-CAP-01); release designation (C-CAP-05); lineage and evolution operations (C-CAP-06).

Required behaviours.

- create candidate and governed versions;
- preserve prior versions;
- relate versions to their object identity;

- prevent overwrite;
- identify predecessor and successor versions;
- preserve correction and supersession relationships;
- distinguish version state from acceptance and release state.

Principal information objects. Version Record; Version State; Predecessor–Successor Link; Correction Record.

Capability relationships. - *Prerequisites:* A-CAP-01, A-CAP-02 - *Collaborates with:* E-CAP-05 - *Constrained by:* E-CAP-03 - *Evidence providers:* E-CAP-01

Protected invariants. INV-03; INV-12; INV-14; and constraint IC-06.

Operating criticality. Governance-critical; fail-closed — a prior version is never overwritten.

Failure conditions. Version overwritten; version detached from object identity; predecessor or successor relationship lost; version state conflated with release state.

Conformance condition. For any object, a reviewer can enumerate its immutable versions, their order, and their correction and supersession relationships, distinct from whether any version is accepted or released.

C-CAP-05 · Release Management

Capability statement. The architecture can designate an accepted version as eligible for a defined scope and period of operational reliance.

Purpose. Separate controlled operational release from both version creation and acceptance, so that creating a version does not release it and accepting a version does not automatically make every successor released.

Upstream derivation. KF-2002 C2; KF-3001 AP-04, AP-06; INV-05, INV-17; IC-08.

Scope. Includes verification of acceptance; binding of release to a specific immutable version; release scope and effective period; prevention of automatic release inheritance; withdrawal, suspension and supersession of releases; release history. Excludes version creation (C-CAP-04); the acceptance decision (C-CAP-01); actionable-form release (D-CAP-01).

Required behaviours.

- verify acceptance;
- bind release to a specific immutable version;
- define release scope and effective period;
- prevent automatic inheritance of release by a new version;
- withdraw, suspend or supersede releases;
- preserve release history.

Principal information objects. Release Record; Release Scope; Effective Period; Withdrawal Record; Supersession Record.

Capability relationships. - *Prerequisites:* C-CAP-01, C-CAP-04 - *Collaborates with:* C-CAP-08 - *Constrained by:* E-CAP-03 - *Evidence providers:* E-CAP-05

Protected invariants. INV-05; INV-17; and constraint IC-08.

Operating criticality. Governance-critical; normally fail-closed — no release without a bound, accepted version.

Failure conditions. Release without acceptance; a new version inheriting release automatically; release not bound to a specific version; withdrawal that deletes history.

Conformance condition. For every released form, the architecture can identify the immutable version released, its acceptance basis, scope, effective period and predecessor or successor releases.

C-CAP-06 · Lineage and Evolution Management

Capability statement. The architecture can govern amendment, branching, consolidation, supersession and reactivation while preserving history and requiring renewed warrant.

Purpose. Make semantic change a controlled and attributable operation.

Upstream derivation. KF-2002 C3; KF-3001 AP-10, AP-11; INV-12, INV-13, INV-14, INV-16; IC-06, IC-07.

Scope. Includes change triggers and rationale; creation of successor versions or objects; preservation of branch, consolidation and supersession lineage; correction history; routing changed states through evaluation, justification and acceptance. Excludes version-record mechanics (C-CAP-04); dependency impact (C-CAP-07); the acceptance decision itself (C-CAP-01).

Required behaviours.

- record change triggers and rationale;
- create successor versions or objects appropriately;
- preserve branch, consolidation and supersession lineage;
- distinguish object evolution from object replacement;
- return changed states through evaluation, justification and acceptance;
- preserve correction history;
- prevent silent mutation.

Principal information objects. Change Record; Lineage Record; Branch Record; Consolidation Record; Correction Record; Reactivation Proposal.

Capability relationships. - *Prerequisites:* C-CAP-04 - *Collaborates with:* C-CAP-07, C-CAP-08 - *Evidence providers:* E-CAP-01 - *May trigger:* B-CAP-02, B-CAP-03, C-CAP-01

Protected invariants. INV-12; INV-13; INV-14; INV-16; and constraints IC-06, IC-07.

Operating criticality. Governance-critical — silent mutation and lineage loss are the failures it exists to prevent.

Failure conditions. Silent mutation; lineage lost on branch or merge; predecessor identity mutated; change without renewed warrant; correction that overwrites the corrected record.

Conformance condition. For any current state, a reviewer can reconstruct its predecessors, changes, rationales, responsible actors and renewed evaluation, justification and acceptance.

C-CAP-07 · Dependency Impact Management

Capability statement. The architecture can identify and disposition objects, actionable forms and applications affected by a proposed or completed change.

Purpose. Prevent locally correct change from silently invalidating dependent knowledge.

Upstream derivation. KF-2002 C3; KF-3001 AP-12; INV-13, INV-15, INV-18, INV-24.

Scope. Includes representation of material dependencies; determination of impacted dependants; impact classification; disposition as revalidation, evolution, suspension or no-action; preservation of impact decisions. Excludes the change execution (C-CAP-06); the application itself (D-CAP-02); discovery of dependencies beyond governed objects.

Required behaviours.

- represent material dependencies;
- determine impacted dependants;
- classify impact;
- require revalidation, evolution, suspension or no-action disposition;
- preserve impact decisions;
- prevent return to operational use where material dependencies remain undispositioned.

Principal information objects. Dependency Record; Dependency Impact Record; Revalidation Requirement; Dependency Disposition Record; Suspension Recommendation.

Capability relationships. - *Prerequisites:* A-CAP-03 - *Collaborates with:* C-CAP-06 - *May trigger:* C-CAP-06, C-CAP-08

Protected invariants. INV-13; INV-15; INV-18; INV-24.

Operating criticality. Governance-critical; normally fail-closed — a changed object does not return to use with undispositioned dependants.

Failure conditions. Dependants unidentified; a changed object returning to use with undispositioned dependants; impact decision unrecorded; material dependency representation absent.

Conformance condition. For every material change, the architecture can identify affected dependants and show their recorded disposition before the changed object returns to operational use.

C-CAP-08 · Event, Gate and Transition Management

Capability statement. The architecture can register lifecycle events, evaluate gates and execute attributable state transitions as one governed operation.

Purpose. Preserve the controlled movement of objects through the lifecycle. Operational control determines whether a transition may execute; independent assurance (E-CAP-02) later determines whether the control operated correctly, and is therefore an evidence consumer, not a prerequisite of the transition.

Upstream derivation. KF-2001 lifecycle states, events, gates and transitions; KF-3001 AP-09, AP-16; INV-08, INV-09, INV-10, INV-14, INV-24; IC-01, IC-02.

Scope. Includes event registration; gate identification and eligibility evaluation; transactional binding of gate decision and transition; separate attribution of execution and authority; forward, return, suspension and retirement paths. Excludes the definition of gate criteria (KF-3005); the content of the acceptance decision (C-CAP-01); dependency disposition (C-CAP-07).

Required behaviours.

- register triggering events;
- determine applicable current and proposed states;
- identify the required gate;
- evaluate transition eligibility;
- bind Gate Decision and Transition Records transactionally under one transition identifier;
- attribute execution and accountable authority separately;
- support forward, return, suspension and retirement paths;
- prevent invalid or indeterminate gates from silently passing.

Principal information objects. Event Record; Gate Decision Record; Transition Record; Transition Warrant; Transition Status.

Capability relationships. - *Prerequisites:* applicable gate criteria and current state - *Collaborates with:* C-CAP-01, C-CAP-03, C-CAP-05, C-CAP-06, C-CAP-07, D-CAP-04 - *Constrained by:* E-CAP-03, E-CAP-04 - *Evidence providers:* E-CAP-01, E-CAP-05 - *Provides evidence to:* E-CAP-02

Protected invariants. INV-08; INV-09; INV-10; INV-14; INV-24.

Operating criticality. Governance-critical; normally fail-closed — an indeterminate or unavailable gate does not silently pass.

Failure conditions. Lifecycle state changed outside this capability; gate decision and transition not atomically bound; indeterminate gate silently passing; transition unattributed.

Conformance condition. For every transition, the architecture can recover the event, prior state, proposed state, applicable gate, gate result, execution actor, accountable authority and transition outcome under one transition identifier.

Part V · Domain D — Operational Use

Domain Statement

The architecture must be able to derive usable forms from accepted objects, govern their application and return application outcomes to the lifecycle.

D-CAP-01 · Actionable Form Derivation and Release

Capability statement. The architecture can derive actionable forms from governed objects and control when those forms become eligible for operational use.

Purpose. Convert represented knowledge into usable forms without severing warrant. Form-level release governed here is distinct from object-level release (C-CAP-05).

Upstream derivation. KF-2002 D1; KF-3001 AP-05; INV-17, INV-18; IC-10, IC-12.

Scope. Includes derivation of actionable forms; the provisional-versus-released distinction; source-version linkage; warrant and acceptance linkage of released forms; withdrawal or replacement on source change. Excludes application and use (D-CAP-02); acceptance (C-CAP-01); the object's own versioning (C-CAP-04).

Required behaviours.

- derive rules, playbooks, procedures, prompts, patterns, models and other actionable forms;
- distinguish provisional form from released form;
- link each form to its source object version;
- link released forms to justification and acceptance;
- prevent provisional forms from operational use;
- identify forms affected by source change;
- withdraw or replace forms when source standing changes.

Principal information objects. Actionable Form; Derivation Record; Warrant Linkage; Form Release Record; Form Status.

Capability relationships. - *Prerequisites:* C-CAP-01, C-CAP-04 - *Collaborates with:* C-CAP-05 - *Constrained by:* E-CAP-03 - *Evidence providers:* E-CAP-01

Protected invariants. INV-17; INV-18; and constraints IC-10, IC-12.

Operating criticality. Operationally critical; normally fail-closed for release — a form is not made usable without a warrant linkage.

Failure conditions. Form severed from warrant; provisional form used operationally; form not linked to a source version; stale form used after source change.

Conformance condition. Every operationally eligible actionable form traces to an accepted object version and carries an inspectable release and warrant linkage.

D-CAP-02 · Controlled Application and Explanation

Capability statement. The architecture can govern material use of actionable forms and preserve an explanation appropriate to the modality, risk and decision context.

Purpose. Ensure that use remains authorised, traceable and explainable.

Upstream derivation. KF-2002 D2; KF-3001 AP-13; INV-17, INV-18, INV-20; IC-11, IC-12, IC-13.

Scope. Includes eligibility verification; recording of material application context; identification of human and computational participants; preservation of the input-to-result path; modality-appropriate explanation; enforcement of disclosure and use restrictions. Excludes outcome capture (D-CAP-03); the form derivation (D-CAP-01); user-interface design.

Required behaviours.

- verify object, version, form, release and governance eligibility before use;
- record material application context;
- identify human and computational participants;
- preserve relevant input-to-result path;
- provide explanation adequate to the applicable modality, materiality, risk and decision context;
- detect use of invalid or withdrawn forms;
- enforce disclosure and use restrictions.

Principal information objects. Application Record; Use Context Record; Eligibility Decision; Explanation Record; Application Participant Record.

Capability relationships. - *Prerequisites:* D-CAP-01 - *Collaborates with:* C-CAP-08 - *Constrained by:* C-CAP-03, E-CAP-03 - *Evidence providers:* E-CAP-01 - *May trigger:* D-CAP-03

Protected invariants. INV-17; INV-18; INV-20; and constraints IC-11, IC-12, IC-13.

Operating criticality. Operationally critical; normally fail-closed for use — an ineligible object is not applied.

Failure conditions. Use without an eligibility check; application for which no explanation adequate to the applicable modality, materiality, risk and decision context can be produced; use of a withdrawn or invalid form; disclosure restriction bypassed; use detached from warrant.

Conformance condition. For each material application, the architecture can identify what was used, why it was eligible, who or what participated, the context of use, the resulting action or recommendation, and an explanation proportionate to the applicable risk and modality.

D-CAP-03 • Outcome and Feedback Capture

Capability statement. The architecture can capture application outcomes and return them as governed evidence, review triggers or change signals.

Purpose. Close the lifecycle from operational use back to evaluation and evolution.

Upstream derivation. KF-2002 D2; KF-3001 AP-11; INV-06, INV-19, INV-24.

Scope. Includes attachment of outcomes to application and object version; the distinction of observation from interpretation; preservation of adverse, contradictory and null outcomes; materiality assessment under an applicable profile; assignment of a disposition; aggregation preserving individual provenance. Excludes the application itself (D-CAP-02); evolution execution (C-CAP-06); the definition of materiality thresholds, which belong to the applicable profile.

Required behaviours.

- attach outcomes to application and object version;
- distinguish observed result from interpretation;
- preserve adverse, contradictory and null outcomes;
- assess outcome materiality under an applicable profile;
- assign a disposition — no action, additional observation, investigation, temporary restriction, suspension, review, evolution, retirement or continued use;
- raise the corresponding lifecycle trigger where required;
- aggregate outcomes while preserving individual provenance.

Principal information objects. Outcome Record; Interpretation Record; Feedback Record; Materiality Assessment; Disposition Record; Lifecycle Trigger.

Capability relationships. - *Prerequisites:* D-CAP-02 - *Evidence providers:* E-CAP-01 - *May trigger:* C-CAP-03, C-CAP-06, C-CAP-08

Protected invariants. INV-06; INV-19; INV-24.

Operating criticality. Operationally critical; recoverable within a defined period — outcome capture may lag briefly, provided the gap is visible and bounded.

Failure conditions. Outcomes not captured; observation and interpretation conflated; adverse outcomes suppressed; aggregation that erases provenance; outcomes without disposition.

Conformance condition. Every material outcome is traceable to the application, actionable form and object version involved and has a recorded disposition under the applicable governance profile.

D-CAP-04 • Retirement and Historical Preservation

Capability statement. The architecture can conclude operational use of an object while preserving identity, record integrity, lineage and the possibility of governed reactivation.

Purpose. Remove knowledge from active reliance without destroying institutional memory.

Upstream derivation. KF-2002 D3; KF-3001 AP-11; INV-09, INV-13, INV-16, INV-18; IC-04.

Scope. Includes retirement of the Operational Knowledge Object from active operational eligibility; determination and recording of retirement reason and authority; preservation of the full historical record; maintenance of successor and supersession links; retrieval as historical material; governed reactivation. **Withdrawal of a particular release or actionable form without retiring the object remains governed by C-CAP-05 and D-CAP-01 respectively**; retirement is reserved for the object as a whole. Excludes destructive deletion; the evolution path itself (C-CAP-06); acceptance (C-CAP-01).

Required behaviours.

- determine and record retirement reason;
- identify retirement authority;
- remove the object and its forms from active eligibility;
- preserve the full historical record;
- maintain successor and supersession links;
- support retrieval as historical material;
- permit reactivation only through governed evolution;
- prevent destructive deletion of governed history.

Principal information objects. Retirement Record; Historical Object Record; Preservation Record; Reactivation Request; Successor Link.

Capability relationships. - *Prerequisites:* A-CAP-01 - *Collaborates with:* C-CAP-08 - *Constrained by:* E-CAP-03 - *Evidence providers:* E-CAP-05 - *May trigger:* C-CAP-06 (reactivation)

Protected invariants. INV-09; INV-13; INV-16; INV-18; and constraint IC-04.

Operating criticality. Governance-critical — destructive deletion of governed history is prohibited.

Failure conditions. Retirement without a recorded reason or authority; destructive deletion; lineage lost; a retired object remaining operationally usable; reactivation outside governed evolution.

Conformance condition. A retired object cannot be used operationally, but its identity, versions, warrant, governance history, applications, outcomes and lineage remain recoverable.

Part VI · Domain E — Assurance, Protection & Continuity

Domain Statement

The architecture must be able to preserve the provenance, integrity, operating condition and conformance evidence required for Domains A–D to remain trustworthy across time and computational modality. Domain E preserves and evaluates governed context; it does not author or alter substantive content, nor independently authorise transitions.

E-CAP-01 · Provenance and End-to-End Traceability

Capability statement. The architecture can preserve origin, derivation, custody, attribution and relationship traceability across every material object, record, transition and application.

Purpose. Make current states and historical change reconstructable.

Upstream derivation. KF-2002 A1, A2, B2, C2, C3, D1; KF-3001 AP-11, AP-12; INV-02, INV-04, INV-08, INV-13, INV-17, INV-19, INV-24; IC-20.

Scope. Includes preservation of source origin; why-, where- and how-provenance; traceability across representations, changes, actionable forms, applications and outcomes; provenance across modality hand-offs; explicit identification of provenance gaps. Excludes substantive content (Domains A–D); access control (E-CAP-03); the authorising of transitions (C-CAP-08).

Required behaviours.

- preserve source origin;
- capture why-, where- and how-provenance where applicable;
- trace representations to sources and objects;
- trace changes to actors, evidence and events;
- trace actionable forms to warrant;
- trace applications and outcomes to the forms used;
- preserve provenance across human, symbolic, parametric and hybrid hand-offs;
- identify incomplete or unavailable provenance explicitly.

Principal information objects. Provenance Record; Custody Record; Derivation Link; Attribution Link; Traceability Path; Provenance Gap Record.

Capability relationships. - *Prerequisites:* A-CAP-01 - *Constrained by:* E-CAP-03 - *Provides evidence to:* all capabilities (E-CAP-01 is a primary evidence provider across the model)

Protected invariants. INV-02; INV-04; INV-08; INV-13; INV-17; INV-19; INV-24.

Operating criticality. Governance-critical and conformance-critical; fail-closed at hand-offs — a material hand-off that cannot preserve provenance is not silently completed.

Failure conditions. Broken lineage; provenance regenerated rather than carried through a hand-off; a provenance gap concealed rather than represented; attribution lost at a hand-off.

Conformance condition. For a representative material outcome, the architecture can reconstruct the path from source and object creation through evaluation, acceptance, operationalisation, use and outcome, while exposing any known provenance gaps.

E-CAP-02 · Conformance and Independent Assurance

Capability statement. The architecture can determine and demonstrate whether required capabilities, invariants, gates, controls and interaction constraints are satisfied.

Purpose. Prevent architectural compliance from depending solely on structural resemblance or self-assertion. E-CAP-02 receives evidence from all capabilities and may raise findings, exceptions, suspension recommendations or corrective-action triggers. It does not participate in the substantive decision or service action it later assesses, and is never an operational prerequisite of the operations it evaluates.

Upstream derivation. KF-3001 Part VI; KF-3001 AP-03; INV-21, INV-22, INV-23, INV-24; IC-17, IC-18.

Scope. Includes inspection of required records and control evidence; invariant and prohibited-path testing; cross-service reconciliation; recording of findings, exceptions and violations; the distinction of conformance, variation, exception and non-conformance. Excludes altering the records under examination; authorising transitions; acting as sole judge of its own outputs; participating as a prerequisite in the operations it assesses.

Required behaviours.

- inspect required records and control evidence;
- test invariant conditions;
- test prohibited paths;
- reconcile evidence across services;
- preserve assurance independence;
- record findings, exceptions and violations;
- distinguish conformance, variation, exception and non-conformance;
- avoid altering records under examination.

Principal information objects. Conformance Assessment; Invariant Test Result; Constraint Test Result; Assurance Finding; Non-Conformance Record; Exception Reference.

Capability relationships. - *Evidence consumers:* receives evidence from all capabilities - *May trigger:* C-CAP-08 (via suspension recommendation), E-CAP-04 - *Constrained by:* its own read-only and independence rules (IC-17, IC-18)

Protected invariants. INV-21; INV-22; INV-23; INV-24.

Operating criticality. Conformance-critical but not transaction-blocking — assurance must exist and be independent, but its unavailability does not by itself halt governed operations; it renders them unproven until reconciled.

Failure conditions. Self-certification; assurance that alters the records examined; non-conformance recorded as a mere exception; an indeterminate condition treated as a pass; assurance made a prerequisite of the operation it audits.

Conformance condition. The architecture can produce independently inspectable evidence for every required capability and report unmet, indeterminate or violated conditions without modifying the evidence assessed.

E-CAP-03 • Access, Authority Boundary and Integrity Protection

Capability statement. The architecture can restrict access and material action according to assigned permissions while preserving the distinction between permission and authority.

Purpose. Protect governed records and operations against unauthorised access, alteration or disclosure.

Upstream derivation. KF-3000 §1 access, disclosure and integrity obligations; KF-3001 AP-06; INV-08, INV-09, INV-11, INV-23; IC-13, IC-14, IC-15, IC-16.

Scope. Includes actor and service identification; permission assignment and revocation; the permission-versus-authority distinction; record-integrity protection; disclosure and use restriction; administrative-action recording; permission and authority history. Excludes acceptance-authority decisions (C-CAP-01); specific security technologies (KF-4000); the content of the governance decision.

Required behaviours.

- identify actors and services;
- assign and revoke permissions;
- distinguish technical access from governance authority;
- enforce view, create, amend, accept, release, suspend and retire permissions;
- protect record integrity;
- enforce disclosure and use restrictions;
- record material administrative action;
- preserve permission and authority history.

Principal information objects. Access Policy; Permission Assignment; Authority Assignment Reference; Disclosure Constraint; Integrity Status; Administrative Action Record.

Capability relationships. - *Collaborates with:* C-CAP-02 - *Constrains:* all capabilities performing a material action - *Evidence providers:* E-CAP-05

Protected invariants. INV-08; INV-09; INV-11; INV-23; and constraints IC-13, IC-14, IC-15, IC-16.

Operating criticality. Governance-critical; normally fail-closed — an unauthorised material action is refused, not permitted by default.

Failure conditions. Permission mistaken for authority; unauthorised alteration; record integrity compromised; material administrative action unrecorded; disclosure restriction bypassed.

Conformance condition. For every material action, the architecture can demonstrate that the actor possessed the required permission and, where applicable, that a separate valid authority decision existed.

E-CAP-04 • Operating Condition, Exception and Degraded-State Management

Capability statement. The architecture can represent normal, failed, pending, indeterminate and degraded conditions explicitly and govern their consequences.

Purpose. Prevent absent evidence, failed controls or unavailable services from appearing as successful operation.

Upstream derivation. KF-3001 AP-16; INV-10, INV-18, INV-21, INV-24; IC-19.

Scope. Includes the distinction of passed, failed, pending and indeterminate conditions; representation of service and control degradation; explicit fail-open or fail-closed decision; exception raising, tracking and bounding; compensating controls; corrective triggering. Excludes the corrective transition execution (C-CAP-08); the definition of organisational risk acceptance (governance).

Required behaviours.

- distinguish passed, failed, pending and indeterminate conditions;
- represent service or control degradation;
- determine fail-open or fail-closed behaviour explicitly;
- prevent silent success;
- raise and track exceptions;
- bound duration and scope of authorised exceptions;
- preserve compensating controls;
- trigger suspension or corrective action where required.

Principal information objects. Operating Condition Record; Degradation Record; Exception Record; Compensating Control Record; Recovery Record; Failure Disposition Record.

Capability relationships. - *Collaborates with:* C-CAP-08 - *May trigger:* C-CAP-08 (suspension or corrective action) - *Provides evidence to:* E-CAP-02

Protected invariants. INV-10; INV-18; INV-21; INV-24; and constraint IC-19.

Operating criticality. Conformance-critical and governance-critical — it is the capability that determines fail-safe behaviour, so its own degradation defaults to fail-closed.

Failure conditions. Silent success; an unavailable service presumed available; an exception without bound or expiry; a degraded state concealed; missing evidence read as a pass.

Conformance condition. When a required condition cannot be evaluated or a service is degraded, the architecture records the condition explicitly and applies the predefined operational consequence rather than treating it as satisfied.

E-CAP-05 · Temporal and Historical Continuity

Capability statement. The architecture can preserve the temporal meaning, sequence and authenticity of governed records and states across time.

Purpose. Maintain trust where events, records and effective conditions occur at different times.

Upstream derivation. KF-2002 C2, D3; KF-3001 AP-11; INV-04, INV-12, INV-13, INV-14, INV-16; IC-21, IC-22.

Scope. Includes preservation of recorded, effective and event time; sequencing of transitions and decisions; retention of superseded and corrected records; authenticity across custody and format change; as-of historical reconstruction; retention and disposition. Excludes the version mechanics (C-CAP-04); storage technology; substantive content.

Required behaviours.

- preserve recorded time;
- preserve effective time;
- preserve event time where distinct;
- sequence transitions and decisions;
- retain superseded and corrected records;
- preserve authenticity through custody and format change;
- support historical reconstruction as of a specified time;
- enforce retention and disposition conditions.

Principal information objects. Temporal Metadata Record; Effective Period; Event-Time Record; Custody History; Retention Condition; Historical Reconstruction Result.

Capability relationships. - *Collaborates with:* C-CAP-04, C-CAP-06 - *Evidence providers:* E-CAP-01 - *Provides evidence to:* E-CAP-02

Protected invariants. INV-04; INV-12; INV-13; INV-14; INV-16; and constraints IC-21, IC-22.

Operating criticality. Governance-critical; recoverable with bounded manual control — temporal reconstruction may require manual assembly for a bounded period, but must never rely on overwritten records.

Failure conditions. Temporal semantics collapsed; effective time lost; as-of reconstruction relying on overwritten current records; authenticity broken across a format change; retention condition unenforced.

Conformance condition. The architecture can reconstruct the object's represented content, warrant, acceptance, governance and operational state at a specified historical point without relying on overwritten current records.

Part VII · Capability Relationships

6. Principal Eligibility Progression

The principal eligibility progression follows the governed life of an object from registration to retirement or evolution:

```
A-CAP-01 Object Registration
        ↓
A-CAP-02 Structured Representation
        ↓
C-CAP-04 Version Management
        ↓
B-CAP-01 Evidence Registration
        ↓
B-CAP-02 Quality Evaluation
        ↓
B-CAP-03 Justification Construction
        ↓
B-CAP-04 Epistemic Status Management
        ↓
C-CAP-01 Authority and Acceptance
        ↓
C-CAP-02 Stewardship Assignment
        ↓
C-CAP-05 Release Management
        ↓
D-CAP-01 Actionable Form Derivation and Release
        ↓
D-CAP-02 Controlled Application
        ↓
D-CAP-03 Outcome and Feedback
        ↓
C-CAP-06 Evolution   or   D-CAP-04 Retirement
```

This represents the principal eligibility progression of an object. It is neither a complete lifecycle diagram nor a mandatory procedural workflow. **C-CAP-08 governs every state transition** along the progression, and the Domain E capabilities — provenance, conformance, access and integrity, operating condition, and temporal continuity — operate throughout rather than at any single point.

7. Critical Capability Relationships

Relationships are typed. The most consequential are:

Capability	Relationship	Related capability
A-CAP-02	Prerequisite	A-CAP-01
A-CAP-03	Prerequisite	A-CAP-01, A-CAP-02
C-CAP-04	Prerequisite	A-CAP-01, A-CAP-02
B-CAP-02	Prerequisite	B-CAP-01
B-CAP-03	Prerequisite	B-CAP-01, B-CAP-02
B-CAP-04	Prerequisite	B-CAP-01, B-CAP-02, B-CAP-03
C-CAP-01	Prerequisite	B-CAP-03, B-CAP-04
C-CAP-01	Collaborates with	C-CAP-08
C-CAP-01	Constrained by	E-CAP-03
C-CAP-05	Prerequisite	C-CAP-01, C-CAP-04
C-CAP-06	Prerequisite	C-CAP-04
C-CAP-06	Collaborates with	C-CAP-07, C-CAP-08
C-CAP-07	Prerequisite	A-CAP-03
C-CAP-08	Collaborates with	C-CAP-01, C-CAP-03, C-CAP-05, C-CAP-06, C-CAP-07, D-CAP-04
C-CAP-08	Constrained by	E-CAP-03, E-CAP-04
C-CAP-08	Provides evidence to	E-CAP-02
D-CAP-01	Prerequisite	C-CAP-01, C-CAP-04
D-CAP-02	Prerequisite	D-CAP-01
D-CAP-03	Prerequisite	D-CAP-02
D-CAP-04	Collaborates with	C-CAP-08
E-CAP-01	Provides evidence to	all capabilities
E-CAP-02	Evidence consumer	all capabilities

No relationship in this table is a mutual prerequisite. Where two capabilities appear to depend on one another — acceptance and transition, or evolution and dependency impact — the relationship is *collaboration* within one governed operation, not circular prerequisite.

8. Prohibited Capability Substitution

The following substitutions are not permitted: discovery may not substitute for provenance; evaluation may not substitute for justification; justification may not substitute for acceptance; acceptance may not substitute for release; version creation may not substitute for release; permission may not substitute for authority; audit logging may not substitute for accountability; explainability may not substitute for evidential warrant; retention may not substitute for historical authenticity; model confidence may not substitute for epistemic status; and service availability may not be presumed when operating condition is unknown.

Part VIII · Capability Completeness

9. Practice-to-Capability Crosswalk

KF-2002 Practice	Principal capabilities
A1 · OKO Identification	A-CAP-01, A-CAP-03, E-CAP-01
A2 · Knowledge Structuring	A-CAP-02, A-CAP-03
B1 · Evidence Evaluation	B-CAP-01, B-CAP-02
B2 · Knowledge Justification	B-CAP-01, B-CAP-03, B-CAP-04
C1 · Knowledge Acceptance	C-CAP-01, C-CAP-08
C2 · Governance & Stewardship	C-CAP-02, C-CAP-03, C-CAP-04, C-CAP-05
C3 · Knowledge Evolution	C-CAP-06, C-CAP-07, C-CAP-08, C-CAP-04
D1 · Operationalisation	D-CAP-01
D2 · Knowledge Application	D-CAP-02, D-CAP-03
D3 · Knowledge Retirement	D-CAP-04, C-CAP-06, C-CAP-08

Domain E supports every practice.

10. Invariant-to-Capability Crosswalk (family level)

Invariant family	Principal capabilities
Identity and representation (INV-01–03)	A-CAP-01, A-CAP-02, A-CAP-03, C-CAP-04, E-CAP-01
Evidence and justification (INV-04–07)	B-CAP-01, B-CAP-02, B-CAP-03, B-CAP-04
Governance and accountability (INV-08–11)	C-CAP-01, C-CAP-02, C-CAP-03, C-CAP-08, E-CAP-03
Change and history (INV-12–16)	C-CAP-04, C-CAP-06, C-CAP-07, D-CAP-04, E-CAP-05
Operational use (INV-17–20)	C-CAP-05, D-CAP-01, D-CAP-02, D-CAP-03
Assurance and explicit condition (INV-21–24)	E-CAP-02, E-CAP-04

11. Capability Coverage Test

The model is complete when: every KF-2002 practice maps to at least one principal capability; every mandatory KF-2002 artefact has a capability responsible for its creation or maintenance; every KF-3001 invariant is protected by at least one capability (demonstrated individually in Appendix A); every KF-3001 interaction constraint is enforceable through the capabilities in Appendix B; every capability has an upstream derivation; every capability has at least one independently observable conformance condition; no capability exists solely because a familiar technology category suggests it; and no upstream obligation remains unsupported.

Part IX · Capability Conformance

12. Capability Conformance

A capability is conformant when an implementation demonstrates that all required behaviours are realised; all principal information objects can be produced, maintained and retrieved; relationships are satisfied; protected invariants remain true whenever their scope applies; failure and indeterminate conditions are visible; the stated conformance condition can be demonstrated; and no prohibited substitution is used to claim satisfaction.

Capability conformance may be assessed individually for diagnostic purposes, but **architectural conformance is systemic**. A capability cannot be declared conformant where an unsatisfied prerequisite or collaborating capability prevents its required behaviours or protected invariants from being realised. Conformance is not a component-by-component certification.

13. Capability Evidence

Evidence may include representative end-to-end lifecycle cases; information-object inspection; invariant tests; prohibited-path tests; authority and permission checks; historical reconstruction; dependency-impact tests; degraded-state tests; cross-service reconciliation; and independent assurance results. Technology features or vendor claims are not, by themselves, evidence of capability conformance.

14. Capability Conformance and Capability Maturity

Capability conformance is not the same as capability maturity, and the two are assessed separately. A conforming capability may exhibit different degrees of automation, scale, latency, usability, coverage, reliability, modality support and assurance independence. A manually maintained stewardship register may *conform* to C-CAP-02 while being operationally immature; a sophisticated AI governance platform may be *non-conforming* if it silently passes indeterminate evidence or collapses acceptance into model confidence. Maturity does not compensate for an absent required behaviour or a violated invariant.

Part X · Appendices

Appendix A · Invariant-to-Capability Crosswalk (individual)

Invariant	Principal capability	Supporting capabilities
INV-01 · Unique Object Identity	A-CAP-01	A-CAP-03, E-CAP-03
INV-02 · Representation Traceability	A-CAP-02	A-CAP-03, E-CAP-01
INV-03 · Logical Distinction	A-CAP-02	B-CAP-04, C-CAP-04, C-CAP-05

Invariant	Principal capability	Supporting capabilities
INV-04 · Provenance Persistence	E-CAP-01	A-CAP-01, B-CAP-01, E-CAP-05
INV-05 · Justification before Reliance	B-CAP-03	C-CAP-01, C-CAP-05, C-CAP-08
INV-06 · Counter-Evidence Visibility	B-CAP-01	B-CAP-03, B-CAP-04, D-CAP-03
INV-07 · Acceptance Does Not Rewrite Status	B-CAP-04	C-CAP-01
INV-08 · Attributable Transition	C-CAP-08	E-CAP-01, E-CAP-03
INV-09 · Human Accountability	C-CAP-01	C-CAP-02, C-CAP-08, D-CAP-04, E-CAP-03
INV-10 · Standing Governance	C-CAP-03	C-CAP-02, E-CAP-04
INV-11 · Authorised Change	E-CAP-03	C-CAP-08
INV-12 · Version Preservation	C-CAP-04	E-CAP-05
INV-13 · Lineage Preservation	C-CAP-06	C-CAP-04, D-CAP-04, E-CAP-05
INV-14 · No Silent Mutation	C-CAP-08	C-CAP-04, C-CAP-06, E-CAP-01
INV-15 · Dependency Disposition	C-CAP-07	C-CAP-06, C-CAP-08
INV-16 · Retirement Preserves History	D-CAP-04	C-CAP-06, E-CAP-05
INV-17 · Warranted Actionable Form	D-CAP-01	C-CAP-05, C-CAP-01
INV-18 · Governed Application	D-CAP-02	D-CAP-01, C-CAP-03, C-CAP-08
INV-19 · Outcome Traceability	D-CAP-03	E-CAP-01
INV-20 · Modality-Appropriate Explainability	D-CAP-02	A-CAP-02
INV-21 · Inspectable Conformance	E-CAP-02	E-CAP-04
INV-22 · Assurance Independence	E-CAP-02	—
INV-23 · Audit Non-Interference	E-CAP-02	E-CAP-03
INV-24 · No Silent Success	E-CAP-04	B-CAP-02, C-CAP-03, C-CAP-08, E-CAP-02

Every one of the twenty-four invariants has a principal capability, and no invariant is left without coverage — the check a family-level mapping alone can conceal.

Appendix B · Interaction-Constraint-to-Capability Crosswalk

Constraint	Enforcing / participating capabilities
IC-01 · Lifecycle State Authority	C-CAP-08
IC-02 · Gate Enforcement	C-CAP-08, E-CAP-04
IC-03 · Suspension Ownership	C-CAP-08; triggers from C-CAP-03, C-CAP-07, E-CAP-02
IC-04 · Reactivation through Evolution	D-CAP-04, C-CAP-06, C-CAP-08
IC-05 · Identity Immutability	A-CAP-01
IC-06 · Version Immutability	C-CAP-04
IC-07 · Branch and Consolidation Identity	C-CAP-06, A-CAP-01
IC-08 · Release Separation	C-CAP-05
IC-09 · Justification–Acceptance Separation	B-CAP-03, C-CAP-01, B-CAP-04
IC-10 · Operationalisation Eligibility	D-CAP-01
IC-11 · Application Eligibility	D-CAP-02
IC-12 · Warrant Link Preservation	D-CAP-01, D-CAP-02, E-CAP-01
IC-13 · Authority Is Not Permission	E-CAP-03, C-CAP-01
IC-14 · Least Necessary Alteration	E-CAP-03
IC-15 · Material Administrative Action	E-CAP-03
IC-16 · Automated Authority Boundary	C-CAP-01, E-CAP-03
IC-17 · Read-Only Assurance Boundary	E-CAP-02
IC-18 · No Sole Self-Certification	E-CAP-02
IC-19 · Exception Visibility	E-CAP-04
IC-20 · Provenance Continuity	E-CAP-01
IC-21 · Append-Only Correction	C-CAP-04, E-CAP-05
IC-22 · Time Semantics Preservation	E-CAP-05

Every interaction constraint is enforceable through at least one capability, demonstrating enforceability rather than asserting it.

Part XI · Relationship to Subsequent Documents

KF-3003 defines the logical services that expose and coordinate these capabilities; a service may realise several capabilities and a capability may be distributed across several services, and it will carry the typed relationships of §4 and §7 into service collaboration, constraint and evidence contracts. **KF-3004** defines the information objects, record classes and relationships named here, and fixes the recorded, effective and event time semantics of Domain E. **KF-3005** defines the lifecycle controls and gate logic through which the capabilities operate safely,

including the transactional binding of gate decision and transition in C-CAP-08. **KF-3008** assigns the authority and accountability of the participants exercising these capabilities (technical permission is assigned by SVC-16). **KF-3007** defines the tests and evidence by which capability conformance is demonstrated, including the operating-criticality consequences of unavailability. **KF-3009** preserves traceability from Canonical Principle through practice, invariant, capability, service, record, control and implementation.

Closing Statement

The Capability Model defines the complete set of abilities that a governed knowledge architecture must possess. It does not prescribe how those abilities are implemented. It ensures that, whatever form an implementation takes, it can identify and structure knowledge objects, establish and preserve their warrant, govern authority, versioning, release and change, control operational use, learn from outcomes, and demonstrate that these obligations remain satisfied over time.

KF-3003 · Service Architecture

The Logical Services Through Which Capabilities Are Exposed and Governed

Document Class: Reference Architecture · Parent Document: KF-3000 · Governed By: KF-3001 and KF-3002 · Status: Draft · Version: 0.2

```
Reference Architecture (KF-3000)
        ↓
Architectural Rules (KF-3001)
        ↓
Capability Model (KF-3002)
        ↓
▸ Service Architecture (KF-3003)
        ↓
Information & Record Model (KF-3004)
        ↓
Lifecycle & Control (KF-3005)
        ↓
Implementation (KF-4000)
```

Purpose

KF-3002 defines *what* a governed knowledge architecture must be able to do. KF-3003 defines *through which logical services those abilities are exposed, coordinated and governed* — the execution architecture of the Knowledge Foundry.

KF-3003 introduces no new obligations. It does not weaken an invariant, relax an interaction constraint, or select a technology. Its single question is:

> *Through which logical services are the capabilities exposed, coordinated and governed, without violating any interaction constraint?*

A logical service is a stable responsibility exposed to authorised actors and other services through defined operations, information objects and controls. A service may later be realised by software, assigned human work, or a governed combination of both; KF-3003 fixes the responsibility, not the realisation.

Relationship to Upstream Documents

KF-3003 is bounded on three sides and may not exceed any of them.

- From **KF-3002**, it inherits the twenty-four capabilities and their typed relationships. Every service realises one or more capabilities; no service exists without a capability basis.
- From **KF-3001**, it inherits the twenty-two interaction constraints. Every constraint must be enforced at a defined service boundary; a service architecture that cannot enforce a constraint is non-conforming.
- From **KF-3000**, it inherits the intent of the service catalogue. KF-3000 sketched a provisional service list and explicitly deferred the definitive catalogue to KF-3003. This

document establishes that definitive catalogue — nineteen logical services — and where it refines KF-3000's provisional numbering, KF-3003 is authoritative and KF-3000's §11–12 list and §22–24 crosswalks are read as pointing here.

The direction of authority is strict: a service may choose *how* to expose a capability, but may never add to, weaken, or reinterpret the capability, invariant or constraint it inherits.

The Distinctive Property — Interaction Governance

A service architecture is not merely a decomposition of capabilities into components. Its distinctive and load-bearing property is **interaction governance**: it must guarantee that the interaction constraints IC-01–IC-22 hold *in execution*, not only on paper.

Decomposition alone can satisfy every capability individually while allowing services to bypass one another — a version service that also releases, an application service that also accepts, an assurance service that authorises the transitions it audits. KF-3003 exists to make those bypasses structurally impossible by assigning each constraint to a boundary and each governed record to a single authoritative maintainer. Part VII is therefore the centre of this document, not an appendix to it.

Service Admission Test

A proposed logical service is admitted only where:

1. it realises one or more KF-3002 capabilities;
2. every capability it realises names it as the single accountable service, or names another service and this one as a supporting collaborator;
3. it is the authoritative maintainer of a defined, non-overlapping set of information objects;
4. its operations can be stated without naming a technology or product;
5. it enforces, or participates in enforcing, at least one interaction constraint, or is required by a capability that does;
6. it acquires no authority beyond the capabilities it realises;
7. its conformance can be demonstrated independently.

Design Principles for Service Definition

- **SP-1 · Capability-derived services.** Every service is justified by the capabilities it realises; none is admitted because a technology category suggests it.
- **SP-2 · Single accountable service.** Each capability has exactly one accountable service; other services may support it, but accountability is not shared.
- **SP-3 · Authoritative information ownership.** Each governed information object has exactly one authoritative-maintainer service; other services read it or request changes through a governed interaction.
- **SP-4 · Constraint enforcement at a boundary.** Every KF-3001 interaction constraint is enforced at a named service boundary, not left to convention.
- **SP-5 · No authority accrual.** Execution capacity confers no authority. A service that performs an operation does not thereby gain the right to authorise it.

- **SP-6 · Coherent coarseness.** Capabilities are grouped into one service only where they share information ownership and lifecycle role. Distinct governance obligations remain distinct services even where a single component might implement them. Version, acceptance and release are therefore three services, not one.
- **SP-7 · Logical, not physical.** Services are logical responsibilities. One may be distributed across components and several combined into one, provided their boundaries and authoritative ownership remain inspectable.
- **SP-8 · Assurance independence.** The conformance service is read-only with respect to the records it assesses and is never an operational prerequisite of the operations it audits.

Service Specification

Each service is specified through ten fields: **Service statement**, **Purpose**, **Realised capabilities**, **Authoritative information objects**, **Service operations**, **Collaborations** (typed: prerequisite, collaborating, constraining, evidence), **Enforced interaction constraints**, **Authority boundary** (what the service must not do), **Operating criticality**, and **Conformance condition**.

Part I · Service Taxonomy

1. The Logical Services

KF-3003 defines nineteen logical services, numbered sequentially and grouped by domain. Cross-cutting services are grouped together at the end of the sequence.

Service	Domain	Accountable capabilities
SVC-01 · Object Registry	A	A-CAP-01, A-CAP-03
SVC-02 · Representation	A	A-CAP-02
SVC-03 · Evidence	B	B-CAP-01
SVC-04 · Evaluation	B	B-CAP-02
SVC-05 · Justification	B	B-CAP-03, B-CAP-04
SVC-06 · Authority and Acceptance	C	C-CAP-01
SVC-07 · Stewardship and Review	C	C-CAP-02, C-CAP-03
SVC-08 · Lifecycle and Transition	C	C-CAP-08
SVC-09 · Version and Lineage	C	C-CAP-04, C-CAP-06
SVC-10 · Release	C	C-CAP-05
SVC-11 · Dependency	C	C-CAP-07
SVC-12 ·	D	D-CAP-01

Service	Domain	Accountable capabilities
Operationalisation		
SVC-13 · Application and Outcome	D	D-CAP-02, D-CAP-03
SVC-14 · Retirement and Preservation	D	D-CAP-04
SVC-15 · Provenance and Traceability	E	E-CAP-01
SVC-16 · Access and Integrity	E	E-CAP-03
SVC-17 · Operating Condition and Exception	E	E-CAP-04
SVC-18 · Conformance and Audit	E	E-CAP-02
SVC-19 · Temporal Continuity	E	E-CAP-05

Every one of the twenty-four capabilities names exactly one accountable service (§5). The five Domain E services are cross-cutting: Provenance, Access, Operating Condition and Temporal Continuity constrain or supply evidence to the others, and Conformance reads all of them without altering any.

Version, acceptance and release are three services. Version creation (SVC-09) is not acceptance (SVC-06) and neither is release (SVC-10). This three-way separation is the structural expression of IC-06, IC-08 and IC-09, and is the principal refinement of this document over the KF-3000 sketch.

Part II · Domain A Services

SVC-01 · Object Registry Service

Service statement. Assigns and resolves durable object identity and routes intake to the responsible downstream service.

Purpose. Establish identity as the single anchor of governance, distinct from version, representation and record identity.

Realised capabilities. Accountable for A-CAP-01, A-CAP-03. Supports A-CAP-02.

Authoritative information objects. Object Identity Record; Intake Record; Identity Resolution Decision; Resolution Result.

Service operations. Register candidate; assign immutable identifier; resolve identifier to one object; assess intake relation to existing objects; retrieve current and historical governed context; expose unresolved identity conditions.

Collaborations. *Constrained by:* SVC-16, SVC-17. *Evidence from:* SVC-15. *Triggers:* SVC-09 (version or lineage creation). *Collaborates with:* SVC-02, SVC-05, SVC-08 (context for resolution).

Enforced interaction constraints. IC-05 (Identity Immutability).

Authority boundary. Does not create versions, branches or consolidated successors; does not accept, release or retire; does not alter representation content.

Operating criticality. Governance-critical; fail-closed — no progression without a resolved, unique identity.

Conformance condition. Any registered object resolves to exactly one identity with recoverable origin and intake decision, and no identifier is reused.

SVC-02 · Representation Service

Service statement. Creates and maintains the inspectable representation of an object's content, type and relationships.

Purpose. Make objects inspectable so that evaluation, justification, governance and explanation can operate on explicit structure.

Realised capabilities. Accountable for A-CAP-02. Supports A-CAP-03.

Authoritative information objects. Structured Representation; Semantic Type Record; Metadata Record; Object Relationship Record; Representation Limitation Record.

Service operations. Assign semantic type; represent constituent elements and relationships; attach metadata; link representation to object version; declare modality limitations.

Collaborations. *Prerequisite:* SVC-01. *Constrained by:* SVC-16. *Evidence from:* SVC-15.

Enforced interaction constraints. Participates in INV-02, INV-03 preservation; no primary IC boundary.

Authority boundary. Does not evaluate, justify or accept; does not assign identity; does not conflate content with warrant.

Operating criticality. Governance-critical — inspectability precedes all downstream governance.

Conformance condition. For any representation, a reviewer can determine what it expresses, which object version it represents, how its elements relate and what modality limitations apply.

Part III · Domain B Services

SVC-03 · Evidence Service

Service statement. Registers evidence and counter-evidence as governed records related to claims and objects.

Purpose. Make evidential support, challenge and origin independently inspectable, distinct from the conclusions drawn from them.

Realised capabilities. Accountable for B-CAP-01.

Authoritative information objects. Evidence Record; Counter-Evidence Record; Evidence-to-Claim Link; Evidence Classification; Evidence Availability Status.

Service operations. Register evidence; classify support, challenge or context; relate evidence to claims and objects; record availability; retain conflicting evidence.

Collaborations. *Prerequisite:* SVC-01, SVC-02. *Constrained by:* SVC-16. *Evidence from:* SVC-15.

Enforced interaction constraints. Participates in IC-20 (provenance of evidence).

Authority boundary. Does not evaluate quality, construct justification or determine acceptance; does not delete counter-evidence.

Operating criticality. Governance-critical — warrant depends on inspectable evidence.

Conformance condition. For each material claim, the related evidence and counter-evidence, their origins, roles and availability are recoverable.

SVC-04 · Evaluation Service

Service statement. Assesses quality and fitness against explicit criteria and records the verdict.

Purpose. Determine whether an object is suitable to proceed toward justification, distinct from establishing why it may be believed.

Realised capabilities. Accountable for B-CAP-02.

Authoritative information objects. Evaluation Criteria Record; Quality Assessment; Evaluation Verdict; Evaluation Exception; Confidence Record.

Service operations. Apply criteria; assess quality, consistency, completeness and confidence; record pass, fail, return, pending or indeterminate; retain reasons; route incomplete assessments.

Collaborations. *Prerequisite:* SVC-02, SVC-03. *Constrained by:* SVC-17. *Evidence to:* SVC-18.

Enforced interaction constraints. Participates in INV-24 (no silent pass).

Authority boundary. Does not construct justification or accept; does not define organisational thresholds.

Operating criticality. Governance-critical — the quality gate precedes justification.

Conformance condition. For any evaluation, the criteria, inputs, evaluator, result, confidence, reasons and disposition are recoverable.

SVC-05 · Justification Service

Service statement. Assembles and exposes inspectable warrant, and maintains epistemic status distinct from acceptance.

Purpose. Establish why an object may be believed, and preserve uncertainty and dispute even where reliance is later permitted.

Realised capabilities. Accountable for B-CAP-03, B-CAP-04.

Authoritative information objects. Justification Record; Assumption Record; Rationale Record; Traceability Chain; Challenge Record; Epistemic Status Record; Confidence Record; Contestation Record; Epistemic Status Change Record.

Service operations. Relate claims to evidence and counter-evidence; record assumptions and rationale; construct traceability; represent epistemic status; preserve prior epistemic states; expose warrant for independent inspection.

Collaborations. *Prerequisite:* SVC-03, SVC-04. *Collaborating:* SVC-06 (justification–acceptance boundary). *Constrained by:* SVC-16. *Evidence from:* SVC-15.

Enforced interaction constraints. IC-09 (Justification–Acceptance Separation), enforced jointly with SVC-06.

Authority boundary. Does not accept, release or apply; does not let acceptance overwrite epistemic status; does not claim justification guarantees truth.

Operating criticality. Governance-critical — justification is the warrant on which reliance rests.

Conformance condition. An independent reviewer can reconstruct the reasons for and against belief, the assumptions, and the object's epistemic standing separately from its acceptance state.

Part IV · Domain C Services

SVC-06 · Authority and Acceptance Service

Service statement. Records the governed decision by which an identified authority permits reliance upon a justified object.

Purpose. Separate epistemic warrant from institutional permission to rely.

Realised capabilities. Accountable for C-CAP-01.

Authoritative information objects. Acceptance Record; Refusal Record; Scope-of-Reliance Record; Authority Assignment Reference; Acceptance Condition Record.

Service operations. Identify accepting authority; verify mandate, scope and effective period; record acceptance, provisional acceptance or refusal; define reliance scope and conditions; route refusals.

Collaborations. *Prerequisite:* SVC-05, SVC-16. *Collaborating:* SVC-08. *Triggers:* SVC-10 (release), SVC-08 (transition). *Evidence from:* SVC-15.

Enforced interaction constraints. IC-09 (with SVC-05); IC-13 (Authority Is Not Permission); IC-16 (Automated Authority Boundary).

Authority boundary. Does not construct justification; does not create or release versions itself; does not assign permissions (that is SVC-16); does not become the final accountable authority where a human authority is required.

Operating criticality. Governance-critical; fail-closed — no acceptance without an applicable authority.

Conformance condition. Every accepted object carries a decision attributable to an authority whose mandate, scope and effective period applied, with reliance scope, conditions, date and supporting justification.

SVC-07 · Stewardship and Review Service

Service statement. Maintains ownership and accountability and determines whether standing governance conditions remain satisfied.

Purpose. Preserve sustained human accountability and legitimacy across operational life.

Realised capabilities. Accountable for C-CAP-02, C-CAP-03.

Authoritative information objects. Stewardship Assignment; Responsibility Record; Stewardship History; Accountable Body Record; Review Schedule; Review Record; Compliance Assessment; Compliance Finding; Governance Exception.

Service operations. Assign and transfer stewardship; schedule and record reviews; detect overdue reviews; evaluate compliance; raise triggers for suspension, evolution or retirement.

Collaborations. *Prerequisite:* SVC-01. *Collaborating:* SVC-08. *Constrained by:* SVC-16, SVC-17. *Triggers:* SVC-08, SVC-09. *Evidence from:* SVC-15; *Evidence to:* SVC-18.

Enforced interaction constraints. Participates in INV-10 (standing governance) upkeep; raises IC-03 suspension triggers.

Authority boundary. Does not accept or release; does not execute the transitions it triggers (that is SVC-08); does not assign access permissions.

Operating criticality. Governance-critical; recoverable within a defined period — a late review is bounded and must be visible as overdue.

Conformance condition. Every operational object has a recoverable steward, review status and compliance standing, and no operational object is ownerless.

SVC-08 · Lifecycle and Transition Service

Service statement. Registers lifecycle events, evaluates gates and executes attributable state transitions as one governed operation.

Purpose. Preserve controlled movement between lifecycle states and hold the sole right to change recorded state.

Realised capabilities. Accountable for C-CAP-08.

Authoritative information objects. Event Record; Gate Decision Record; Transition Record; Transition Warrant; Transition Status.

Service operations. Register events; identify applicable gate; evaluate eligibility; bind gate decision and transition transactionally; attribute execution and authority separately; execute forward, return, suspension and retirement transitions.

Collaborations. *Collaborating:* SVC-06, SVC-07, SVC-09, SVC-10, SVC-11, SVC-14. *Constrained by:* SVC-16, SVC-17. *Evidence from:* SVC-15, SVC-19; *Evidence to:* SVC-18.

Enforced interaction constraints. IC-01 (Lifecycle State Authority); IC-02 (Gate Enforcement); IC-03 (Suspension Ownership).

Authority boundary. Does not make the acceptance decision (that is SVC-06); does not define gate criteria (that is KF-3005); does not disposition dependencies (that is SVC-11). Only SVC-08 may alter recorded lifecycle state.

Operating criticality. Governance-critical; fail-closed — an indeterminate or unavailable gate does not silently pass.

Conformance condition. For every transition, the event, prior and proposed state, gate result, executor, accountable authority and outcome are recoverable under one transition identifier.

SVC-09 · Version and Lineage Service

Service statement. Creates immutable versions and preserves lineage across change, without altering enduring identity and without releasing anything.

Purpose. Preserve every state of content and warrant as an immutable, recoverable version, and preserve predecessor–successor lineage — kept strictly distinct from acceptance and release.

Realised capabilities. Accountable for C-CAP-04 (version), C-CAP-06 (evolution and lineage).

Authoritative information objects. Version Record; Version State; Predecessor–Successor Link; Correction Record; Change Record; Lineage Record; Branch Record; Consolidation Record; Reactivation Proposal.

Service operations. Create candidate and governed versions; preserve prior versions; prevent overwrite; identify predecessor and successor versions; branch, consolidate and supersede with explicit lineage; preserve correction history; route changed states for renewed warrant.

Collaborations. *Prerequisite:* SVC-01, SVC-02. *Collaborating:* SVC-08, SVC-11. *Constrained by:* SVC-16. *Evidence from:* SVC-15; *Evidence with:* SVC-19.

Enforced interaction constraints. IC-06 (Version Immutability); IC-07 (Branch and Consolidation Identity); IC-21 (Append-Only Correction, with SVC-19).

Authority boundary. Does not accept, release or apply; does not create or mutate identity; version creation never implies release. A new version is never automatically released.

Operating criticality. Governance-critical; fail-closed — no overwrite of a prior version.

Conformance condition. For any object, its immutable versions, their order, correction and supersession relationships and full lineage are recoverable, distinct from whether any version is accepted or released.

SVC-10 • Release Service

Service statement. Designates an accepted version as eligible for a defined scope and period of operational reliance, and governs withdrawal and supersession of that eligibility.

Purpose. Hold operational release as a distinct governed decision — separate from version creation and from acceptance — so that eligibility can be granted, retained, revalidated and withdrawn without touching version or identity.

Realised capabilities. Accountable for C-CAP-05.

Authoritative information objects. Release Record; Release Scope; Effective Period; Release Status; Withdrawal Record; Supersession Record; Revalidation Record.

Service operations. Verify an existing acceptance; bind release to a specific immutable version; define release scope and effective period; set and change release status; run eligibility checks; withdraw, supersede or revalidate a release; preserve release history.

Collaborations. *Prerequisite:* SVC-06 (acceptance), SVC-09 (version). *Collaborating:* SVC-08. *Constrained by:* SVC-16, SVC-17. *Evidence from:* SVC-15, SVC-19.

Enforced interaction constraints. IC-08 (Release Separation).

Authority boundary. Does not create versions (that is SVC-09); does not make the acceptance decision (that is SVC-06); does not derive or apply actionable forms. Release requires an existing acceptance and never manufactures one.

Operating criticality. Governance-critical; fail-closed — no release without a bound, accepted version, and no automatic inheritance of release by a new version.

Conformance condition. For every release, the immutable version released, its acceptance basis, scope, effective period, current status and predecessor or successor releases are recoverable, and no version is released without acceptance.

SVC-11 · Dependency Service

Service statement. Represents dependencies and dispositions objects and forms affected by change.

Purpose. Prevent locally correct change from silently invalidating dependent knowledge.

Realised capabilities. Accountable for C-CAP-07.

Authoritative information objects. Dependency Record; Dependency Impact Record; Revalidation Requirement; Dependency Disposition Record; Suspension Recommendation.

Service operations. Represent dependencies; determine impacted dependants; classify impact; disposition as revalidation, evolution, suspension or no-action; preserve impact decisions.

Collaborations. *Prerequisite:* SVC-01 (resolution). *Collaborating:* SVC-09. *Triggers:* SVC-08 (suspension), SVC-09 (evolve dependants). *Evidence to:* SVC-18.

Enforced interaction constraints. Participates in INV-15 (dependency disposition); raises IC-03 suspension triggers.

Authority boundary. Does not execute change or transition; does not accept or release; does not itself apply forms.

Operating criticality. Governance-critical; fail-closed — a changed object does not return to use with undispositioned dependants.

Conformance condition. For every material change, affected dependants and their recorded disposition are recoverable before the changed object returns to operational use.

Part V · Domain D Services

SVC-12 · Operationalisation Service

Service statement. Derives actionable forms from governed objects and controls their eligibility for use.

Purpose. Convert represented knowledge into usable forms without severing warrant. Form-level release governed here is distinct from object-level release (SVC-10).

Realised capabilities. Accountable for D-CAP-01.

Authoritative information objects. Actionable Form; Derivation Record; Warrant Linkage; Form Release Record; Form Status.

Service operations. Derive forms; distinguish provisional from released; link forms to source version, justification and acceptance; withdraw or replace forms on source change.

Collaborations. *Prerequisite:* SVC-06, SVC-09. *Collaborating:* SVC-10 (release alignment). *Constrained by:* SVC-16. *Evidence from:* SVC-15.

Enforced interaction constraints. IC-10 (Operationalisation Eligibility); IC-12 (Warrant Link Preservation).

Authority boundary. Does not accept or release the underlying object; does not apply forms; does not permit provisional forms into use.

Operating criticality. Operationally critical; fail-closed for release — a form is not made usable without a warrant linkage.

Conformance condition. Every operationally eligible form traces to an accepted object version and carries an inspectable warrant linkage.

SVC-13 · Application and Outcome Service

Service statement. Governs material use of actionable forms and captures outcomes as governed evidence.

Purpose. Keep use authorised, traceable and explainable, and close the loop back to the lifecycle.

Realised capabilities. Accountable for D-CAP-02, D-CAP-03.

Authoritative information objects. Application Record; Use Context Record; Eligibility Decision; Explanation Record; Application Participant Record; Outcome Record; Interpretation Record; Feedback Record; Materiality Assessment; Disposition Record; Lifecycle Trigger.

Service operations. Verify eligibility before use; record context and participants; preserve input-to-result path; produce modality-appropriate explanation; capture outcomes; assess materiality; assign disposition; raise lifecycle triggers.

Collaborations. *Prerequisite:* SVC-12. *Collaborating:* SVC-08. *Constrained by:* SVC-07 (governance status), SVC-16. *Triggers:* SVC-07, SVC-09, SVC-08. *Evidence from:* SVC-15.

Enforced interaction constraints. IC-11 (Application Eligibility); IC-12 (Warrant Link Preservation).

Authority boundary. Does not derive forms; does not accept or release; does not execute evolution or retirement, only trigger them.

Operating criticality. Operationally critical; fail-closed for use — an ineligible object is not applied. Outcome capture is recoverable within a defined, visible period.

Conformance condition. For each material application, what was used, why it was eligible, who participated, the result and a proportionate explanation are recoverable, and every material outcome has a recorded disposition.

SVC-14 · Retirement and Preservation Service

Service statement. Concludes operational use of an object while preserving identity, records and lineage.

Purpose. Remove knowledge from active reliance without destroying institutional memory.

Realised capabilities. Accountable for D-CAP-04.

Authoritative information objects. Retirement Record; Historical Object Record; Preservation Record; Reactivation Request; Successor Link.

Service operations. Record retirement reason and authority; remove object and forms from active eligibility; preserve the historical record; maintain successor links; support historical retrieval; route reactivation through evolution.

Collaborations. *Prerequisite:* SVC-01. *Collaborating:* SVC-08. *Constrained by:* SVC-16. *Evidence from:* SVC-19. *Triggers:* SVC-09 (reactivation via evolution).

Enforced interaction constraints. IC-04 (Reactivation through Evolution).

Authority boundary. Does not delete governed history; does not accept; does not permit reactivation outside governed evolution.

Operating criticality. Governance-critical — destructive deletion of governed history is prohibited.

Conformance condition. A retired object cannot be used operationally, yet its identity, versions, warrant, governance history and lineage remain recoverable.

Part VI · Domain E Services (Cross-Cutting)

The five Domain E services preserve and evaluate the governed context surrounding substantive knowledge. None authors or alters substantive content, and none independently authorises a lifecycle transition. They are ordered here from the most foundational (provenance) to the most derived (temporal reconstruction).

SVC-15 · Provenance and Traceability Service

Service statement. Preserves origin, derivation, custody, attribution and end-to-end traceability across all objects, records, transitions and applications.

Purpose. Make current states and historical change reconstructable.

Realised capabilities. Accountable for E-CAP-01.

Authoritative information objects. Provenance Record; Custody Record; Derivation Link; Attribution Link; Traceability Path; Provenance Gap Record.

Service operations. Preserve origin and derivation; trace representations, changes, forms, applications and outcomes; carry provenance across modality hand-offs; identify provenance gaps explicitly.

Collaborations. *Prerequisite:* SVC-01. *Constrained by:* SVC-16. *Evidence to:* all services.

Enforced interaction constraints. IC-20 (Provenance Continuity).

Authority boundary. Does not author or alter substantive content; does not authorise transitions; does not evaluate conformance.

Operating criticality. Governance- and conformance-critical; fail-closed at hand-offs — a hand-off that cannot preserve provenance is not silently completed.

Conformance condition. For a representative outcome, the path from source and creation through use and outcome is reconstructable, with any provenance gaps exposed.

SVC-16 · Access and Integrity Service

Service statement. Restricts access and material action by assigned permission while preserving the permission ≠ authority distinction, and protects record integrity.

Purpose. Protect governed records and operations against unauthorised access, alteration or disclosure.

Realised capabilities. Accountable for E-CAP-03.

Authoritative information objects. Access Policy; Permission Assignment; Authority Assignment Reference; Disclosure Constraint; Integrity Status; Administrative Action Record.

Service operations. Identify actors; assign and revoke permissions; enforce action permissions; protect integrity; enforce disclosure and use restrictions; record administrative action.

Collaborations. *Collaborating:* SVC-07. *Constrains:* all services performing a material action. *Evidence from:* SVC-19.

Enforced interaction constraints. IC-13 (Authority Is Not Permission); IC-14 (Least Necessary Alteration); IC-15 (Material Administrative Action); IC-16 (Automated Authority Boundary).

Authority boundary. Does not make governance decisions (acceptance, stewardship, retirement); grants permission, never authority; does not select security technologies.

Operating criticality. Governance-critical; fail-closed — an unauthorised material action is refused by default.

Conformance condition. For every material action, the actor's required permission and, where applicable, the separate valid authority decision are demonstrable.

SVC-17 · Operating Condition and Exception Service

Service statement. Determines and exposes whether required services, controls and evidence are operational, degraded, failed, pending or indeterminate, and governs the consequences.

Purpose. Prevent absent evidence, failed controls or unavailable services from appearing as success. This responsibility is materially different from provenance or protection: it is the architecture's fail-safe determinant, and it directly protects INV-24 (No Silent Success).

Realised capabilities. Accountable for E-CAP-04.

Authoritative information objects. Operating Condition Record; Degradation Record; Exception Record; Compensating Control Record; Recovery Record; Failure Disposition Record.

Service operations. Distinguish pass, fail, pending and indeterminate; represent degradation; determine fail-open or fail-closed posture; raise and bound exceptions; preserve compensating controls; trigger suspension or correction.

Collaborations. *Collaborating:* SVC-08. *Constrains:* SVC-08, SVC-07, SVC-04, SVC-10 (fail-posture on gates, reviews, evaluations and release). *Triggers:* SVC-08. *Evidence to:* SVC-18.

Enforced interaction constraints. IC-19 (Exception Visibility); participates in IC-02 (fail-posture at gates).

Authority boundary. Does not execute the corrective transition (that is SVC-08); does not define organisational risk acceptance.

Operating criticality. Conformance- and governance-critical; its own degradation defaults to fail-closed.

Conformance condition. When a condition cannot be evaluated or a service is degraded, the condition is recorded explicitly and the predefined consequence applied, rather than treated as satisfied.

SVC-18 · Conformance and Audit Service

Service statement. Determines and demonstrates whether required capabilities, invariants, gates, controls and constraints are satisfied.

Purpose. Prevent compliance from resting on structural resemblance or self-assertion.

Realised capabilities. Accountable for E-CAP-02.

Authoritative information objects. Conformance Assessment; Invariant Test Result; Constraint Test Result; Assurance Finding; Non-Conformance Record; Exception Reference.

Service operations. Inspect records and control evidence; test invariants and prohibited paths; reconcile evidence across services; record findings; distinguish conformance, variation, exception and non-conformance; recommend suspension or correction.

Collaborations. *Evidence from:* all services. *Triggers:* SVC-08 (via suspension recommendation), SVC-17.

Enforced interaction constraints. IC-17 (Read-Only Assurance Boundary); IC-18 (No Sole Self-Certification).

Authority boundary. Does not alter the records it examines; does not authorise transitions; is never an operational prerequisite of an operation it audits; does not certify its own outputs without independent evidence.

Operating criticality. Conformance-critical but not transaction-blocking — its unavailability renders operations unproven, not halted.

Conformance condition. The architecture can produce independently generated evidence for every required capability and report unmet, indeterminate or violated conditions without modifying the evidence assessed.

SVC-19 · Temporal Continuity Service

Service statement. Owns the governed temporal contract of the architecture: the meaning, sequence and authenticity of governed records and states across time, including as-of reconstruction.

Purpose. Maintain trust where events, records and effective conditions occur at different times. This is a distinct authoritative responsibility, not a timestamp utility: it owns effective dating, event sequencing, as-of reconstruction, retention and correction history, and it alone can answer "what did the governed record say, and hold true, as of a stated time."

Realised capabilities. Accountable for E-CAP-05.

Authoritative information objects. Temporal Metadata Record; Effective Period; Event-Time Record; Custody History; Retention Condition; Correction History; Historical Reconstruction Result.

Service operations. Preserve recorded, effective and event time as three distinct semantics; sequence transitions and decisions; maintain correction history; preserve authenticity across format and custody change; reconstruct governed state as of a specified time; enforce retention and disposition.

Collaborations. *Collaborating:* SVC-09 (append-only correction). *Evidence from:* SVC-15; *Evidence to:* SVC-18.

Enforced interaction constraints. IC-21 (Append-Only Correction, with SVC-09); IC-22 (Time Semantics Preservation).

Authority boundary. Does not perform version mechanics (that is SVC-09); does not select storage technology; does not alter substantive content. It owns *time semantics and reconstruction*, not the records' substantive content.

Operating criticality. Governance-critical; recoverable with bounded manual control, but never by relying on overwritten records.

Conformance condition. The object's content, warrant, acceptance, governance and operational state can be reconstructed as of a specified historical point — using recorded, effective and event time — without relying on overwritten current records.

Part VII · Interaction Governance

This part is the centre of KF-3003. It fixes how services interact so that the interaction constraints hold in execution.

2. Service Interaction Principles

1. Only the accountable service may alter an information object it authoritatively maintains; others request change through a governed interaction.
2. Only SVC-08 may alter recorded lifecycle state.
3. No transition executes without a gate decision bound to it in the same operation.
4. No object is released (SVC-10) without an existing acceptance recorded by SVC-06 and a version created by SVC-09.
5. No actionable form is derived or applied without a preserved warrant linkage.
6. No governed record is overwritten; corrections append or version.
7. No service acquires authority by holding permission.
8. No automated service is the final accountable authority where human authority is required.
9. SVC-18 reads all services and alters none, and is never a prerequisite of the operation it audits.
10. Every material inter-service action is attributable and carries provenance.
11. When a required condition is unavailable or indeterminate, SVC-17 determines the fail-posture; the default is fail-closed for governance-critical operations.

3. Constraint Enforcement Matrix

Every interaction constraint is enforced at a named service boundary.

Constraint	Enforcing boundary	Enforcement mechanism
IC-01 · Lifecycle State Authority	SVC-08	sole authority to write lifecycle state
IC-02 · Gate Enforcement	SVC-08 (with SVC-17)	gate decision bound to transition; fail-posture on indeterminacy
IC-03 · Suspension Ownership	SVC-08	owns suspension; SVC-07/11/18 supply reason
IC-04 · Reactivation through Evolution	SVC-14 → SVC-09 → SVC-08	reactivation routed through evolution and re-acceptance
IC-05 · Identity Immutability	SVC-01	write-once identifier
IC-06 · Version Immutability	SVC-09	immutable version records
IC-07 · Branch and Consolidation Identity	SVC-09 (with SVC-01)	successor identity with lineage links
IC-08 · Release Separation	SVC-10 (with SVC-06, SVC-09)	release record distinct from version, bound to acceptance
IC-09 · Justification–Acceptance Separation	SVC-05 and SVC-06	separate authoritative maintainers; boundary tested by SVC-18
IC-10 · Operationalisation Eligibility	SVC-12	warrant linkage required before release

Constraint	Enforcing boundary	Enforcement mechanism
IC-11 · Application Eligibility	SVC-13	eligibility check recorded before use
IC-12 · Warrant Link Preservation	SVC-12, SVC-13	non-nullable warrant links
IC-13 · Authority Is Not Permission	SVC-16 (with SVC-06)	permission and authority held as separate records
IC-14 · Least Necessary Alteration	SVC-16	authoritative-maintainer boundary enforced
IC-15 · Material Administrative Action	SVC-16	administrative action log
IC-16 · Automated Authority Boundary	SVC-06, SVC-16	final authority resolves to a human role
IC-17 · Read-Only Assurance Boundary	SVC-18	read-only access to examined records
IC-18 · No Sole Self-Certification	SVC-18	independent evidence required per judgement
IC-19 · Exception Visibility	SVC-17	exception persists until closed
IC-20 · Provenance Continuity	SVC-15	provenance carried, not regenerated, at hand-off
IC-21 · Append-Only Correction	SVC-09, SVC-19	corrections append
IC-22 · Time Semantics Preservation	SVC-19	recorded, effective and event time preserved

4. Prohibited Cross-Service Actions

No service may accept on behalf of another; write another service's authoritative records directly; change lifecycle state outside SVC-08; release a version without an SVC-06 acceptance and an SVC-09 version; treat a permission from SVC-16 as an authority; present an SVC-18 assessment as an authorisation; or complete a hand-off that SVC-15 cannot make traceable.

Part VIII · Completeness

5. Capability-to-Service Crosswalk

Each capability names exactly one accountable service.

Capability	Accountable service	Supporting services
A-CAP-01	SVC-01	SVC-15, SVC-16
A-CAP-02	SVC-02	SVC-01, SVC-15

Capability	Accountable service	Supporting services
A-CAP-03	SVC-01	SVC-02, SVC-05, SVC-08
B-CAP-01	SVC-03	SVC-15
B-CAP-02	SVC-04	SVC-17
B-CAP-03	SVC-05	SVC-03, SVC-04
B-CAP-04	SVC-05	SVC-06
C-CAP-01	SVC-06	SVC-05, SVC-16, SVC-08
C-CAP-02	SVC-07	SVC-16
C-CAP-03	SVC-07	SVC-08, SVC-17
C-CAP-04	SVC-09	SVC-01, SVC-19
C-CAP-05	SVC-10	SVC-06, SVC-09
C-CAP-06	SVC-09	SVC-11, SVC-08
C-CAP-07	SVC-11	SVC-01, SVC-09
C-CAP-08	SVC-08	SVC-06, SVC-16, SVC-17
D-CAP-01	SVC-12	SVC-06, SVC-09, SVC-10
D-CAP-02	SVC-13	SVC-07, SVC-16
D-CAP-03	SVC-13	SVC-15
D-CAP-04	SVC-14	SVC-08, SVC-19
E-CAP-01	SVC-15	SVC-01
E-CAP-02	SVC-18	all
E-CAP-03	SVC-16	SVC-07
E-CAP-04	SVC-17	SVC-08
E-CAP-05	SVC-19	SVC-15, SVC-09

Every capability has exactly one accountable service; no capability is unassigned and no service is without a capability basis.

6. Coverage Test

The service architecture is complete when: every capability names exactly one accountable service; every governed information object has exactly one authoritative-maintainer service; every interaction constraint is enforced at a named boundary (§3); no service holds authority beyond its capabilities; no prohibited cross-service action is structurally possible; and the conformance service is read-only and non-prerequisite.

Part IX · Service Conformance

7. Service Conformance

A service conforms when it realises its accountable capabilities; is the sole authoritative maintainer of its information objects; exposes its operations without technology dependence; enforces its assigned interaction constraints; performs no prohibited action; and can demonstrate its conformance condition. As in KF-3002, conformance is systemic: a service is not conformant where an unmet collaboration prevents a constraint it must enforce.

8. Service Evidence

Evidence includes authoritative-ownership maps; operation traces; constraint-enforcement tests; prohibited-path tests; attribution and provenance checks; and independent assurance from SVC-18. Physical consolidation of services is permitted only where the authoritative-ownership boundary and constraint enforcement remain inspectable.

Part X · Relationship to Subsequent Documents

KF-3004 defines the information objects and record classes named here and fixes their authoritative-maintainer assignments and the recorded/effective/event time semantics owned by SVC-19. **KF-3005** defines the gate logic and lifecycle controls that SVC-08 and SVC-17 apply, including the transactional gate–transition binding and the release eligibility checks of SVC-10. **KF-3008** defines the human authority and accountability model that SVC-06 and SVC-07 depend upon, kept distinct from the technical permission SVC-16 assigns. **KF-3007** defines the tests and evidence for service and constraint conformance. **KF-3009** preserves traceability from principle through capability, service, record and control.

Closing Statement

The Service Architecture converts the twenty-four capabilities into nineteen governed logical services and, more importantly, into a set of enforced boundaries. Its contribution is not that the capabilities are exposed, but that they cannot be exercised in a way that bypasses one another: version is not release, release is not acceptance, permission is not authority, and assurance never authorises what it audits. Every interaction constraint is enforced at a named boundary; every governed record has a single authoritative maintainer; and no service holds authority it was not given. That is what makes this an execution architecture for governed knowledge rather than a catalogue of components.

KF-3004 · Governed Information and Record Model

The Logical Information Architecture of the Knowledge Foundry

Document Class: Reference Architecture · Parent Document: KF-3000 · Governed By: KF-3001 · Derived From: KF-3002 and KF-3003 · Status: Draft · Version: 0.2

```
Reference Architecture (KF-3000)
        ↓
Architectural Rules (KF-3001)
        ↓
Capability and Service Models (KF-3002 / KF-3003)
        ↓
▸ Governed Information and Record Model (KF-3004)
        ↓
Control, Authority and Conformance Models (KF-3005–KF-3009)
        ↓
Implementation Schemas (KF-4000)
```

Purpose

KF-3004 defines the logical information objects, record classes, relationships, identity rules and temporal semantics required by the Knowledge Foundry architecture. It establishes what governed information must exist; how it is identified; how information objects relate; which single logical service maintains each authoritative record; how substantive, epistemic, governance, operational and assurance information remain distinguishable; how current and historical states are reconstructed; and how correction, supersession and retention operate without silent mutation.

KF-3004 does not prescribe database technology, table structure, file format, graph implementation, indexing strategy, API payload, physical partitioning, vendor platform or user-interface representation.

It answers:

> *What logical information must a conforming Knowledge Foundry implementation preserve?*

Every record family in this document resolves to **exactly one authoritative maintaining service** in the frozen KF-3003 nineteen-service catalogue. There is no dual authoritative ownership: where two services touch a concept, one holds the authoritative record and the other holds an adjacent, related or derived view (§7.1).

Information Derivation

```
Canonical Obligation → Lifecycle/Practice Requirement → Invariant and
Capability
→ Logical Service Responsibility → Required Information Object
→ Relationship and Temporal Semantics → Implementation Schema
```

Every principal information object must be traceable to a lifecycle or practice obligation; a KF-3001 invariant or constraint; a KF-3002 capability; a KF-3003 service responsibility; and a conformance or historical-reconstruction need.

Part I · Information Architecture Foundations

1. Core Information Concepts

1.1 Operational Knowledge Object. The enduring governed subject whose identity persists across versions, representations, assessments, releases and applications. The object is not identical to a document, file, row, model instance, prompt, version, actionable form or application result; it may be *represented* through one or more of these.

1.2 Object Version. An immutable represented state of an object. It belongs to exactly one enduring identity, captures a bounded substantive state, cannot be silently overwritten, may have several representations, may be evaluated/justified/accepted/released, and does not automatically inherit acceptance or release from another version.

1.3 Representation. An inspectable expression of the substantive content of a version — textual, symbolic, graphical, structured, computational, parametric, multimodal or hybrid. Representation identity is distinct from object and version identity.

1.4 Governed Record. An attributable, temporally situated information object preserving evidence of an event, decision, assessment, relationship, condition, assignment, action, outcome or conformance finding. It does not necessarily contain substantive knowledge.

1.5 Reference. A governed relationship to an information object maintained authoritatively elsewhere, preserving enough to resolve its target, identify the target version or effective state, detect broken resolution and preserve historical meaning.

1.6 Information Projection. A derived, cached or indexed view of authoritative information. A projection is not authoritative; it identifies its source records, preserves refresh/effective-state information, cannot silently override the authoritative record, and may be regenerated where required.

2. Information Classes

Five classes, physically co-locatable but logically distinguishable: **Substantive** (what the object expresses); **Epistemic** (why, how strongly and under what uncertainty it may be believed); **Governance** (who may rely on it and under what conditions); **Operational** (how knowledge was transformed, used and observed); **Assurance and continuity** (how integrity, provenance, temporal continuity and conformance are demonstrated).

3. Information Object Admission Test

Admitted where it is required to discharge an upstream obligation; has a distinct architectural meaning; has an identifiable single authoritative maintainer; has stateable identity or relationship semantics; has definable temporal behaviour; its absence would prevent control,

traceability or conformance; it is not merely an undocumented attribute of another object; and it does not duplicate an existing authoritative record.

4. Information Object Specification

Each principal information object may be specified through: Definition; Architectural purpose; Information class; Authoritative maintaining service; Identity rule; Required properties; Principal relationships; Lifecycle or status; Temporal semantics; Integrity and mutation rule; Retention and disposition; Protected invariants; Conformance test.

4.1 Information Object Granularity and Record Categories

KF-3004 identifies logical information obligations. A named information object may be implemented as an independently identified governed record, an immutable sub-record, an effective-dated status assertion, a governed relationship, a controlled property, or a non-authoritative projection. A named item requires **independent record identity** only where it has its own authority, effective period, correction history, provenance, retention obligation, lifecycle or conformance significance. KF-4000 determines physical normalisation while preserving these semantics — a record code is not a mandatory database table.

Each named object carries one of six **record categories**, which sets the specification depth required:

Category	Required specification
Core Record	all thirteen fields of §4
Supporting Record	definition; maintainer; identity rule; required properties; parent/related core record; temporal and mutation rule
Projection	source records; freshness rule; invalidation rule; non-authoritative status
Relationship Record	the §9.1 relationship-type specification
Status Assertion	controlled value set; the core record it qualifies; effective-dating rule
Controlled Vocabulary	permitted values and their governing service

Core records in this document are OBJ-01, VER-01, ACC-01, TRN-01, REL-01, ACT-01, APP-01, RET-01, PRO-01, CNF-05. All others are supporting records, projections, status assertions or relationship records as marked.

5. Common Record Properties

Every governed record must contain or resolve to: stable record identity; record class; related object identity; related version identity where applicable; creator or originating service; initiating actor; accountable authority where applicable; recorded time; event time where distinct; effective time or period where applicable; status; provenance; integrity status; access classification; supersession or correction relationship; retention condition; authoritative

maintaining service. A governed reference suffices where resolution and historical meaning are preserved.

5.1 Material Information Activity

"Material" is defined, so implementations neither record everything nor lose accountability. An activity is **material** where it can affect: identity; substantive content; epistemic standing; authority; acceptance; lifecycle state; release eligibility; dependency disposition; operational application; outcome interpretation; access or integrity; historical reconstruction; or conformance. The authoritative definition of materiality, and the assurance-class scale that grades the *depth* to which material obligations must be discharged, are fixed by **KF-2003 · Materiality and Assurance**; KF-3005 or a domain governance profile may set additional thresholds within it.

5.2 Event-Native Conformance Evidence

Conformance evidence is, wherever possible, **emitted by the governed act, not authored separately from it.** The record that a governed act produces *is* its conformance evidence: an Acceptance Record is the evidence of acceptance; a Transition Record is the audit event; an Evaluation Verdict is the test result; a provenance link is the evidence of derivation; a Supersession Link is the evidence of lineage. A distinct evidence artefact is created only where the act's own record cannot carry the required proof. This rule prevents a second knowledge-management problem — the management of evidence that the knowledge was managed — and is enforced architecturally by KF-3001 IC-24 and realised in the transition pattern of KF-3005 §35.

6. Identity Layers

```
Object Identity → Version Identity → Representation Identity → Record Identity
→ Actionable Form Identity → Application Identity → Outcome Identity
```

Identifiers are not interchangeable, and none may be reused for a different governed entity.

7. Authoritative Maintenance

Every information object class has one authoritative maintaining service, which creates or accepts the record, applies authorised change, preserves history, rejects unauthorised mutation, exposes current and historical state, records correction and supersession, and supplies assurance evidence. Other services may reference, annotate, request change, create supporting records or maintain projections — never overwrite the authoritative record. Maintenance responsibility does not confer human decision authority.

7.1 Adjacent Information Responsibilities

There is no dual authoritative ownership. Where two services touch one concept, one holds the authoritative record and the other holds an adjacent, related or derived view. Enforced by IC-14 (Least Necessary Alteration).

Concern	Authoritative record	Authoritative service	Related / derived information
Institutional authority / mandate / delegation	AUT-01, AUT-02	**SVC-06** Authority and Acceptance	KF-3008 governs the authority *model*; SVC-16 assigns the technical permissions through which authorised actions execute
Technical permission	ACS-02	**SVC-16** Access and Integrity	executes actions authorised under AUT-01
Effective period of a substantive record	that record	**its own maintaining service**	SVC-19 governs the common temporal model, sequencing and reconstruction — not each record's validity decision
Version correction	COR-01	**SVC-09** Version and Lineage	temporally reconstructed by SVC-19 (no separate "correction history" record)
Version supersession	SUP-01	**SVC-09** Version and Lineage	release supersession is a distinct assertion held by SVC-10
Release supersession	REL-01/REL-02	**SVC-10** Release	references SUP-01 where a version supersession is involved
Current custody	PRO-03	**SVC-15** Provenance	historical custody sequence reconstructed by SVC-19 (no separate "custody history" record)
Evaluation confidence	EVA-02	**SVC-04** Evaluation	distinct from epistemic confidence
Epistemic standing / confidence	EPS-01	**SVC-05** Justification	distinct from evaluation confidence
Operational exception	EXC-01	**SVC-17** Operating Condition	governance exception GOV-01 is held by SVC-07; SVC-

Concern	Authoritative record	Authoritative service	Related / derived information
Reactivation	REA-01	**SVC-14** Retirement and Preservation	18 reads both enters governed evolution through a Change Record (CHG-01, SVC-09) — there is no separate "reactivation proposal" record

Revalidation is not a separately owned record: it appears as a disposition value in DEP-03 (dependency-driven) and as a condition on REL-01 (release-side).

7.2 Composite Decision Records

A composite decision (for example a release, which relies on an accepted version, valid acceptance, applicable authority, standing governance, dependency disposition and operating condition) **must reference the authoritative records on which it relied — their identity, applicable version or effective state and decision-time status — and must not reproduce their substantive content in a way that creates a competing source of truth.** This rule is load-bearing for KF-3005 gate definitions.

Part II · Relationship Architecture

8. Relationship Types

- **Identity** — VERSION_OF, REPRESENTS_VERSION, RECORD_CONCERNS_OBJECT.
- **Derivation** — DERIVED_FROM, TRANSFORMED_FROM, GENERATED_USING, CONSOLIDATED_FROM.
- **Evidential** — SUPPORTS, CHALLENGES, CONTEXTUALISES, EVALUATES, JUSTIFIES.
- **Governance** — ACCEPTS, STEWARDS, AUTHORIZES_RELIANCE_ON, RELEASES, SUSPENDS, RETIRES.
- **Lifecycle** — TRIGGERED_BY, EVALUATED_UNDER_GATE, TRANSITIONS_FROM, TRANSITIONS_TO, RETURNS_TO.
- **Dependency** — DEPENDS_ON, IMPLEMENTS, USES, CONSTRAINED_BY, INVALIDATED_BY_CHANGE_TO.
- **Operational** — DERIVES_FORM, APPLIES_FORM, PRODUCES_OUTCOME, EXPLAINED_BY, FEEDS_BACK_TO.
- **Historical** — PRECEDES, SUCCEEDS, SUPERSEDES, CORRECTS, BRANCHES_FROM, CONSOLIDATES, REACTIVATES.

9. Relationship Properties

Every material relationship must identify: relationship identity; **source information object**; **target information object**; relationship type; asserting actor/service; recorded time; effective

period where relevant; provenance; status; confidence or qualification where applicable; supersession or correction history. (Relationships may connect records, versions, forms or outcomes — not only Operational Knowledge Objects.)

9.1 Relationship-Type Specification

Each load-bearing relationship is specified through: relationship code; definition; permitted source class; permitted target class; direction; cardinality; symmetry; transitivity; effective-time behaviour; qualification or confidence; authoritative asserting service; correction and withdrawal rule. The most load-bearing relationships:

Relationship	Source → Target	Cardinality	Sym.	Trans.	Asserting service
VERSION_OF	Version → Object	many-to-one (exactly one target)	no	no	SVC-09
REPRESENTS_VERSION	Representation → Version	many-to-one	no	no	SVC-02
SUPPORTS	Evidence → Claim/Version	many-to-many, qualified	no	no	SVC-03
CHALLENGES	Evidence → Claim/Version	many-to-many, qualified	no	no	SVC-03
ACCEPTS	Authority → Version	many-to-one per decision	no	no	SVC-06
RELEASES	Release → Version	one-to-one per release	no	no	SVC-10
DEPENDS_ON	Object/Form/App → Object/Form	many-to-many, directional	no	no (explicit only)	SVC-11
DERIVED_FROM	Info object → source input(s)	many-to-many	no	yes (preserving basis)	SVC-15
APPLIES_FORM	Application → Actionable Form	many-to-one	no	no	SVC-13
PRODUCES_OUTCOME	Application → Outcome	one-to-many	no	no	SVC-13
SUPERSEDES	Entity → prior Entity	one-to-one per scope	no	yes (chain)	service owning the superseded entity

Remaining relationships are completed in a controlled relationship registry using the same template. **SUPERSEDES is a shared relationship type**: its assertion is maintained by the service responsible for the superseded entity (a superseded version by SVC-09, a superseded release by SVC-10, a superseded form by SVC-12).

10. Relationship Integrity Rules

1. Every relationship resolves to valid source and target identities.
2. A relationship must not imply authority beyond its defined semantics.
3. A broken relationship is represented explicitly.
4. Historical relationships are not deleted when superseded.
5. Derived relationships preserve their basis.
6. Symmetry and transitivity are not assumed unless the type defines them.
7. Dependency direction is explicit.
8. Relationship status and object status remain distinct.
9. Relationship correction creates a new governed state or correction record.

Part III · Temporal and Historical Architecture

11. Time Semantics

Event time, **recorded time**, **effective time**, **decision time** and **processing time** may coincide but must not be treated as identical by default.

SVC-19 (Temporal Continuity) governs the common temporal model, historical sequencing and reconstruction rules. Each authoritative maintaining service remains responsible for the effective period and temporal assertions of the records it maintains — SVC-06 for an Acceptance Record's effective period, SVC-10 for a Release Record's, SVC-07 for a Stewardship Assignment's. SVC-19 owns temporal conventions, sequence integrity, reconstruction and migration history; it does not approve or maintain each substantive validity decision.

12. Effective Periods

A record governing continuing conditions must support effective-from, effective-to, open-ended validity, scheduled future effect, retrospective recording, revocation, supersession, suspension and reactivation. **A later recorded fact may have an earlier effective time**, so reconstruction must not rely solely on record-creation order.

13. Historical Reconstruction

The architecture must reconstruct, at a specified time: substantive representation; epistemic standing; acceptance and release status; steward and applicable authority; lifecycle state; known dependencies; actionable forms; operating and integrity condition; relevant exceptions; and historical application eligibility. Reconstruction must distinguish *what the system knew then* from *what the system knows now about then*.

14. Mutation, Correction and Supersession

Governed records are not silently overwritten. Permitted patterns: append; new immutable version; status transition; explicit correction; supersession; revocation; expiration; historical annotation. A correction must preserve the original record, correction reason, correcting actor/service, correction time, corrected content or interpretation, and the relationship between original and corrected state.

15. Deletion and Disposition

Logical deletion must not erase information required for identity continuity, accountability, provenance, historical reconstruction, legal/policy retention or conformance evidence. Disposition may include active/restricted retention, archival preservation, anonymisation, de-identification, cryptographic erasure of protected content, or deletion after retention expiry. The record of disposition remains governed where accountability requires it.

15.1 Current State and Event History

Where both a current-state record and an event history exist:

1. the event or transition record preserves the authoritative historical act;
2. the current-state record represents the effective current state;
3. both are updated or established within one governed operation;
4. disagreement between them produces an indeterminate integrity condition;
5. current state must be reconstructable from valid historical events;
6. correction does not erase the original event.

This applies to lifecycle state, release status, form status, epistemic standing, stewardship and operating condition. In each case the current-state record is an effective-dated assertion established transactionally with its event; it is not a free-standing mutable value.

Part IV · Domain A — Identity and Representation Information

Record Family A1 · Object Identity — *maintained by SVC-01 · Object Registry*

OBJ-01 · Object Identity Record *(Core Record).* The authoritative record of an object's enduring identity. *Required properties:* object identifier; object class; registration status; origin reference; registration time; registration decision; identity-resolution status; applicable predecessor or source relationship; current lifecycle-state projection or reference *(authoritative state is LIF-01, SVC-08)*. *Principal relationships:* HAS_VERSION; ORIGINATED_FROM; BRANCHES_FROM; CONSOLIDATES; HAS_STEWARD; HAS_LIFECYCLE_STATE. *Mutation rule:* object identifier immutable; classification and descriptive metadata may change through governed correction. *Conformance test:* **every governed object resolves to exactly one enduring Object Identity Record, regardless of lifecycle state.** *Protects:* INV-01, IC-05.

OBJ-02 · Intake Record *(Supporting).* Submitted material, source, intake context, initiating actor, classification request, duplicate-search result, routing decision.

OBJ-03 · Identity Resolution Decision *(Supporting).* Whether intake represents a new object, candidate version, branch candidate, consolidation candidate, duplicate or unresolved identity.

Record Family A2 · Representation — *maintained by SVC-02 · Representation*

REP-01 · Structured Representation *(Supporting).* Substantive content, semantic structure, constituent claims/elements, representation modality, limitations, object and version identity.

REP-02 · Semantic Type Record *(Controlled Vocabulary).* Semantic classification applied to an object, representation, claim or relationship.

REP-03 · Representation Limitation Record *(Supporting).* Known opacity, incompleteness, approximation, modality limitation, interpretation constraint, unsupported use.

REP-04 · Object Relationship Record *(Relationship Record).* Governed semantic relationships among objects, versions or elements.

Record Family A3 · Resolution and Discovery — *maintained by SVC-01 · Object Registry (RES-02 is a projection)*

RES-01 · Resolution Result *(Projection).* Query/identifier supplied; target resolved; ambiguity; resolution basis; current or historical context; access filtering; retrieval time.

RES-02 · Governed Context Package *(Projection).* A non-authoritative assembled view referencing identity, version, representation, epistemic standing, acceptance, release, lifecycle state, provenance, stewardship and restrictions.

RES-03 · Retrieval Provenance Record *(Supporting).* Requested information; requester; source records accessed; projection/cache use; version returned; retrieval time.

Part V · Domain B — Evidence and Justification Information

Record Family B1 · Evidence — *maintained by SVC-03 · Evidence*

EVD-01 · Evidence Record *(Supporting).* Evidence identifier; type; origin; creator/source; content or source reference; custody; availability; integrity status; effective period where applicable.

EVD-02 · Evidence Relationship Record *(Relationship Record).* Relates evidence to claim, object, version, evaluation or justification; role: supports, challenges, limits, contextualises, corroborates. **The "counter" quality belongs to this relationship role, not to the evidence object** — the same evidence may support one claim and challenge another.

EVD-03 · Material Counter-Evidence Assertion *(Supporting).* Exists only where a material challenge must be made visible in acceptance and review. It preserves INV-06 without implying that an item is intrinsically "counter-evidence." *Protects:* INV-06.

EVD-04 · Evidence Availability Status *(Status Assertion).* Available, restricted, unavailable, inaccessible, destroyed, unresolved, provenance-incomplete.

Record Family B2 · Evaluation — *maintained by SVC-04 · Evaluation*

EVA-01 · Evaluation Criteria Record *(Supporting).* Criterion; applicability; threshold; scoring/judgement method; effective period; exception rule.

EVA-02 · Quality Assessment Record *(Supporting).* Object/evidence assessed; criteria applied; evaluator; inputs; findings; **evaluation confidence** *(distinct from epistemic confidence — §7.1)*; limitations; assessment time. Includes an **assessment-progression** value: not assessed, pending, in progress, completed, reassessment required.

EVA-03 · Evaluation Verdict Record *(Status Assertion).* Pass, fail, return, pending, indeterminate; never inferred solely from score. *Protects:* INV-24.

EVA-04 · Evaluation Exception Record *(Supporting).* A permitted procedural deviation that does not convert an invariant violation into conformance.

Record Family B3 · Justification and Epistemic Standing — *maintained by SVC-05 · Justification*

JUS-01 · Justification Record *(Supporting).* Structured warrant referencing claims, evidence, counter-evidence assertions, evaluations, rationale, assumptions, unresolved challenges, confidence, limitations. *Protects:* INV-05, IC-09.

JUS-02 · Assumption Record *(Supporting).* Statement; source/basis; scope; confidence; materiality; review condition.

JUS-03 · Rationale Record *(Supporting).* The reasoning connecting evidence and assumptions to a conclusion.

JUS-04 · Challenge Record *(Supporting).* A resolved or unresolved challenge to the warrant.

EPS-01 · Epistemic Standing Record *(Status Assertion).* The current epistemic conclusion: unsupported, provisionally supported, supported, strongly supported, weakened, indeterminate. Independent of lifecycle, acceptance and release, **and independent of assessment progression (held in EVA-02) and contestation (EPS-03)**. *Protects:* INV-03, INV-07.

EPS-02 · Epistemic Standing Change Record *(Supporting).* Prior standing; new standing; triggering evidence or assessment; responsible actor/service; recorded and effective time.

EPS-03 · Contestation Status Record *(Status Assertion).* Not contested, contested, materially contested, resolved. An object may be *supported* and *contested* simultaneously; these are separate dimensions.

Part VI · Domain C — Governance and Change Information

Record Family C1 · Authority and Acceptance — *maintained by SVC-06 · Authority and Acceptance*

AUT-01 · Authority Mandate Record *(Supporting).* Authority holder or accountable body; source; mandate; scope; effective period; delegation; constraints; revocation status. KF-3008 defines the full authority model; KF-3004 defines the information to represent it. *(Held by SVC-06, not SVC-16: institutional authority is not technical permission.)*

AUT-02 · Accountable Body Record *(Supporting).* Body identity; formal mandate; membership; quorum or decision rule; effective period; delegated authority; decision-record requirements.

ACC-01 · Acceptance Record *(Core Record).* Object version accepted; accepting authority; decision; reliance scope; conditions; duration; justification reference; epistemic standing at decision time; decision rationale. *Protects:* INV-05, INV-08, INV-09.

ACC-02 · Refusal Record *(Supporting).* The refusal decision and reasons, without changing epistemic standing.

ACC-03 · Acceptance Condition Record *(Supporting).* Restricted use; provisional reliance; mandatory review; limited population; geographic scope; expiry; human-oversight requirement.

Record Family C2 · Stewardship and Review — *maintained by SVC-07 · Stewardship and Review*

STW-01 · Stewardship Assignment *(Supporting).* Object; steward; mandate; start/end time; assignment authority; responsibility scope; transfer history. *Protects:* INV-09, INV-10.

STW-02 · Stewardship Transfer Record *(Supporting).* Preserves transfer without creating an ownerless period.

REV-01 · Review Schedule *(Supporting).* Review type; frequency; trigger; due date; responsible party; escalation rule.

REV-02 · Review Record *(Supporting).* Review scope; reviewer; evidence considered; findings; disposition; required actions.

CMP-01 · Compliance Assessment *(Supporting).* Distinguishes formal compliance from substantive review.

GOV-01 · Governance Exception Record *(Supporting).* Rule or principle affected; scope; authority; rationale; duration; compensating control; expiry; review status. *(Distinct from operational exception EXC-01 — §7.1.)*

Record Family C3 · Lifecycle and Transition — *maintained by SVC-08 · Lifecycle and Transition*

LIF-01 · Lifecycle State Record *(Status Assertion).* The authoritative current lifecycle state, established transactionally with its transition (§15.1). States are defined by KF-3005 and are not

inferred from epistemic standing, release, version status or application activity. *Protects:* INV-03, IC-01.

EVT-01 · Lifecycle Event Record *(Supporting).* A material event capable of triggering review, transition, suspension, evolution or retirement.

GAT-01 · Gate Decision Record *(Supporting).* Transition identifier; current state; proposed state; applicable gate; criteria; evidence; result; accountable authority; operating condition; decision time.

TRN-01 · Transition Record *(Core Record).* Transition identifier; prior state; resulting state; execution actor/service; accountable authority; gate-decision reference; effective time; completion status. **Gate Decision and Transition Records are transactionally bound** under one transition identifier. *Protects:* INV-08, INV-14, IC-01, IC-02.

Record Family C4 · Version, Lineage and Change — *maintained by SVC-09 · Version and Lineage*

VER-01 · Object Version Record *(Core Record).* Version identifier; object identifier; predecessor version; change rationale; creation event; substantive fingerprint; candidate or governed status; correction and supersession links. *Integrity rule:* substantive content immutable after establishment. *Protects:* INV-12, IC-06.

CHG-01 · Change Record *(Supporting).* Proposed change; trigger; rationale; affected content; responsible actor; resulting version or object; review route. Also the record through which a REA-01 reactivation enters governed evolution.

LIN-01 · Lineage Record *(Relationship Record).* Predecessor; successor; branch; consolidation; correction; supersession; reactivation. *Protects:* INV-13.

BRN-01 · Branch Record *(Supporting).* Creation of a new enduring object identity derived from an existing object *(successor identity assigned via SVC-01; IC-07)*.

CON-01 · Consolidation Record *(Supporting).* Creation of one successor object from multiple predecessors.

COR-01 · Correction Record *(Supporting).* Preserves the original record and the correction. **The authoritative record of correction**; SVC-19 reconstructs correction history from COR-01 plus recorded time, effective time, supersession relationships and provenance — there is no separate correction-history record. *Protects:* INV-14, IC-21.

SUP-01 · Supersession Record *(Relationship Record).* Version and lineage supersession. Supersession of a release or form is asserted by the service owning that entity (SVC-10, SVC-12) — see §9.1.

Record Family C5 · Release — *maintained by SVC-10 · Release*

REL-01 · Release Record *(Core Record).* Released object version; acceptance reference; release authority; release scope; effective period; conditions; status; withdrawal or supersession reference; revalidation condition. A composite decision record per §7.2. *Protects:* INV-17, IC-08.

REL-02 · Release Status Record *(Status Assertion).* Candidate, active, restricted, suspended, withdrawn, expired, superseded. Independent of object lifecycle state.

Record Family C6 · Dependency and Impact — *maintained by SVC-11 · Dependency*

DEP-01 · Dependency Record *(Relationship Record).* Material reliance between objects, versions, representations, actionable forms, applications or controls.

DEP-02 · Dependency Impact Record *(Supporting).* Triggering change; affected dependants; impact classification; analysis basis; uncertainty; required action. *Protects:* INV-15.

DEP-03 · Dependency Disposition Record *(Supporting).* No action, review, revalidation, evolution, restriction, suspension, withdrawal, retirement.

Part VII · Domain D — Operational Use Information

Record Family D1 · Actionable Forms — *maintained by SVC-12 · Operationalisation*

ACT-01 · Actionable Form Record *(Core Record).* Form identity; source object and version; form type; derivation method; warrant linkage; release status; intended context; prohibited use; effective period. *Protects:* INV-17, IC-10, IC-12.

ACT-02 · Derivation Record *(Supporting).* How the form was derived from its source knowledge.

ACT-03 · Actionable Form Release Record *(Supporting).* Form eligibility, independent of the underlying object release.

ACT-04 · Form Status Record *(Status Assertion).* Provisional, under review, eligible, restricted, suspended, withdrawn, superseded.

Record Family D2 · Application — *maintained by SVC-13 · Application and Outcome*

APP-01 · Application Record *(Core Record).* Application identity; form identity; source object version; **applicable warrant/justification reference (resolved via the actionable form)**; purpose; context; actor or system; effective eligibility decision; inputs; output or action; time; materiality. *Protects:* INV-18, INV-19, IC-12.

APP-02 · Eligibility Decision Record *(Supporting).* Whether application was permitted at the time of use. *Protects:* IC-11.

APP-03 · Use Context Record *(Supporting).* Population; environment; jurisdiction; process; materiality; risk; constraints; human-oversight condition.

APP-04 · Application Participant Record *(Supporting).* Initiating human; accountable human; executing service; computational model; reviewing human; affected party where governed.

EXP-01 · Explanation Record *(Supporting).* Proportionate to modality, materiality, risk, decision context and governance profile. *Protects:* INV-20.

Record Family D3 · Outcome and Feedback — *maintained by SVC-13 · Application and Outcome*

OUT-01 · Outcome Record *(Supporting).* An observed result; observation distinguishable from interpretation.

OUT-02 · Outcome Interpretation Record *(Supporting).* The meaning attributed to an outcome and its basis.

OUT-03 · Materiality Assessment *(Supporting).* No action, monitoring, review, evidence registration, restriction, suspension, evolution, retirement.

FDB-01 · Feedback Record *(Supporting).* Routes material learning back into evidence, review, evaluation or evolution.

Record Family D4 · Retirement and Preservation — *maintained by SVC-14 · Retirement and Preservation*

RET-01 · Retirement Record *(Core Record).* Object retired; retirement authority; rationale; effective time; successor where applicable; preservation disposition; reactivation conditions. *Protects:* INV-16, IC-04.

PRE-01 · Preservation Record *(Supporting).* How the retired object's identity, versions, warrant, governance history, forms, applications, outcomes and provenance remain recoverable.

REA-01 · Reactivation Request *(Supporting).* Initiates governed evolution through CHG-01; does not restore operational status directly.

HIS-01 · Historical Object Record *(Projection).* A governed historical view or package used for retrieval after retirement.

Part VIII · Domain E — Assurance and Continuity Information

Record Family E1 · Provenance and Custody — *maintained by SVC-15 · Provenance and Traceability*

PRO-01 · Provenance Record *(Core Record).* Origin, derivation, attribution and custody across material information objects (why-, where-, how-, who-provenance). *Protects:* INV-04, IC-20.

PRO-02 · Derivation Link *(Relationship Record).* Relates a derived information object to its source inputs and transformation event.

PRO-03 · Custody Record *(Supporting).* A custody event or custody assignment. The ordered historical custody sequence is reconstructed by SVC-19 from these events.

PRO-04 · Provenance Gap Record *(Supporting).* Missing, incomplete, disputed or inaccessible provenance; never represented as complete lineage.

Record Family E2 · Access and Integrity — *maintained by SVC-16 · Access and Integrity*

ACS-01 · Access Policy Record *(Supporting).* Permitted access and action by actor/role, object class, service, information classification, action, context, effective period.

ACS-02 · Permission Assignment *(Supporting).* Technical permission, independent of institutional authority (AUT-01). *Protects:* IC-13.

ACS-03 · Disclosure Constraint *(Supporting).* Confidentiality, sharing and use limitations.

INT-01 · Integrity Status Record *(Status Assertion).* Verified, altered, corrupted, incomplete, unverifiable, under investigation.

ADM-01 · Administrative Action Record *(Supporting).* Material privileged actions affecting permissions, policies, configuration, retention, integrity or access. *Protects:* INV-11, IC-15.

Record Family E3 · Operating Condition and Exception — *maintained by SVC-17 · Operating Condition and Exception*

OPR-01 · Operating Condition Record *(Status Assertion).* Operational, degraded, unavailable, pending, failed, indeterminate. *Protects:* INV-24.

OPR-02 · Degradation Record *(Supporting).* Affected capability/service; scope; start time; consequence; fail-open or fail-closed rule; compensating control; recovery status.

OPR-03 · Failure Disposition Record *(Supporting).* The governed response to a failure or indeterminate condition.

EXC-01 · Operational Exception Record *(Supporting).* Distinguished from governance exceptions (GOV-01) and assurance exception references. *Protects:* IC-19.

REC-01 · Recovery and Reconciliation Record *(Supporting).* Recovered operations; deferred records; reconciliation checks; residual discrepancies; closure authority.

Record Family E4 · Temporal Continuity — *maintained by SVC-19 · Temporal Continuity*

TMP-01 · Temporal Metadata Record *(Supporting).* Recorded, event, effective, decision and processing time where applicable. *Protects:* IC-22.

TMP-02 · Effective Period Record *(Supporting).* The common effective-period *model* and scheduled-effect conventions. Substantive validity is asserted by each maintaining service (§11); TMP-02 provides the shared representation, not the decision.

TMP-03 · Historical Reconstruction Result *(Projection).* A generated, non-authoritative view of the object and its context at a specified point or interval.

TMP-04 · Retention Condition Record *(Supporting).* Retention, archive, restriction and disposition conditions.

TMP-05 · Format or Custody Migration Record *(Supporting).* Preserves authenticity through an actual format or custody migration event.

Record Family E5 · Conformance and Audit — *maintained by SVC-18 · Conformance and Audit*

CNF-01 · Conformance Assessment *(Supporting).* Scope, criteria, evidence, assessor, conclusion. *Protects:* INV-21, INV-22.

CNF-02 · Invariant Test Result *(Supporting).* Holds, fails, indeterminate, outside scope.

CNF-03 · Interaction Constraint Test Result *(Supporting).* Whether a prohibited or mandatory service path was respected.

CNF-04 · Assurance Finding *(Supporting).* No action, monitoring, correction, control improvement, suspension, formal non-conformance.

CNF-05 · Non-Conformance Record *(Core Record).* Violated obligation; affected scope; severity; evidence; corrective action; accountable authority; status.

CNF-06 · Corrective Action Record *(Supporting).* Required remediation and verification. SVC-18 must not alter the operational records assessed. *Protects:* INV-23, IC-17.

Part IX · Information Integrity Rules

16. Object and version integrity. One enduring identity per object; every version belongs to one object; version content immutable after establishment; new versions do not create new object identities; branching and consolidation create successor identities; release and acceptance do not create versions; retirement does not erase identity.

17. Record integrity. Every material event and decision (per §5.1) has a governed record; records are attributable; material records preserve applicable authority; corrections do not erase originals; record status is explicit; missing information is not represented as affirmative evidence; projections do not supersede authoritative records; records under assurance review cannot be silently altered; a composite decision references rather than copies the records it relied on (§7.2).

18. State separation. These remain distinct, none inferred solely from another unless KF-3005 defines a formal derivation rule: object lifecycle state; version status; representation status; epistemic standing; assessment progression; contestation; acceptance status; release status; actionable-form status; service operating condition; conformance status.

19. Authority and permission separation. A permission record (ACS-02, SVC-16) establishes that an actor/service may technically perform an action; an authority record (AUT-01, SVC-06) establishes that a human authority or accountable body may make a governed decision. Both may be required; neither substitutes for the other.

20. Provenance integrity. Every material information object preserves or resolves to origin, creator/source, derivation, custody, related object and version, material transformation and known gaps. Broken provenance is explicit.

21. Temporal integrity. Recorded time is never substituted for effective time; retrospective recording is identified; future-effective decisions are not treated as currently effective; expired

decisions do not remain silently active; reconstruction preserves knowledge available at the target time; later correction is distinguishable from the earlier state; sequence ambiguity is represented as indeterminate where unresolved.

Part X · Information Interaction Contracts

22. Record creation. A service may create an authoritative record only where KF-3003 assigns it maintenance responsibility. **23. Record reference.** A referencing service preserves target identity, target version or effective state, relationship type, reference time and resolution status. **24. Change request.** A service may request but not directly apply change to another service's record. **25. Event publication.** Material creation, change, suspension, withdrawal, correction and retirement events are made available to affected services without transferring authority or mutating receiving records. **26. Projection and caching.** A projection identifies authoritative source, source version or effective state, creation time, freshness and invalidation conditions and access restrictions; a stale projection is never presented as current authoritative state.

Part XI · Information Conformance

27. Information conformance. Every required object class is represented; every authoritative record has one maintaining service; identity semantics are preserved; required relationships are resolvable; current and historical states are distinguishable; correction and supersession preserve prior states; temporal semantics are explicit; protected information is access-controlled; missing or indeterminate information is visible; and information supports all required invariant and interaction-constraint tests.

28. Information conformance evidence. Logical information model; authoritative-maintainer matrix; identity tests; relationship-integrity tests; temporal-reconstruction tests; correction and supersession tests; version-immutability tests; state-separation tests; access and authority checks; provenance-path reconstruction; cross-service reconciliation; retention and disposition evidence.

29. Representative reconstruction test.

```
Outcome → Application → Actionable Form → Released Object Version →
Acceptance Decision → Epistemic Standing → Justification →
Evidence and Counter-Evidence → Representation → Object Identity and
Origin
```

The reconstruction must also reveal the responsible human authority, applicable steward, lifecycle transitions, dependencies, access and integrity condition, operating condition, known provenance gaps, relevant exceptions and conformance findings.

Part XII · Crosswalks

30. Information-Object-to-Service Crosswalk *(frozen KF-3003 catalogue; one maintainer per family)*

Record family	Authoritative maintaining service
A1 Object Identity (OBJ)	SVC-01 · Object Registry
A2 Representation (REP)	SVC-02 · Representation
A3 Resolution and Discovery (RES)	SVC-01 · Object Registry
B1 Evidence (EVD)	SVC-03 · Evidence
B2 Evaluation (EVA)	SVC-04 · Evaluation
B3 Justification and Epistemic Standing (JUS, EPS)	SVC-05 · Justification
C1 Authority and Acceptance (AUT, ACC)	SVC-06 · Authority and Acceptance
C2 Stewardship and Review (STW, REV, CMP, GOV)	SVC-07 · Stewardship and Review
C3 Lifecycle and Transition (LIF, EVT, GAT, TRN)	SVC-08 · Lifecycle and Transition
C4 Version, Lineage and Change (VER, CHG, LIN, BRN, CON, COR, SUP)	SVC-09 · Version and Lineage
C5 Release (REL)	SVC-10 · Release
C6 Dependency and Impact (DEP)	SVC-11 · Dependency
D1 Actionable Forms (ACT)	SVC-12 · Operationalisation
D2 Application (APP, EXP)	SVC-13 · Application and Outcome
D3 Outcome and Feedback (OUT, FDB)	SVC-13 · Application and Outcome
D4 Retirement and Preservation (RET, PRE, REA, HIS)	SVC-14 · Retirement and Preservation
E1 Provenance and Custody (PRO)	SVC-15 · Provenance and Traceability
E2 Access and Integrity (ACS, INT, ADM)	SVC-16 · Access and Integrity
E3 Operating Condition and Exception (OPR, EXC, REC)	SVC-17 · Operating Condition and Exception
E4 Temporal Continuity (TMP)	SVC-19 · Temporal Continuity
E5 Conformance and Audit (CNF)	SVC-18 · Conformance and Audit

31. Capability-to-Information Crosswalk

Capability	Principal information objects
A-CAP-01 Object Registration	OBJ-01, OBJ-02, OBJ-03
A-CAP-02 Structured Representation	REP-01, REP-02, REP-03, REP-04
A-CAP-03 Resolution and Discovery	RES-01, RES-02, RES-03
B-CAP-01 Evidence Registration	EVD-01, EVD-02, EVD-03, EVD-04

Capability	Principal information objects
B-CAP-02 Quality Evaluation	EVA-01, EVA-02, EVA-03, EVA-04
B-CAP-03 Justification	JUS-01, JUS-02, JUS-03, JUS-04
B-CAP-04 Epistemic Status	EPS-01, EPS-02, EPS-03
C-CAP-01 Authority and Acceptance	ACC-01, ACC-02, ACC-03, AUT-01, AUT-02
C-CAP-02 Stewardship	STW-01, STW-02
C-CAP-03 Review and Compliance	REV-01, REV-02, CMP-01, GOV-01
C-CAP-04 Version	VER-01, CHG-01, COR-01
C-CAP-05 Release	REL-01, REL-02
C-CAP-06 Lineage and Evolution	LIN-01, BRN-01, CON-01, SUP-01, CHG-01
C-CAP-07 Dependency Impact	DEP-01, DEP-02, DEP-03
C-CAP-08 Event, Gate and Transition	EVT-01, GAT-01, TRN-01, LIF-01
D-CAP-01 Actionable Form	ACT-01, ACT-02, ACT-03, ACT-04
D-CAP-02 Application	APP-01, APP-02, APP-03, APP-04, EXP-01
D-CAP-03 Outcome	OUT-01, OUT-02, OUT-03, FDB-01
D-CAP-04 Retirement	RET-01, PRE-01, REA-01, HIS-01
E-CAP-01 Provenance	PRO-01, PRO-02, PRO-03, PRO-04
E-CAP-02 Conformance	CNF-01…CNF-06
E-CAP-03 Access and Integrity	ACS-01, ACS-02, ACS-03, INT-01, ADM-01
E-CAP-04 Operating Condition	OPR-01, OPR-02, OPR-03, EXC-01, REC-01
E-CAP-05 Temporal Continuity	TMP-01, TMP-02, TMP-03, TMP-04, TMP-05

32. Invariant-to-Information Crosswalk

Invariant	Demonstrating records
INV-01 Unique Object Identity	OBJ-01
INV-02 Representation Traceability	REP-01, VER-01
INV-03 Logical Distinction	OBJ-01, VER-01, EPS-01, ACC-01, REL-02, LIF-01
INV-04 Provenance Persistence	PRO-01, PRO-03, PRO-04
INV-05 Justification before Reliance	JUS-01, ACC-01, GAT-01
INV-06 Counter-Evidence Visibility	EVD-02 (challenge role), EVD-03, JUS-04
INV-07 Acceptance Does Not Rewrite Standing	EPS-01, EPS-02, ACC-01
INV-08 Attributable Transition	TRN-01, GAT-01
INV-09 Human Accountability	ACC-01, STW-01, AUT-01, TRN-01
INV-10 Standing Governance	STW-01, REV-02, CMP-01
INV-11 Authorised Change	ADM-01, ACS-02, AUT-01, CHG-01, TRN-01, COR-01

Invariant	Demonstrating records
INV-12 Version Preservation	VER-01, COR-01
INV-13 Lineage Preservation	LIN-01, BRN-01, CON-01, SUP-01
INV-14 No Silent Mutation	CHG-01, COR-01, TRN-01
INV-15 Dependency Disposition	DEP-02, DEP-03
INV-16 Retirement Preserves History	RET-01, PRE-01, HIS-01
INV-17 Warranted Actionable Form	ACT-01, ACT-03, REL-01
INV-18 Governed Application	APP-01, APP-02
INV-19 Outcome Traceability	OUT-01, APP-01, PRO-01
INV-20 Modality-Appropriate Explainability	EXP-01, REP-03
INV-21 Inspectable Conformance	CNF-01, CNF-02, CNF-03
INV-22 Assurance Independence	CNF-01, AUT-01, ACS-02, ADM-01, APP-04 (assurance participant)
INV-23 Audit Non-Interference	CNF-01, CNF-03, ACS-01, ACS-02, ADM-01, INT-01 (integrity fingerprints)
INV-24 No Silent Success	OPR-01, EVA-03, GAT-01, CNF-02

33. Interaction-Constraint-to-Information Crosswalk

Constraint	Records and relationships
IC-01 Lifecycle State Authority	LIF-01, TRN-01
IC-02 Gate Enforcement	GAT-01 + TRN-01 (transactionally bound)
IC-03 Suspension Ownership	TRN-01, REL-02, ACT-04, OPR-03, GOV-01, DEP-03
IC-04 Reactivation through Evolution	REA-01, CHG-01, ACC-01
IC-05 Identity Immutability	OBJ-01
IC-06 Version Immutability	VER-01
IC-07 Branch and Consolidation Identity	BRN-01, CON-01, LIN-01
IC-08 Release Separation	REL-01 (distinct from VER-01), ACC-01 ref
IC-09 Justification–Acceptance Separation	JUS-01 vs ACC-01
IC-10 Operationalisation Eligibility	ACT-01 (warrant linkage)
IC-11 Application Eligibility	APP-02
IC-12 Warrant Link Preservation	ACT-01, APP-01 (warrant reference)
IC-13 Authority Is Not Permission	AUT-01 (SVC-06) vs ACS-02 (SVC-16)
IC-14 Least Necessary Alteration	§30 authoritative-maintainer matrix; §7.1
IC-15 Material Administrative Action	ADM-01
IC-16 Automated Authority Boundary	ACC-01, APP-04
IC-17 Read-Only Assurance Boundary	CNF-01…06 (no mutation of assessed records)

Constraint	Records and relationships
IC-18 No Sole Self-Certification	CNF-01 (independent evidence)
IC-19 Exception Visibility	EXC-01, GOV-01, OPR-03
IC-20 Provenance Continuity	PRO-01, PRO-02
IC-21 Append-Only Correction	COR-01, supersession/correction relationship, TMP-01, PRO-01
IC-22 Time Semantics Preservation	TMP-01, TMP-02

34. Lifecycle-to-Information Crosswalk *(KF-2001 stages)*

Lifecycle stage	Required records
Capture	OBJ-01, OBJ-02, OBJ-03, PRO-01
Structure	REP-01, REP-02, VER-01
Evaluate	EVA-01, EVA-02, EVA-03
Justify	JUS-01, JUS-02, JUS-03, EPS-01
Accept (transition)	ACC-01, GAT-01, TRN-01
Release (eligibility)	REL-01, REL-02 *(shown as an eligibility step/gate between Accept and Operationalise, not part of Operationalise)*
Operationalise	ACT-01, ACT-03
Govern (standing)	STW-01, REV-02, CMP-01, GOV-01
Apply	APP-01, APP-02, APP-03, APP-04, EXP-01, OUT-01
Evolve	CHG-01, LIN-01, DEP-02
Retire	RET-01, PRE-01, HIS-01
Cross-cutting (all stages)	EVT-01, GAT-01, TRN-01 (transitions); PRO-01–04 (provenance); TMP-01–05 (temporal); OPR-01 (condition); CNF-01–06 (assurance)

Part XIII · Relationship to Subsequent Documents

KF-3005 defines lifecycle states, gate conditions, transition rules and permissible/prohibited state changes; KF-3004 defines the records through which those controls are represented and evidenced, and the current-state/event-history rule (§15.1) they must honour. **KF-3008** defines the authority model behind AUT-01/02 and the delegation, stewardship, access and computational-participation assignments; KF-3004 defines the logical records that represent them. **KF-3007** defines the tests and evidence for information, service and system conformance. **KF-3009** preserves traceability from Canonical Principle through practice, invariant, capability, service, information object, control and implementation.

KF-4000 may implement this logical architecture through relational, graph, document, event-stream, object-store, ledger, file or human-maintained structures, and may denormalise,

replicate, index or distribute — provided it does not create conflicting authoritative records, collapse required semantic distinctions, overwrite immutable history, remove required provenance, substitute a projection for authoritative state, make historical reconstruction impossible, treat missing or indeterminate information as satisfied, or prevent independent conformance inspection.

Closing Statement

The Governed Information and Record Model defines the memory of the Knowledge Foundry architecture. It ensures that substantive knowledge, warrant, authority, operational use and assurance remain distinguishable yet connected; that every material decision and transition is evidenced; that history survives change; and that the architecture can reconstruct not merely what it currently holds, but how that state came to exist and why it was permitted to be relied upon. Each record family resolves to exactly one authoritative maintainer, each historical view is derived rather than duplicated, each status represents one conceptual dimension, and each crosswalk demonstrates the obligation it claims.

KF-3005 · Lifecycle and Control Architecture

The State, Transition, Gate and Control Model of the Knowledge Foundry

Document Class: Reference Architecture · Parent Document: KF-3000 · Governed By: KF-3001 · Derived From: KF-2001, KF-2002, KF-3002, KF-3003 and KF-3004 · Status: Draft · Version: 0.2

```
Canon and Method (KF-1000 / KF-2001 / KF-2002)
        ↓
Architectural Rules (KF-3001)
        ↓
Capabilities and Services (KF-3002 / KF-3003)
        ↓
Governed Information (KF-3004)
        ↓
▸ Lifecycle and Control Architecture (KF-3005)
        ↓
Conformance, Authority and Traceability (KF-3007 / KF-3008 / KF-3009)
        ↓
Implementation (KF-4000)
```

Editorial Position

KF-3005 is the **behavioural architecture** of the Knowledge Foundry. The documents before it are largely structural: KF-3001 states what must remain true, KF-3002 what the architecture must be able to do, KF-3003 where logical responsibility resides, and KF-3004 what information exists and how it is preserved. KF-3005 defines *how governed change is permitted and executed*.

It is therefore neither a restatement of the nine-stage methodology lifecycle in KF-2001 nor a workflow specification for a particular implementation. KF-2001 defines the lifecycle **method**; KF-3005 defines the architectural **machinery** that makes that lifecycle enforceable.

Purpose

KF-3005 defines the behavioural architecture through which governed entities move between valid states. It specifies lifecycle domains; state machines; permitted transitions; transition-request semantics; gate compositions; reusable controls; execution and persistence rules; exception and degraded-state behaviour; recovery and reconciliation; concurrency and conflict handling; and control and transition evidence.

KF-3004 defines *what* governed information exists. KF-3005 defines *how* that information may change. KF-3005 introduces no new authoritative information classes; it uses the KF-3004 objects and governs their creation, update, correction, status change and historical preservation.

The document is centred on the **transition** rather than on the state. A state is a governed condition; a transition is the operation through which the architecture changes that condition. Accordingly, KF-3005 answers a single, narrow question:

Under what conditions may a governed entity move from one state to another; which authority may permit that movement; which controls must be satisfied; how is the transition executed; and what evidence must remain afterwards?

Alignment and Reconciliation Note

This document is reconciled against the **nineteen-service catalogue** shared by KF-3002 and the KF-3004 v0.2 record model (SVC-01 … SVC-19), in which the authoritative owners referenced here are: **SVC-05** Justification (epistemic standing); **SVC-06** Authority and Acceptance; **SVC-08** Lifecycle and Transition; **SVC-09** Version and Lineage; **SVC-10** Release; **SVC-12** Operationalisation; **SVC-13** Application and Outcome; **SVC-14** Retirement and Preservation; **SVC-15** Provenance; **SVC-16** Access and Integrity; **SVC-17** Operating Condition and Exception; **SVC-18** Conformance and Audit; **SVC-19** Temporal Continuity. All service, record (KF-3004), invariant (KF-3001, `INV-*`) and interaction-constraint (`IC-*`) references in this document use that catalogue and the KF-3004 v0.2 record codes, including its §15.1 current-state / event-history rule and its §7.2 composite-decision rule.

Upstream caution. Where the scaffold named services that do not exist under a single owner — an "Epistemic Status Service," a standalone "Release Service," an "Operating Condition and Exception Service" — this document binds them to the frozen catalogue owners above. Any copy of KF-3003 that still enumerates only sixteen services predates this catalogue and should be read as superseded by the KF-3002 / KF-3004 v0.2 service model; the mismatch is an upstream KF-3003 versioning item, not a KF-3005 divergence.

Changes in v0.2

Relative to v0.1: the abnormal-condition vocabulary (failure, exception, violation, degradation, indeterminate, conflict) is consolidated into a single distinction table (§3); the "no single universal state machine" and "recovery is forward restoration, not historical erasure" principles are made explicit (§7, §44); "gates orchestrate reusable controls" is stated directly with a worked decomposition (§29); the three governed catalogues — Transition, Gate and Control — are named as an organising principle (§21); and the mis-labelled state-assertion clause is relocated to its proper place at the end of the state architecture (§20). Substantive content, identifiers and crosswalks from v0.1 are preserved.

Part I · Behavioural Architecture Foundations

1. Scope

KF-3005 governs changes involving Operational Knowledge Objects; object versions; representations; epistemic standing; acceptance; release; stewardship and governance standing; actionable forms; applications; retirement; service operating condition; and conformance standing.

It excludes organisational workflow design; user-interface sequence; technology-specific transaction management; physical event-bus design; database locking; service deployment; detailed authority assignment (KF-3008); and assurance methodology (KF-3007).

2. Core Behavioural Concepts

2.1 Lifecycle — the permitted broad phases through which a governed entity may pass. **2.2 State** — the effective governed condition of one entity within one state machine. **2.3 Transition** — a governed operation attempting to move an entity from a valid source state to a permitted target state. **2.4 Gate** — the authoritative decision composition determining whether a proposed transition may execute. **2.5 Control** — a reusable mechanism that validates, prevents, detects, constrains, records or evidences one condition required by a gate or transition. **2.6 Trigger** — an event, request, decision, elapsed period or condition initiating consideration of a transition. **2.7 Transition Request** — the governed proposal to perform a specific transition on an identified entity. **2.8 Exception** — a formally approved, bounded departure from normal control behaviour that does not violate an invariant. **2.9 Failure** — an unsuccessful attempt to execute an otherwise valid operation. **2.10 Violation** — a breach of a Canonical Principle, invariant or mandatory interaction constraint. **2.11 Recovery** — the governed restoration of architectural consistency following interruption, failure, degradation or conflict.

The five terms are canonical throughout KF-3005 and subsequent documents:

```
Lifecycle    defines the permitted journey
State        records the present condition
Transition   attempts governed change
Gate         decides whether change may proceed
Control      establishes whether required conditions hold
```

3. Abnormal Conditions

The architecture distinguishes six conditions that must never be silently collapsed into one another. Each has different evidence, ownership and disposition.

Condition	Meaning
Failure	A valid operation did not complete successfully.
Exception	A formally authorised and bounded departure from normal control behaviour that does **not** violate an invariant.
Violation	A breach of a Canonical Principle, invariant or mandatory constraint.
Degradation	Reduced ability while operating within explicitly governed limits.
Indeterminate condition	The architecture cannot establish whether a required condition holds.
Conflict	Authoritative records or service states disagree.

Two rules govern the boundary between them. First, **an exception cannot convert an invariant violation into conformance** — an authorised departure and a breach of an

invariant are different in kind, not degree. Second, **an indeterminate condition is never treated as satisfied**: it is surfaced, not resolved by default (see §30, §43).

4. Behavioural Grammar

The signature behavioural sequence of KF-3005 is centred on the **transition** — the transition, not the state, is the unit of governed change:

```
Trigger → Transition Request → Entity and Current-State Resolution
→ Transition Eligibility → Required Evidence Assembly
→ Authority and Permission Validation → Control Execution → Gate
Decision
→ Transition Execution → Authoritative State Update → Record
Persistence
→ Event Publication → Conformance Evidence
```

5. Behavioural Invariants

Every transition must preserve: (1) the affected entity is uniquely identified; (2) the authoritative current state is resolved; (3) the transition is legally permitted by the applicable state machine; (4) required evidence is available or explicitly indeterminate; (5) applicable permission and authority are valid; (6) gate controls are executed; (7) Gate Decision and Transition Records are bound to one transition identifier; (8) no completed transition is partially recorded; (9) prior history remains recoverable; (10) affected services receive the resulting event; (11) failures and indeterminate outcomes remain visible; (12) conformance evidence is produced; (13) no transition bypasses a protected invariant; (14) semantic reversal does not erase the original transition.

6. Behavioural Specification Types

KF-3005 defines four specification types: **State-Machine Specification** (permitted states and transitions for one entity class); **Transition Specification** (one governed state change); **Gate Specification** (the decision composition controlling one or more transitions); **Control Specification** (one reusable validation or enforcement mechanism).

Part II · Lifecycle Domain Architecture

7. No Single Universal State Machine

The architecture defines **no single universal state machine**. KF-3004 has already established that several state dimensions remain distinct, and KF-3005 preserves that separation: each governed condition belongs to its own state machine, and an entity may occupy states across several machines simultaneously. None of these states silently determines another. For one Operational Knowledge Object at one instant:

```
Object lifecycle state:   Operational
Version status:           Released
Epistemic standing:       Supported but contested
Actionable-form status:   Restricted
```

```
Operating condition:      Degraded
Conformance status:       Exception open
```

Each of the six values above is owned by a different authoritative service and changes only through its own governed transition.

8. Lifecycle Domains

KF-3005 defines several related but independent lifecycle domains, each with an authoritative state service drawn from the frozen KF-3003 catalogue.

Lifecycle domain	Governed entity	Authoritative state service
Object lifecycle	Operational Knowledge Object	SVC-08 · Lifecycle and Transition
Version lifecycle	Object Version	SVC-09 · Version and Lineage
Epistemic standing	Object version or claim set	SVC-05 · Justification
Acceptance lifecycle	Acceptance decision	SVC-06 · Authority and Acceptance
Release lifecycle	Released version	SVC-10 · Release
Actionable-form lifecycle	Actionable Form	SVC-12 · Operationalisation
Application lifecycle	Application instance	SVC-13 · Application and Outcome
Retirement lifecycle	Operational Knowledge Object	SVC-14 · Retirement (with SVC-08)
Operating condition	Service, control or transaction	SVC-17 · Operating Condition and Exception
Conformance standing	Architecture, service or capability	SVC-18 · Conformance and Audit

9. Cross-Domain Coordination

A transition in one domain may *trigger evaluation* in another but does not silently alter another domain's state. Acceptance may enable consideration of release but does not create release; release withdrawal may suspend actionable-form eligibility but does not alter epistemic standing; weakened epistemic standing may trigger review or suspension but does not itself execute either; retirement removes operational eligibility but preserves history and may require related release and form withdrawals; a degraded operating condition may block a transition without changing the object's epistemic state.

Part III · State Architecture

10. State-Machine Specification

Each state machine is defined through: identifier; governed entity; purpose; authoritative state record; authoritative service; permitted states; initial state; terminal states; permitted transitions; prohibited transitions; state-entry and state-exit conditions; cross-domain effects; historical-reconstruction rule; conformance test.

11. SM-OBJ · Operational Knowledge Object Lifecycle

Authoritative: SVC-08; record LIF-01.

States: Candidate → Registered → Structured → Under Evaluation → Justified → Accepted → Operational; with Suspended, Evolving, Retired and Historical.

Key transitions: Register, Structure, Begin Evaluation, Justify, Accept, Operationalise, Suspend, Evolve, Retire, Historicise, Reactivate.

Cross-domain: Operational status depends on an active release (SM-REL) but does not equal it.

Editorial note (retained): the final state list must be derived carefully from KF-2001 and KF-3004; some terms may ultimately be represented as **stage participation** rather than durable states. This machine must remain aligned with KF-2001 while distinguishing architectural state from method-stage activity.

12. SM-VER · Object Version State

Authoritative: SVC-09; record VER-01.

States: Draft → Established → Under Evaluation → Justified → Accepted → Released; with Superseded, Withdrawn, Historical.

A version may remain **Established but unreleased**. Creating a new version does not alter another version's release status unless a separate transition occurs.

13. SM-EPS · Epistemic Standing

Authoritative: SVC-05; record EPS-01.

A governed **standing model**, not a workflow. **Values:** Unsupported, Provisionally Supported, Supported, Strongly Supported, Weakened, Indeterminate. Contestation (EPS-03) and assessment progression (EVA-02) remain separate dimensions and are not encoded here.

14. SM-ACC · Acceptance State

Authoritative: SVC-06; record ACC-01.

States: Not Submitted, Pending, Accepted, Provisionally Accepted, Refused, Expired, Withdrawn, Superseded. Acceptance applies to a defined version, scope and effective period.

15. SM-REL · Release State

Authoritative: SVC-10; record REL-01 / REL-02.

States: Candidate, Pending, Active, Restricted, Suspended, Withdrawn, Expired, Superseded. Release state is independent of version state and object lifecycle state.

16. SM-ACT · Actionable Form State

Authoritative: SVC-12; record ACT-01 / ACT-04.

States: Draft, Under Review, Eligible, Restricted, Suspended, Withdrawn, Superseded, Historical.

17. SM-APP · Material Application State

Authoritative: SVC-13; record APP-01.

States: Requested, Eligibility Pending, Authorised, In Progress, Completed, Failed, Aborted, Reconciled.

18. SM-OPR · Operating Condition

Authoritative: SVC-17; record OPR-01.

States: Operational, Degraded, Unavailable, Pending, Failed, Indeterminate, Recovering, Recovered.

19. SM-CNF · Conformance Standing

Authoritative: SVC-18; record CNF-01 / CNF-05.

States: Not Assessed, Assessment Pending, Conformant, Conformant with Approved Exception, Indeterminate, Non-Conformant, Remediation in Progress, Remediated.

20. State-Assertion Rule *(applying KF-3004 §15.1)*

A **transition record preserves the historical act**; a **state record represents the effective current condition**; both are established consistently through **one** governed transition operation. A discrepancy between transition history and current state must produce an integrity or indeterminate condition (a Conflict, per §3). Consistent with §7, none of the state dimensions silently determines another: an object may simultaneously be Operational, Released, Supported-but-Contested, Restricted (form), Degraded (condition) and Exception-Open (conformance).

Part IV · Transition Architecture

21. Three Governed Catalogues

KF-3005 defines its permitted behaviour through **three distinct catalogues**, deliberately kept separate rather than flattened into one control list:

- the **Transition Catalogue** (§27) — the permitted governed changes;
- the **Gate Catalogue** (§31) — the decision compositions controlling those changes;
- the **Control Pattern Library** (Part VI) — the reusable checks and enforcement mechanisms invoked by gates and transitions.

Separation is what allows one control to be reused across many gates, and one gate across many transitions.

22. Transition Specification

Every governed transition is specified through: transition identifier; name; purpose; governed entity; source state; target state; permitted trigger; preconditions; required evidence; required authority class; required permissions; applicable gate; required controls; authoritative executing service; collaborating services; records created or changed; execution atomicity; events published; cross-domain effects; failure behaviour; recovery behaviour; conformance evidence.

23. Transition Request

Every request identifies: request identity; transition type; governed entity; source state claimed; target state requested; trigger; requesting actor or service; authority basis where applicable; supporting evidence; requested effective time; urgency; related exception; concurrency token or state reference. **A request does not authorise or execute the transition.**

24. Transition Eligibility

A transition is eligible for gate evaluation only where: the entity exists; the source state is authoritative and current; the transition is permitted by the state machine; required evidence can be resolved; required authority and permissions are identifiable; no unresolved state conflict prevents assessment; and the applicable operating-condition policy permits evaluation.

25. Transition Atomicity

A governed transition is architecturally **atomic**: it has one indivisible governed outcome — *Completed* or *Not Completed*. A completed transition establishes consistently: Gate Decision Record; Transition Record; resulting authoritative state; required status records; material provenance; event-publication record; conformance evidence.

A transition must never leave the architecture in a state where a gate says pass but the state did not change; the state changed but no transition record exists; a release was created but its acceptance reference is missing; or one service believes an object suspended while another treats it as active. A transition that cannot establish all required effects must **abort**, **remain**

pending, **enter recovery**, or complete under an explicitly defined **bounded-degradation** pattern. It must not appear completed while required records remain absent.

Atomicity does not mandate a single database transaction. Where technical atomicity is impossible, the architecture must define provisional state, recovery obligations, reconciliation, bounded duration and compensating controls.

26. Idempotency and Replay

A repeated request carrying the same transition identity must not produce duplicate transitions. Replay distinguishes continuation of an interrupted transition; retry after failure; deliberate repetition of a prior transition; and semantic re-execution under new evidence.

27. Transition Catalogue *(provisional)*

Domain-sensitive identifiers; representative, not exhaustive.

Transition	Source → Target	Applicable gate
TR-OBJ-01 Register	Candidate → Registered	GATE-REG-01
TR-OBJ-02 Structure	Registered → Structured	GATE-STR-01
TR-OBJ-03 Evaluate	Structured → Under Evaluation	GATE-EVA-01
TR-OBJ-04 Justify	Under Evaluation → Justified	GATE-JUS-01
TR-OBJ-05 Accept	Justified → Accepted	GATE-ACC-01
TR-OBJ-06 Operationalise	Accepted → Operational	GATE-REL-01, GATE-ACT-01
TR-OBJ-07 Suspend	Operational → Suspended	GATE-SUS-01
TR-OBJ-08 Evolve	Operational / Suspended → Evolving	GATE-EVO-01
TR-OBJ-09 Retire	any active → Retired	GATE-RET-01
TR-OBJ-10 Reactivate	Suspended / Retired → Evolving	GATE-REA-01
TR-VER-01 Establish	Draft → Established	GATE-STR-01
TR-VER-02 Release version	Accepted → Released	GATE-REL-01
TR-VER-03 Supersede	Released → Superseded	GATE-EVO-01
TR-ACC-01 Accept	Pending → Accepted	GATE-ACC-01
TR-ACC-02 Withdraw acceptance	Accepted → Withdrawn	GATE-SUS-01
TR-REL-01 Release	Candidate → Active	GATE-REL-01
TR-REL-02 Suspend release	Active → Suspended	GATE-SUS-01
TR-REL-03 Withdraw release	Active → Withdrawn	GATE-SUS-01
TR-ACT-01 Make form eligible	Under Review → Eligible	GATE-ACT-01
TR-APP-01 Authorise	Eligibility Pending →	GATE-APP-01

Transition	Source → Target	Applicable gate
application	Authorised	
TR-EPS-01 Update standing	any → any (evidence-driven)	— (control CTRL-EPS-01)
TR-OPR-01 Recover	Degraded / Failed → Recovered	GATE-REC-01
TR-RET-01 Retire	Operational → Retired	GATE-RET-01

Part V · Gate Architecture

28. Gate Specification

Each gate is specified through: identifier; name; purpose; applicable transitions; required entry state; required information objects; required authority; required controls; decision composition; permitted decision outcomes; blocking conditions; exception rules; records produced; events published; conformance evidence.

Assurance-class parameterisation. A gate's exit criteria, the depth of evidence it requires, and the review or authority it demands are applied to the depth the object's **assurance class** requires (KF-2003), resolved at the release / scope-of-reliance to which the class attaches and enforced by KF-3001 IC-23. A higher class raises required rigour; it does not add new gates. The gate's protection of the six ungraded lifecycle invariants is identical at every class — only the depth above that floor scales.

29. Gates Orchestrate Controls

A gate is **not** one large control. A gate *orchestrates* several reusable controls, each of which establishes whether one required condition holds. This is what permits the same control to be reused across several gates. The Release Gate illustrates the pattern:

```
GATE-REL-01 · Release Gate
     ├── State Eligibility Control          (CTRL-STA-01)
     ├── Acceptance Validity Control        (CTRL-ACC-01)
     ├── Authority Validation Control       (CTRL-AUT-01)
     ├── Dependency Clearance Control       (CTRL-DEP-01)
     ├── Governance Standing Control        (CTRL-STW-01 / CTRL-REV-01)
     ├── Operating Condition Control        (CTRL-OPR-01)
     ├── Exception Control                  (CTRL-EXC-01)
     └── Provenance Completeness Control    (CTRL-PRO-01)
```

The gate composes the *results* of these controls into one decision; it does not re-implement them. The full gate-to-control mapping is given in §56.

30. Standard Gate Outcomes

Every gate supports, where applicable: **Pass, Fail, Return, Pending, Indeterminate, Pass with Approved Restriction.** An outcome of **Indeterminate is never silently treated as Pass.**

31. Gate Catalogue *(provisional)*

Gate	Determines
GATE-REG-01 · Registration	whether candidate intake may be registered as a new object or routed to an existing identity, version or lineage path
GATE-STR-01 · Structure	whether the representation is inspectable and complete enough for evaluation
GATE-EVA-01 · Evaluation	whether the object or evidence meets the threshold to proceed toward justification
GATE-JUS-01 · Justification	whether the recorded warrant is sufficiently complete and inspectable
GATE-ACC-01 · Acceptance	whether an authorised authority may permit reliance on a justified version
GATE-REL-01 · Release	whether an accepted version may become operationally eligible within a defined scope
GATE-ACT-01 · Actionable-Form	whether a derived form may become eligible for governed application
GATE-APP-01 · Application Eligibility	whether a specific form may be used in a specific context at a specific time
GATE-REV-01 · Continuing Review	whether governance standing remains valid
GATE-EVO-01 · Evolution	whether a changed version has satisfied re-evaluation, re-justification, dependency disposition and renewed acceptance
GATE-SUS-01 · Suspension	whether operational eligibility must be restricted or suspended
GATE-RET-01 · Retirement	whether the object may leave active operational use
GATE-REA-01 · Reactivation	whether a retired or suspended object may re-enter governed evolution
GATE-REC-01 · Recovery	whether an interrupted or degraded operation may return to normal execution

Part VI · Control Pattern Library

32. Control Specification

Each control is specified through: identifier; name; type; purpose; input condition; information required; executing service; authority dependency; evaluation logic; possible results; failure behaviour; evidence produced; protected invariants; enforced interaction constraints; reused by gates; conformance test.

Class-graded strength. Where a control's evaluation depends on a threshold — evidence sufficiency, review independence, monitoring frequency — that threshold scales with the object's assurance class (KF-2003) under IC-23. A control that protects an invariant enforces it at every class without relaxation; a control that discharges a graded obligation applies the minimum for the object's class. The same control is thereby reused across classes at different depths, just as it is reused across gates (§29).

33. Control Types

Preventive (blocks an invalid action); **Detective** (identifies an invalid, missing or degraded condition); **Corrective** (returns the architecture to a governed condition); **Evidentiary** (produces inspectable proof); **Historical** (preserves prior states and reconstructability); **Directive** (defines required behaviour or routing).

34. Control Families

Family A · Identity and State — CTRL-ID-01 Object Identity Resolution; CTRL-ID-02 Duplicate and Prior-Object Detection; CTRL-STA-01 Authoritative Current-State Validation; CTRL-STA-02 Transition Legality Validation; CTRL-STA-03 State-Conflict Detection; CTRL-VER-01 Version Immutability; CTRL-LIN-01 Lineage Integrity.

Family B · Evidence and Warrant — CTRL-EVD-01 Required Evidence Presence; CTRL-EVD-02 Evidence Availability; CTRL-EVD-03 Counter-Evidence Visibility; CTRL-EVA-01 Evaluation Completion; CTRL-JUS-01 Justification Completeness; CTRL-EPS-01 Epistemic Standing Validation; CTRL-WAR-01 Warrant-Link Preservation.

Family C · Authority and Governance — CTRL-AUT-01 Authority Mandate Validation; CTRL-AUT-02 Delegation Validity; CTRL-PRM-01 Permission Validation; CTRL-ACC-01 Acceptance Validity; CTRL-STW-01 Stewardship Validation; CTRL-REV-01 Review Currency; CTRL-SEG-01 Separation-of-Duties Validation; CTRL-GOV-01 Governance-Exception Validation.

Family D · Release and Operational — CTRL-REL-01 Release Eligibility; CTRL-DEP-01 Dependency Impact Clearance; CTRL-ACT-01 Actionable-Form Eligibility; CTRL-APP-01 Application Eligibility; CTRL-CTX-01 Use-Context Compatibility; CTRL-EXP-01 Explainability Requirement; CTRL-OUT-01 Outcome Capture Requirement.

Family E · Integrity, Condition and Assurance — CTRL-PRO-01 Provenance Completeness; CTRL-INT-01 Record Integrity; CTRL-TMP-01 Effective-Time Validation; CTRL-OPR-01 Operating-Condition Validation; CTRL-EXC-01 Blocking-Exception Check; CTRL-ATM-01 Transition Atomicity; CTRL-EVT-01 Event Publication; CTRL-CNF-01 Conformance-Evidence Recording; CTRL-AUD-01 Independent-Audit Visibility.

Part VII · Transition Pattern Library

35. Standard Transition Pattern

```
Request → Resolve Entity and Current State → Validate Transition
Legality
→ Resolve Required Evidence → Validate Permission and Authority
```

```
→ Validate Dependencies and Operating Condition → Execute Gate
Controls
→ Record Gate Decision → Execute Transition → Persist State and
Transition Records
→ Publish Events → Record Conformance Evidence
```

The final step, **Record Conformance Evidence**, is the *emission* of evidence the preceding steps already produced — the gate decision, the transition record, the published events — not the separate authoring of an evidence artefact. The governed act is its own evidence (KF-3001 IC-24; KF-3004 §5.2). A distinct artefact is added only where the act's own records cannot carry the required proof.

36. Specialised Patterns

Creation (creates a governed entity where none existed); **Promotion** (moves an entity into a more operationally eligible state after satisfying additional obligations — candidate→evaluated, accepted→released, provisional form→eligible); **Restriction** (reduces permitted reliance without removing all eligibility); **Suspension** (temporarily removes eligibility pending review, recovery or correction); **Withdrawal** (ends a release, acceptance or form eligibility without retiring the object); **Supersession** (replaces a version, release, representation or form with a successor, preserving lineage); **Correction** (creates a governed correction without erasing the original); **Retirement** (ends active eligibility of the enduring object while preserving history); **Reactivation** (routes a retired or suspended entity through renewed evolution before eligibility is restored); **Recovery** (restores a failed, interrupted or degraded operation to a consistent governed condition).

Part VIII · Composite Decision Architecture

37. Composite Decisions

A gate decision may depend on several independently authoritative conditions. **The gate records references to those authoritative records rather than reproducing them** (KF-3004 §7.2), so no gate becomes a competing source of truth.

38. Release Decision Composition (GATE-REL-01)

```
Eligible Object Version + Valid Justification + Applicable Epistemic
Standing
+ Valid Acceptance + Applicable Authority + Assigned Steward
+ Dependency Clearance + Valid Operating Condition + No Blocking
Exception
+ Required Provenance → Release Gate Decision
```

39. Application Eligibility Composition (GATE-APP-01)

```
Active Object + Eligible Actionable Form + Valid Release + Compatible
Use Context
+ Current Governance Standing + Applicable Permission + Required Human
Oversight
```

```
+ Acceptable Operating Condition + No Blocking Restriction →
Application Eligibility Decision
```

40. Retirement Decision Composition (GATE-RET-01)

```
Valid Retirement Reason + Applicable Authority + Dependency
Disposition
+ Successor or Supersession Handling + Form and Release Withdrawal
+ Preservation Plan + Historical Continuity → Retirement Gate Decision
```

Part IX · Exception and Degraded-State Architecture

41. Exception Classes

Operational Exception (a bounded departure affecting execution without changing architectural obligations); **Governance Exception** (a formally authorised, time-limited departure from a governance practice where no invariant is violated); **Architectural Variation** (a conforming alternative design); **Architectural Violation** (a breach of a mandatory invariant or constraint); **Indeterminate Condition** (validity cannot currently be established). Consistent with §3, an exception cannot convert an invariant violation into conformance.

42. Exception Specification

Each exception records: identity; affected transition / gate / control; scope; reason; authority; effective period; compensating controls; prohibited consequences; monitoring; expiry; review; closure; conformance effect.

43. Fail-Open and Fail-Closed Behaviour

Every control and gate declares whether failure or indeterminate standing causes fail-closed, bounded degraded operation, manual review, temporary restriction, retry, suspension or abort. **Fail-open behaviour requires explicit justification and bounded scope.** The default for governance-critical operations is fail-closed.

Part X · Recovery and Reconciliation Architecture

44. Recovery Is Forward Restoration, Not Historical Erasure

Recovery restores permitted operation; it does not rewrite the past. In a governed architecture, “rollback” must not erase the attempted transition, the gate decision, the failed action, intermediate records or recovery activity. A **semantic reversal adds to history** — it is a new governed transition returning an entity to an earlier operational condition, not a deletion of the original act. Accordingly the architecture separates technical rollback (transactional consistency) from semantic reversal (a new restoring transition), and both from reinstatement (restoration of eligibility after revalidation), replay (re-execution of an interrupted or deferred

operation) and reconciliation (detection and correction of inconsistency between services or records).

45. Recovery Modes

Abort (terminate without changing governed state); **Retry** (repeat an idempotent operation); **Replay** (continue or re-execute from governed event history); **Technical Rollback** (restore transactional consistency without erasing evidence of the attempt); **Semantic Reversal** (a new transition restoring an earlier operational condition — it *adds* to history); **Reconciliation** (compare authoritative records and repair inconsistency); **Revalidation** (reassess whether prior eligibility remains valid); **Reinstatement** (restore eligibility after successful revalidation and gate approval).

46. Recovery Requirements

Recovery must preserve the failed or interrupted attempt; identify the last consistent state; prevent duplicate execution; reconcile authoritative services; preserve temporal order; record compensating action; validate post-recovery invariants; publish recovery events; and produce conformance evidence.

Part XI · Concurrency and Conflict Architecture

47. Concurrency Principles

(1) A transition evaluates authoritative state immediately before execution. (2) A stale source-state reference invalidates the request. (3) Conflicting transitions are serialised, rejected or explicitly reconciled. (4) A future-effective decision is reconsidered where a blocking event occurs before effect. (5) No service assumes a previously retrieved eligibility condition remains valid indefinitely. (6) Cross-service state changes are reconciled through authoritative events and transition identifiers. The architecture requires transition identifiers, state-version or concurrency tokens, effective-time checks, immediate pre-execution revalidation, conflict detection, explicit precedence and idempotent retry — without prescribing a locking technology.

48. Conflict Types

Concurrent version changes; acceptance withdrawal during release; release withdrawal during application; retirement during pending evolution; conflicting authority decisions; stewardship transfer during approval; retrospective correction affecting a completed decision; dependency discovery after release; service-state disagreement; event-ordering ambiguity.

49. Conflict Disposition

Reject the later request; prioritise mandatory suspension; queue for reassessment; merge non-conflicting changes; create a branch; suspend operational use; escalate to accountable authority; mark indeterminate; or reconcile after recovery.

Part XII · Control and Transition Conformance

50. Transition Conformance

Source and target states permitted; required controls executed; applicable gate produced a valid decision; authority and permission valid; records complete and consistent; atomicity preserved; events published; history recoverable; no invariant violated; conformance evidence independently inspectable.

51. Gate Conformance

Required conditions explicit; controls identified; evidence authoritative and current; decision outcomes distinguishable; indeterminate conditions visible; exception handling bounded; Gate Decision Records complete; decision composition reconstructable.

52. Control Conformance

Purpose and scope defined; required inputs available; evaluation logic inspectable; outcomes recorded; failure behaviour defined; protected invariants identifiable; evidence independently testable.

53. Behavioural Reconstruction Test

For any material transition, the architecture must reconstruct:

```
Trigger → Transition Request → Source State → Evidence and Authority
→ Controls Executed → Gate Decision → Transition Execution → Resulting
State
→ Records Persisted → Events Published → Conformance Evidence
```

Part XIII · Crosswalks

54. State-Machine-to-Service Crosswalk

State machine	Authoritative service	Principal KF-3004 records
SM-OBJ	SVC-08	LIF-01, EVT-01, GAT-01, TRN-01
SM-VER	SVC-09	VER-01, CHG-01, LIN-01, COR-01
SM-EPS	SVC-05	EPS-01, EPS-02, EPS-03
SM-ACC	SVC-06	ACC-01, ACC-02, ACC-03, AUT-01
SM-REL	SVC-10	REL-01, REL-02
SM-ACT	SVC-12	ACT-01, ACT-03, ACT-04
SM-APP	SVC-13	APP-01, APP-02, OUT-01

State machine	Authoritative service	Principal KF-3004 records
SM-OPR	SVC-17	OPR-01, OPR-02, EXC-01
SM-CNF	SVC-18	CNF-01, CNF-02, CNF-05
SM-RET	SVC-14 (with SVC-08)	RET-01, PRE-01, HIS-01

55. Transition-to-Gate Crosswalk

Transition group	Applicable gate(s)
Registration (TR-OBJ-01)	GATE-REG-01
Structuring (TR-OBJ-02, TR-VER-01)	GATE-STR-01
Evaluation (TR-OBJ-03)	GATE-EVA-01
Justification (TR-OBJ-04)	GATE-JUS-01
Acceptance (TR-OBJ-05, TR-ACC-01)	GATE-ACC-01
Release / operationalisation (TR-OBJ-06, TR-VER-02, TR-REL-01)	GATE-REL-01, GATE-ACT-01
Application (TR-APP-01)	GATE-APP-01
Continuing review	GATE-REV-01
Evolution / supersession (TR-OBJ-08, TR-VER-03)	GATE-EVO-01
Suspension / withdrawal (TR-OBJ-07, TR-ACC-02, TR-REL-02/03)	GATE-SUS-01
Retirement (TR-OBJ-09, TR-RET-01)	GATE-RET-01
Reactivation (TR-OBJ-10)	GATE-REA-01
Recovery (TR-OPR-01)	GATE-REC-01

56. Gate-to-Control Crosswalk

Gate	Required controls
GATE-REG-01	CTRL-ID-01, CTRL-ID-02, CTRL-STA-02, CTRL-PRO-01, CTRL-CNF-01
GATE-STR-01	CTRL-STA-01, CTRL-STA-02, CTRL-VER-01, CTRL-CNF-01
GATE-EVA-01	CTRL-EVD-01, CTRL-EVD-02, CTRL-EVA-01, CTRL-OPR-01, CTRL-CNF-01
GATE-JUS-01	CTRL-JUS-01, CTRL-EVD-03, CTRL-EPS-01, CTRL-WAR-01, CTRL-CNF-01
GATE-ACC-01	CTRL-STA-01, CTRL-JUS-01, CTRL-EPS-01, CTRL-AUT-01, CTRL-PRM-01, CTRL-SEG-01, CTRL-OPR-01, CTRL-CNF-01
GATE-REL-01	CTRL-STA-01, CTRL-ACC-01, CTRL-AUT-01, CTRL-DEP-01, CTRL-STW-01, CTRL-REV-01,

Gate	Required controls
	CTRL-OPR-01, CTRL-EXC-01, CTRL-PRO-01, CTRL-ATM-01, CTRL-CNF-01
GATE-ACT-01	CTRL-WAR-01, CTRL-REL-01, CTRL-ACT-01, CTRL-OPR-01, CTRL-CNF-01
GATE-APP-01	CTRL-STA-01, CTRL-ACT-01, CTRL-REL-01, CTRL-CTX-01, CTRL-REV-01, CTRL-PRM-01, CTRL-EXP-01, CTRL-OPR-01, CTRL-EXC-01, CTRL-OUT-01, CTRL-CNF-01
GATE-REV-01	CTRL-REV-01, CTRL-STW-01, CTRL-GOV-01, CTRL-OPR-01, CTRL-CNF-01
GATE-EVO-01	CTRL-STA-02, CTRL-EVA-01, CTRL-JUS-01, CTRL-DEP-01, CTRL-ACC-01, CTRL-LIN-01, CTRL-CNF-01
GATE-SUS-01	CTRL-STA-01, CTRL-AUT-01, CTRL-DEP-01, CTRL-EXC-01, CTRL-CNF-01
GATE-RET-01	CTRL-AUT-01, CTRL-DEP-01, CTRL-LIN-01, CTRL-PRO-01, CTRL-INT-01, CTRL-CNF-01
GATE-REA-01	CTRL-STA-02, CTRL-EVA-01, CTRL-JUS-01, CTRL-ACC-01, CTRL-AUT-01, CTRL-CNF-01
GATE-REC-01	CTRL-ATM-01, CTRL-STA-03, CTRL-INT-01, CTRL-OPR-01, CTRL-CNF-01

57. Control-to-Invariant Crosswalk

Control	Protected invariant(s)
CTRL-ID-01, CTRL-ID-02	INV-01
CTRL-STA-01	INV-03, INV-14
CTRL-STA-02	INV-08
CTRL-STA-03	INV-24
CTRL-VER-01	INV-12
CTRL-LIN-01	INV-13
CTRL-EVD-01	INV-05
CTRL-EVD-02	INV-04
CTRL-EVD-03	INV-06
CTRL-EVA-01	INV-24
CTRL-JUS-01	INV-05
CTRL-EPS-01	INV-07
CTRL-WAR-01	INV-17
CTRL-AUT-01, CTRL-AUT-02	INV-09
CTRL-PRM-01	INV-11

Control	Protected invariant(s)
CTRL-ACC-01	INV-05
CTRL-STW-01, CTRL-REV-01, CTRL-GOV-01	INV-10
CTRL-SEG-01	INV-09
CTRL-REL-01, CTRL-ACT-01	INV-17
CTRL-DEP-01	INV-15
CTRL-APP-01, CTRL-CTX-01, CTRL-EXC-01	INV-18
CTRL-EXP-01	INV-20
CTRL-OUT-01, CTRL-EVT-01	INV-19
CTRL-PRO-01	INV-04
CTRL-INT-01	INV-11
CTRL-TMP-01	INV-14
CTRL-OPR-01	INV-24
CTRL-ATM-01	INV-08, INV-14
CTRL-CNF-01	INV-21
CTRL-AUD-01	INV-22, INV-23

58. Control-to-Interaction-Constraint Crosswalk

Constraint	Enforcing control(s)
IC-01	CTRL-STA-01, CTRL-STA-02
IC-02	CTRL-ATM-01 (gate–transition binding)
IC-03	CTRL-EXC-01 (suspension ownership)
IC-04	GATE-REA-01 controls
IC-05	CTRL-ID-01
IC-06	CTRL-VER-01
IC-07	CTRL-LIN-01
IC-08	CTRL-REL-01, CTRL-ACC-01
IC-09	CTRL-JUS-01, CTRL-ACC-01
IC-10	CTRL-WAR-01
IC-11	CTRL-APP-01
IC-12	CTRL-WAR-01
IC-13	CTRL-AUT-01, CTRL-PRM-01
IC-14	CTRL-INT-01
IC-15	CTRL-INT-01 (administrative action)
IC-16	CTRL-SEG-01, CTRL-AUT-01

Constraint	Enforcing control(s)
IC-17	CTRL-AUD-01
IC-18	CTRL-AUD-01, CTRL-CNF-01
IC-19	CTRL-EXC-01
IC-20	CTRL-PRO-01
IC-21	CTRL-INT-01 (append-only correction)
IC-22	CTRL-TMP-01

59. Gate-to-Information Crosswalk

Gate	Required KF-3004 records
GATE-REG-01	OBJ-01, OBJ-02, OBJ-03, PRO-01
GATE-STR-01	REP-01, VER-01
GATE-EVA-01	EVA-01, EVA-02, EVA-03, EVD-01
GATE-JUS-01	JUS-01, EPS-01, EVD-03
GATE-ACC-01	ACC-01, AUT-01, JUS-01, EPS-01
GATE-REL-01	REL-01, ACC-01, VER-01, DEP-03, STW-01, OPR-01, PRO-01
GATE-ACT-01	ACT-01, ACT-03, REL-01
GATE-APP-01	APP-01, APP-02, APP-03, ACT-01, REL-01
GATE-REV-01	REV-02, STW-01, CMP-01, GOV-01
GATE-EVO-01	CHG-01, EVA-03, JUS-01, DEP-02, ACC-01, LIN-01
GATE-SUS-01	TRN-01, REL-02, EXC-01
GATE-RET-01	RET-01, PRE-01, LIN-01, DEP-03
GATE-REA-01	REA-01, CHG-01, ACC-01
GATE-REC-01	REC-01, OPR-01, INT-01

60. Transition-to-Authority Crosswalk *(abstract classes for KF-3008)*

Transition group	Required authority class	Separation condition
Registration	Registration Authority	—
Acceptance	Acceptance Authority (human)	distinct from justifying actor
Release	Release Authority	distinct from acceptance where profile requires
Application (material)	Accountable Application Authority	human accountable participant
Suspension	Suspension Authority	may be raised by any service,

Transition group	Required authority class	Separation condition
		decided by authority
Evolution / reactivation	Governance Authority	re-acceptance by acceptance authority
Retirement	Retirement Authority	distinct from steward where profile requires
Exception approval	Exception Authority	distinct from requesting actor

61. Transition-to-Conformance-Evidence Crosswalk *(for KF-3007)*

Every material transition produces, at minimum: Gate Decision Record (GAT-01); Transition Record (TRN-01); the resulting authoritative state record; provenance (PRO-01); an event-publication record; and conformance evidence (CNF-01 / CNF-02). KF-3007 defines how this evidence is sampled, tested and judged.

62. Lifecycle-to-Capability Crosswalk

Behaviour	KF-3002 capability
SM-OBJ transitions, gates, atomicity	C-CAP-08
SM-VER, evolution, lineage	C-CAP-04, C-CAP-06
SM-EPS	B-CAP-04
SM-ACC	C-CAP-01
SM-REL	C-CAP-05
Dependency clearance	C-CAP-07
SM-ACT	D-CAP-01
SM-APP, outcome capture	D-CAP-02, D-CAP-03
Retirement	D-CAP-04
Provenance, operating condition, conformance controls	E-CAP-01, E-CAP-04, E-CAP-02
Access and authority controls	E-CAP-03

63. Lifecycle-to-Service Crosswalk

Behaviour	KF-3003 service
Object state and transition execution	SVC-08
Version and lineage	SVC-09
Epistemic standing	SVC-05
Acceptance and authority validation	SVC-06
Release	SVC-10
Dependency clearance	SVC-11
Actionable-form eligibility	SVC-12
Application and outcome	SVC-13

Behaviour	KF-3003 service
Retirement and preservation	SVC-14
Provenance controls	SVC-15
Access, permission, integrity controls	SVC-16
Operating-condition and exception controls	SVC-17
Conformance-evidence and audit controls	SVC-18
Temporal and effective-time controls	SVC-19

Part XIV · Relationship to Subsequent Documents

KF-3008 defines the authority and accountability model — authority sources, delegation, accountable-body composition, separation of duties, mandate validity and role assignment — satisfying the abstract authority classes KF-3005 identifies in the Transition-to-Authority Crosswalk (§60). KF-3005 references authority only in abstract form (Required Authority Class, Required Decision Role, Required Separation Condition); KF-3008 defines who may occupy those roles.

KF-3007 defines how the evidence produced by state changes, gates and controls (§61) is assessed — sampling, assessment procedure, assurance independence, severity and conformance judgement. KF-3005 defines *what evidence must exist*; KF-3007 defines *how it is tested*.

KF-3009 maps every state, transition, gate and control back to the Canon, lifecycle, practice, capability, service and information obligations from which it derives.

KF-4000 may implement this behavioural architecture through state machines, workflow engines, rules engines, transaction managers, event-driven architectures, human approval processes, AI-agent orchestration, manually maintained registers, or combinations. It may not bypass a required gate; collapse gate decision and transition execution; treat indeterminate evidence as satisfied; permit silent state mutation; erase failed transitions or recovery history; infer authority from permission; release or apply ineligible knowledge; create conflicting authoritative states; make transition history unreconstructable; or substitute automation for accountability.

Closing Statement

KF-3005 defines the governed behaviour of the Knowledge Foundry architecture. It establishes how state may change, how transitions are requested and executed, how gates compose authoritative decisions from independently owned records, how controls preserve invariants, how failures and exceptions remain visible, and how every material change leaves sufficient evidence for historical reconstruction and assurance. KF-3004 defines the architecture's memory; KF-3005 defines the disciplined behaviour through which that memory is allowed to change.

KF-3006 · Architectural Integration Model

How the architectural elements collaborate to produce, govern, evolve and retire an Operational Knowledge Object

Document Class: Reference Architecture · Parent Programme: Knowledge Foundry · Parent Document: KF-3000 · Status: Draft · Version: 0.1

```
Principle (KF-1000) → Lifecycle (KF-2001) → Practices (KF-2002)
        → Reference Architecture (KF-3000)
              → Domains (KF-3001 principles · KF-3002 capabilities ·
KF-3003 services · KF-3004 records · KF-3005 control)
              → ▸ Architectural Integration Model (KF-3006)
              → Architectural Conformance Framework (KF-3007)
        → Implementation (KF-4000)
```

The Knowledge Foundry Reference Architecture is intentionally modular. Each document defines one architectural concern in isolation, so that responsibilities stay clear and the architecture can evolve without unnecessary coupling. Integration therefore cannot be inferred from any single document: it is itself an architectural concern, and must be described as one. KF-3006 provides that description.

Purpose

The documents before this one describe the architecture **part by part**: what must remain true (KF-3001), what it must be able to do (KF-3002), where responsibility sits (KF-3003), what information exists (KF-3004), and how governed change is permitted and executed (KF-3005). Each is complete in its domain and silent on the others.

KF-3006 describes the architecture **as one working system**. It answers a single question:

> *How do all architectural elements collaborate to produce a governed Operational Knowledge Object?*

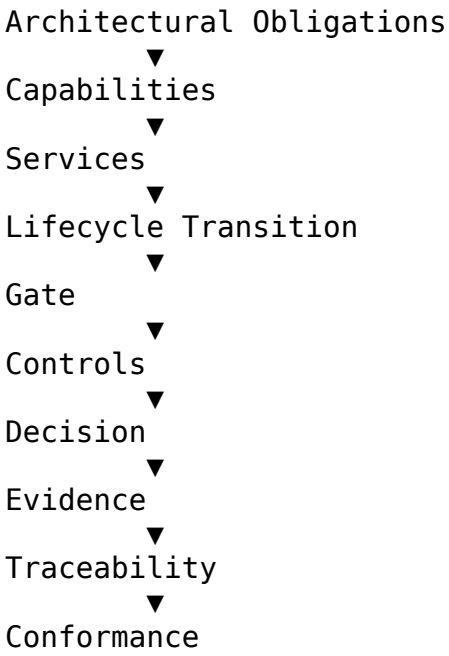
It is the point in the KF-3000 series where the reader should feel the machine come alive: capabilities are realised by services, services are invoked by transitions, transitions pass gates, gates orchestrate controls, controls protect obligations, obligations emit evidence, evidence accumulates as traceability, and traceability is what conformance later inspects. None of that collaboration is visible from any single domain document. This one makes it visible.

KF-3006 **summarises, never redefines.** Where it names a service, a gate, a control, an invariant or a record, the authoritative definition remains in its home document; this model shows only how the element participates in the whole. Its value is integrative, not additive.

Figure 1 · The Integration Spine

One flow carries the whole architecture, from the obligations the Canon imposes to the conformance that inspects the result:

```
Canonical Principles
        ▼
```



Every element in the KF-3000 series occupies one band of this spine, and every chapter of this document explains one of its arrows. A reader who holds this figure in mind can place any architectural element by asking only where on the spine it acts.

Scope and the Integration Admission Test

KF-3006 is technology-neutral and deployment-neutral. It describes collaboration between architectural elements, not the components, products or topologies that realise them (those are KF-4000). A statement belongs in KF-3006 only if:

1. it describes an interaction between two or more architectural elements defined elsewhere in the series;
2. it can be stated without naming a specific technology, product or deployment; and
3. it preserves — never relaxes — the lifecycle invariants and interaction constraints of KF-2001 and KF-3001.

A statement that redefines an element belongs in that element's home document; a statement that names a technology belongs in KF-4000.

---

1 · Introduction

1.1 Purpose

To demonstrate that the Knowledge Foundry architecture functions as a coherent system rather than a collection of independent components, by describing the relationships and behaviours through which its elements collaborate over the life of an Operational Knowledge Object (OKO).

1.2 Scope

In scope: the relationship types between architectural elements; the behaviour of the architecture over time; the integrated journey of one OKO; the recurring integration patterns; the architectural event catalogue; and the way evidence, traceability and assurance are produced as by-products of ordinary architectural work.

Out of scope: redefinition of any element; component design; technology selection; workflow, cadence and staffing (KF-4000); and the *judgement* of whether an operation conforms (KF-3007).

1.3 Relationship to Other Documents

```
Canon (KF-1000)
     ↓ obligations
Methodology (KF-2000 · lifecycle, practices, materiality)
     ↓ methods
Reference Architecture (KF-3000)
     ↓ domains (KF-3001–KF-3005)
Architectural Integration Model (KF-3006)   ← how the architecture
operates
     ↓
Architectural Conformance Framework (KF-3007)  ← whether it conforms
     ↓
Implementation (KF-4000)
```

KF-3006 depends on all five domain documents and on KF-2003 (assurance). It is depended upon by KF-3007, which inspects the traceability this model shows the architecture producing.

---

2 · Principles of Architectural Integration

These are the philosophy of integration, not its mechanics. Each is inherited from a higher level and constrains how the elements may be wired together.

1. **Integration follows the Canon.** Elements may be combined only in ways consistent with the Canonical Principles. Integration is not free wiring; it is constrained composition.
2. **Every interaction preserves the lifecycle invariants.** No collaboration between elements may break single identity, attributable transitions, justified transitions, inspectable transitions, provenance or preserved lineage. The invariants are the floor beneath every interaction (KF-2001; KF-3001).
3. **Integration is event-driven, not document-driven.** Elements collaborate by raising and responding to events that trigger governed transitions, not by passing documents hand to hand. What moves through the system is a governed state change, not a file (KF-2001 event model).
4. **Evidence is a by-product of behaviour.** Conformance evidence is emitted by the governed act, not authored by a separate administrative step (KF-3001 IC-24; KF-3004 §5.2). Integration is designed so that doing the work *is* recording the work.

5. **Integration preserves traceability.** Every interaction leaves the relationships, events and evidence needed to reconstruct what happened, to the depth the object's assurance class requires (KF-2003; KF-3007 Test 4).
6. **Responsibilities remain loosely coupled.** Each capability has one accountable service and each record family one authoritative maintainer; services collaborate through governed interactions across named boundaries, never by reaching into one another's records (KF-3003 SP-2/SP-3; KF-3004 §7). Loose coupling is what lets an implementation substitute one service without disturbing the others.

---

3 · Participating Architectural Elements

The elements that take part in integration, each summarised for use here; the citation is the authority.

Element	In one line	Home
Operational Knowledge Object (OKO)	the governed unit: representation + evidence + justification + acceptance-state, under one identity	KF-1101
Lifecycle	the staged progression Capture → … → Retire through which an OKO evolves	KF-2001
Capability	an enduring ability the architecture must possess	KF-3002
Service	a logical mechanism that realises capabilities and maintains records	KF-3003
Information / Record object	a governed unit of information with one authoritative maintainer	KF-3004
Actor	a person or agent that raises events and performs work	KF-2001
Role	a bundle of responsibilities an actor holds	KF-2001
Authority	the human right to certify, accept, release, steward or retire	KF-1103; KF-3006 §5.3
Gate	a justification checkpoint that decides whether an OKO may advance	KF-2001; KF-3005
Control	a reusable test of one required condition	KF-3005

Element	In one line	Home
Invariant	a property that must hold at every instant	KF-2001; KF-3001
Evidence	the durable record that an obligation was discharged, emitted by the act	KF-3004 §5.2
Traceability	the reconstructable web of relationships, events and evidence	KF-3006 §10
Assurance class	the grade that sets how deep the above must go	KF-2003

4 · Architectural Relationships

This chapter replaces the hundreds of arrows scattered across the domain documents with a **controlled vocabulary of relationship types**. Every legitimate interaction in the architecture is an instance of one of these types. (These are *element-level* integration relationships; the *record-level* relationship types — VERSION_OF, RELEASES, SUPERSEDES and the rest — remain KF-3004's.)

4.1 The Canonical Relationship Vocabulary

Every legitimate interaction is an instance of one of the nine canonical relationships below. Each belongs to a category that fixes what *kind* of relationship it is — so that "Service X invokes Y" and "Service X consumes Y" are never read as the same claim.

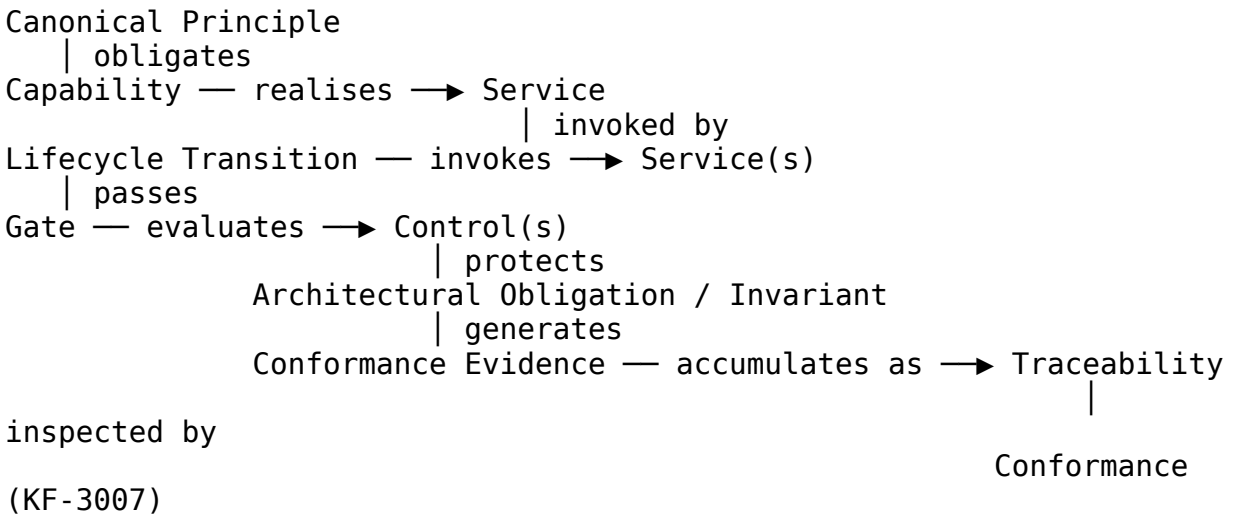
Relationship	Category	Reads as	Governing rule
constrains	normative	Principle / interaction constraint → element	composition must remain consistent with the Canon and the interaction constraints (KF-1000; KF-3001)
realises	implementation	Capability → Service	each capability has exactly one *accountable* service; supporting services may assist (KF-3003 SP-2)
invokes	behavioural	Transition → Service(s)	each transition invokes one or more services; no transition is serviceless (KF-2001)

Relationship	Category	Reads as	Governing rule
consumes	information	Service → Information object	a service reads an authoritative record through a governed interaction, never by overwriting it (KF-3004 §7)
produces	information	Service / Transition → Information object	the authoritative maintainer produces and maintains its records (KF-3004 §7)
evaluates	governance	Gate → Control(s)	a gate orchestrates several reusable controls; it does not re-implement them (KF-3005 §29)
protects	assurance	Control → Obligation / Invariant	a control protects at least one explicit obligation, risk, policy or invariant (KF-3005 §32)
generates	evidential	Transition / Obligation → Conformance Evidence	evidence is emitted by the act — event-native and proportionate — not authored separately (KF-3001 IC-24; KF-3007 Test 3)
traces	provenance	derived state → source; successor → predecessor	provenance and lineage links are carried through and preserved, never overwritten (KF-2001 invariants 5–6)

Three further structural relationships support composite decisions and grading and are used throughout the model: **references** — a composite decision cites the authoritative records it relied on without copying their content (KF-3004 §7.2); **attributed-to** — every transition resolves to an identified actor or human authority (KF-2001 invariant 2); and **graded-by** — the depth to which a graded obligation is discharged is set by the object's assurance class (KF-2003; KF-3001 IC-23).

4.2 The Core Relationship Chain

Read end to end, twelve relationship types compose into one chain — the spine of the architecture:



Every arrow is one of the controlled types in §4.1. The architecture is integrated precisely because these relationships form a closed chain from principle to conformance, with no orphan element and no unnamed arrow.

4.3 Cardinality and constraint, by relationship

For the four relationships most often misread as one-to-one:

- **realises** (Capability→Service): one *accountable* service per capability; a service may be accountable for several capabilities; supporting services are many-to-many. Accountability is single; realisation is shared.
- **evaluates** (Gate→Control): many-to-many. One gate evaluates several controls; one control is reused across several gates.
- **protects** (Control→Obligation): many-to-many. One control may protect several obligations; one obligation may be protected by several controls.
- **produces** (Service→Record): one authoritative maintainer per record family; other services hold adjacent or derived views only (KF-3004 §7.1).

---

5 · Behavioural Architecture

Sections 3–4 are structural — what exists and how it relates. This section is behavioural — how the architecture acts over time.

5.1 The Event Model

The architecture does not move because a document is passed along; it moves because an **event** is raised. An event is what *happened*; the transition it triggers is the governed *move* it causes (KF-2001). Events divide into two kinds, and keeping them distinct is what locates automation safely:

- **Trigger events (engineering events)** — occurrences that *oblige* the system to consider a transition: new evidence, a scheduled review, a policy change, a challenge, an

AI recommendation, an application outcome, an authority decision. A trigger event may be raised by a human **or an agent**.

- **Emitted events (architectural events)** — notifications the architecture *publishes* when a transition completes: Object Created, Acceptance Granted, Version Created, Superseded, Retired, and the rest of the catalogue in §8.

Trigger events cause transitions; transitions emit events. An emitted event may in turn be a trigger event for another service (a Version Created event triggers dependency re-checking). **No event bypasses a gate.** An AI recommendation is a trigger event, never an acceptance.

5.2 The Transition Model

Every governed move follows one shape, whatever the stage:

```
State A
   ↓  trigger event
Transition requested
   ↓  invokes services
Gate
   ↓  orchestrates
Control evaluation
   ↓  composes results
Decision   (Pass · Fail · Return · Pending · Indeterminate · Pass-
with-Restriction)
   ↓  on Pass
Evidence emitted  +  Events published
   ↓
State B
```

This is the KF-3005 standard transition pattern seen from the integration level. The decisive property: **the evidence and events at the bottom are emitted by the act, not authored afterward** (§9). An Indeterminate decision is never silently treated as Pass.

5.3 The Authority Model (integrated view)

Transitions are not all equal. Some are engineering work an agent may perform under supervision; others are acts of authority that must resolve to an identifiable human. The architecture draws the line by transition, not by service:

Transition class	Example	Who may perform
Computation / structuring	Structure, Operationalise	agent or human, under a named role
Evaluation / justification	Evaluate, Justify	human judgement, agent-assisted
Acceptance / release	Accept, Release	**identifiable human authority**
Stewardship / retirement	Govern, Retire	**identifiable human authority**

Automation may *propose* any of these as a trigger event; it may not *hold the authority* for the acceptance, release, stewardship or retirement classes (KF-1103; KF-3000 §18; KF-3001 IC-16). The detailed authority, delegation and accountability model is KF-3008; this section shows only how authority gates the transition flow.

5.4 The Responsibility Model (integrated view)

Authority is exercised through a chain that runs from a person to a lifecycle action:

```
Actor
   ↓ holds
Role
   ↓ carries
Authority   (and, technically, Permission — assigned by SVC-16)
   ↓ enables
Permitted Services
   ↓ perform
Lifecycle Actions (transitions)
```

The chain is load-bearing in both directions: forward, it says who may act; backward, it makes every action attributable to a role, an authority and a person (invariant 2). Institutional authority (the *right*, held at SVC-06) and technical permission (the *ability*, assigned by SVC-16) are distinct and must not be conflated (KF-3001 IC-13: authority is not permission).

---

6 · The Integrated Object Journey

> **Worked example.** KF-5001 instantiates this journey on a concrete Knowledge Brief, with the actual evidence, authority and records at each step.

This is the heart of the document: one OKO walked through the whole architecture. For each step it shows current state, the services invoked, the records produced, the controls evaluated, the evidence emitted, the traceability updated, and the resulting state. The journey below is generic; it is the exact path the first vertical slice — a MacroLab Singapore Business Impact Assessment, provisionally A3 Consequential — is intended to instantiate and test.

This chapter is the canonical walkthrough of the architecture. Every subsequent implementation, demonstration and conformance assessment should be understandable as an instantiation of this journey. A deployment is not a different architecture; it is this journey run on particular technology. A conformance assessment is not a separate procedure; it is this journey read backwards through its evidence.

Notation per step: **State → Services → Records → Controls → Evidence → Traceability → Next state.**

Capture. Candidate → SVC-01 Object Registry, SVC-15 Provenance → Identity Record, Intake Record, Provenance Seed → CTRL-ID-01 identity resolution, CTRL-ID-02 duplicate detection → *evidence:* the Identity and Intake records themselves → *traceability:* origin attached at birth → **Candidate**.

Structure. Candidate → SVC-02 Representation → Structured Representation, Metadata, Relationship Map → CTRL-STA-01 state validity → *evidence:* the structured representation → *traceability:* representation linked to identity → **Structured**.

Evaluate. Structured → SVC-04 Evaluation, SVC-03 Evidence → Evidence Records, Quality Assessment, Evaluation Verdict → CTRL-EVD-01 evidence presence, CTRL-EVA-01 evaluation completion → *evidence:* the verdict is the test result → *traceability:* claims linked to supporting and defeating evidence → **Evaluated**.

Justify. Evaluated → SVC-05 Justification → Justification Record, Assumption Record, Rationale Record, Epistemic Status → CTRL-JUS-01 justification completeness, CTRL-EVD-03 counter-evidence visibility → *evidence:* the justification record → *traceability:* end-to-end claim-to-source chain assembled → **Justified**.

Accept *(transition, authority)*. Justified → SVC-06 Authority and Acceptance → Acceptance Record, Scope-of-Reliance Record → GATE-ACC-01 orchestrating CTRL-AUT-01 authority mandate, CTRL-ACC-01 acceptance validity; **assurance class A3 confirmed by the governance authority here** → *evidence:* the acceptance record is the evidence of acceptance → *traceability:* reliance scope + authority + date bound to the version → **Accepted**.

Operationalise / Release. Accepted → SVC-12 Operationalisation, SVC-10 Release → Actionable Form, Warrant Linkage, Release Record → GATE-REL-01 (state, acceptance, authority, dependency, provenance controls); depth per A3 → *evidence:* the release record → *traceability:* actionable form linked to its warrant → **Operational**.

Govern. Operational → SVC-07 Stewardship and Review → Stewardship Assignment, Review Schedule, Review Record → CTRL-STW-01 stewardship validity, CTRL-REV-01 review currency → *evidence:* review records emitted on schedule → *traceability:* continuing standing recorded over time → **Governed**.

Apply. Governed → SVC-13 Application and Outcome → Application Record, Explanation Record, Outcome Record, Feedback → GATE-APP-01 eligibility, CTRL-EXP-01 explainability, CTRL-OUT-01 outcome capture → *evidence:* application and outcome records → *traceability:* each material use tied to the version and form used → **in use** → **outcomes**.

Evolve. Outcome/new-evidence trigger → SVC-09 Version and Lineage (re-enters at Evaluate/Justify, re-accepts) → Version Record, Change Record, Lineage Record → GATE-EVO-01 (re-evaluation, re-justification, dependency disposition, renewed acceptance) → *evidence:* the version and change records → *traceability:* new version linked to predecessor; warrant re-earned, not inherited → **Next Version**.

Retire / Supersede. Trigger → SVC-14 Retirement and Preservation (+ SVC-09 supersession) → Retirement Record, Historical Object Record, Supersession Link → GATE-RET-01 (reason, authority, dependency disposition, preservation) → *evidence:* the retirement record → *traceability:* lineage preserved, history append-only → **Historical**.

Two things are visible only at this integrated scale. First, **every record named above was emitted by doing the step**, not by a separate filing activity — the journey *is* the evidence trail. Second, **the six invariants held at every step**: one identity throughout, every transition attributed and justified and inspectable, provenance present from Capture, lineage preserved

through Evolve and Retire. The A3 class raised the *depth* at Evaluate, Justify, Accept and Apply; it never changed the *path*.

---

7 · Integration Patterns

The journey recurs as a small set of reusable patterns. Later implementations instantiate these rather than reinventing flow. Each pattern is a named composition of §4 relationships and the §5.2 transition shape.

- **Capture Pattern** — admit candidate; assign identity; record origin. (SVC-01, SVC-15; GATE-REG-01.) Establishes the invariant floor before any judgement.
- **Review Pattern** — evaluate and justify; assemble the claim-to-source chain. (SVC-03/04/05; GATE-EVA-01, GATE-JUS-01.) Depth graded by class.
- **Acceptance Pattern** — an authority certifies a scope of reliance. (SVC-06; GATE-ACC-01.) The point where justification becomes authorised reliance and class is confirmed.
- **Revision Pattern** — controlled change producing a new version that re-earns warrant and authority. (SVC-09; GATE-EVO-01.) Never overwrites; always links.
- **Supersession Pattern** — a successor replaces a predecessor for a defined scope, lineage preserved. (SVC-09/10; SUPERSEDES.)
- **Retirement Pattern** — end active use while preserving history. (SVC-14; GATE-RET-01.)
- **Monitoring Pattern** — scheduled and event-driven review that keeps standing current and raises Evolve/Retire triggers. (SVC-07, SVC-11; GATE-REV-01.)

A pattern is conformant when it preserves the invariants, attributes its transitions, and emits its evidence natively — regardless of the technology that runs it.

---

8 · Architectural Event Catalogue

The canonical vocabulary of **emitted** events (§5.1). These are the notifications the architecture publishes as transitions complete; they are the machine's observable behaviour and the seam along which services collaborate.

Event	Emitted on	Typical downstream trigger
Object Created	Capture registration	provenance seeding; structuring
Evidence Added	new evidence attached	re-evaluation; Evolve proposal
Review Requested	schedule or challenge fires	governance review
Review Completed	review recorded	Evolve, Retire, or standing renewed
Acceptance Granted	authority accepts	release eligibility; operationalisation

Event	Emitted on	Typical downstream trigger
Acceptance Withdrawn	authority withdraws	suspension; dependency re-check
Version Created	controlled change	dependency impact; re-acceptance
Released	release authorised	application eligibility
Superseded	successor replaces	lineage link; dependent revalidation
Retired	active use ends	preservation; dependent notification

Each emitted event carries, at minimum: the object and version identity, the transition, the responsible actor or authority, and recorded time (KF-3004 Common Record Properties). An event is therefore already a unit of evidence (§9). The authoritative event *records* are maintained by SVC-08 (Lifecycle and Transition) and the originating services; this catalogue is the shared vocabulary, not a new store.

---

9 · Evidence Generation Model

The principle, stated once at the integration level:

> **Architectural behaviour generates conformance evidence as a natural consequence of work, not through a separate administrative act.**

```
Transition ⟶ Event ⟶ Evidence Record ⟶ Traceability Update
```

not

```
Transition ⟶ (stop) ⟶ author a separate evidence artefact
```

The acceptance record is the evidence of acceptance; the transition record is the audit event; the evaluation verdict is the test result; the provenance link is the evidence of derivation; the supersession link is the evidence of lineage (KF-3004 §5.2). A distinct evidence artefact is created only where the act's own records cannot carry the required proof, and then against a recorded justification that they cannot (KF-3001 IC-24).

This is what keeps assurance affordable. Without it, a governed system generates a second knowledge-management problem — the management of evidence that the knowledge was managed. With it, the cost of assurance is folded into the cost of doing the work.

---

10 · Traceability Model

Traceability is not a single artefact; it accumulates in three layers, each produced by ordinary architectural behaviour.

- **Structural traceability** — the *relationships* (§4): identity → version → representation; claim → evidence; form → warrant; successor → predecessor. Produced as records are created and linked.
- **Behavioural traceability** — the *events* (§8): the ordered history of what happened to the object and when. Produced as transitions emit events.
- **Evidential traceability** — the *proof* (§9): the records demonstrating that each material obligation was discharged. Produced as the acts emit their evidence.

A conformance claim (KF-3007) is reconstructed by reading the three layers together: the relationships say what is connected, the events say what happened in order, the evidence says on what basis. The **depth** to which each layer must be reconstructable is set by assurance class (§11): an A0 correction need not be traceable through every principle and control; an A4 acceptance must be reconstructable end to end.

---

11 · Assurance Integration

This chapter bridges to KF-2003 without redefining it. It shows how assurance flows through the architecture as a single graded quantity:

```
Materiality (of the intended reliance)
   ↓ determines
Assurance Class   (A0 … A4, + overlays)      ── KF-2003
   ↓ sets the depth of
Required Controls        (which controls, how strong)   ── KF-3005 §32
   ↓ sets the depth of
Evidence                 (how much, how independent)     ── §9; KF-
3007 Test 3
   ↓ sets the depth of
Traceability             (how far back, how complete)    ── §10; KF-
3007 Test 4
```

The class enters the architecture at one place — the scope of reliance recorded at Acceptance and Release (SVC-06/10) — and from there parameterises gate depth (KF-3005 §28), control strength (§32) and reconstruction depth (KF-3007), all under KF-3001 IC-23. **The path is invariant; the depth is graded.** Two objects of different classes traverse the same journey (§6), pass the same gates and satisfy the same six invariants; the higher-class object satisfies each to greater depth. Class never lowers the invariant floor.

---

12 · Implementation Guidance

Technology-neutral obligations an implementation (KF-4000) must honour when realising this model. When implementing Knowledge Foundry, ensure that:

- **relationships remain explicit** — the §4 relationship types are recoverable, not implied by convention;

- **events remain attributable** — every emitted event resolves to an actor or authority and a time;
- **evidence remains event-native and reconstructable** — the act emits its evidence; assurance reads it without re-authoring;
- **invariants remain preserved under composition** — no combination of services, however optimised, breaks single identity, attribution, justification, inspectability, provenance or lineage;
- **coupling remains loose** — services collaborate across named boundaries and never overwrite one another's authoritative records.

Which technologies deliver these is an implementation decision. That they are delivered is not.

---

13 · Relationship with the Conformance Framework

KF-3006 and KF-3007 are complementary and must not be confused:

- **KF-3006 defines how the architecture *operates*** — the relationships, behaviours, events, evidence and traceability through which a governed OKO is produced.
- **KF-3007 determines *whether that operation conforms*** — applying the four proportionate tests (Constitutional, Operational, Integrity, Reconstructability) to the traceability this model shows the architecture producing.

KF-3006 makes the machine run; KF-3007 inspects the run. The traceability layers of §10 are exactly the material KF-3007 Test 4 reconstructs, and the event-native evidence of §9 is exactly what its Test 3 grades. An architecture that integrates as described here is one that can be found conformant there — the two documents are built to meet.

---

14 · Emergence and Architectural Completeness

14.1 Governance Is an Emergent Property

No single architectural element creates governed knowledge. Governance emerges from the coordinated behaviour of principles, lifecycle, capabilities, services, controls, evidence and traceability acting together on one object. The properties that distinguish an Operational Knowledge Object — trustworthiness, inspectability, accountability and institutional reliance — are therefore **emergent properties of the integrated architecture**, not characteristics of any individual component.

This is the systems-engineering claim at the centre of the programme. Trust does not live inside an object, a service or a record; it is produced by the way they behave together. It follows that trust can be *lost* by a failure of integration even where every component is individually sound — a correctly built service that reaches past a boundary, an evidence record detached from the act that should have emitted it, a transition that skips its gate. Integrity is a property of the composition, which is why this document exists.

14.2 Architectural Completeness

The purpose of architectural integration is not to maximise the number of architectural elements or relationships. Its purpose is to demonstrate that **every normative obligation established by the Canon can be realised, governed, evidenced and reconstructed through the integrated behaviour of the architecture** — and no more than that.

This closes the loop opened by the earlier concern about a "completeness proof". Completeness is not the property that every element maps to another element; it is the property that every *obligation* is discharged by the whole. An architecture is complete when the spine of §Figure 1 runs unbroken from principle to conformance for every obligation the Canon imposes, at the depth each object's assurance class requires. Elements and relationships are justified only insofar as they serve that run; any that do not are complexity for its own sake, and the Integration Admission Test exists to keep them out.

---

15 · Summary

The Knowledge Foundry architecture functions as an integrated system because:

- **principles constrain** how elements may be composed;
- **the lifecycle governs** progression through stages;
- **services realise** capabilities under single accountability;
- **transitions invoke** services and **gates evaluate** controls;
- **controls protect** obligations and invariants;
- **events emit** evidence as a by-product of work;
- **evidence accumulates** as three-layer traceability;
- **traceability supports** proportionate conformance;
- **conformance preserves** the trust the whole system exists to protect.

Read alone, each domain document describes one organ. Read through this model, they are one body: an event enters, a governed transition follows, an OKO advances one justified state, and the evidence that it did so is left behind without anyone stopping to write it down.

Relationship to Subsequent Documents

- **KF-3007 · Architectural Conformance Framework** applies the four tests to the traceability this model produces.
- **KF-3008 · Architectural Authority Framework** holds the detailed authority, delegation and permission model this document views integrately in §5.3–§5.4.
- **KF-3009 · Architectural Crosswalks and Traceability** holds the exhaustive element-to-element mappings this document summarises.
- **KF-4000 · Knowledge Operating System** instantiates these relationships, behaviours and patterns in a particular deployment.

Closing Statement

KF-3006 is where the Reference Architecture stops being a set of specifications and becomes a system. It adds no new element and no new obligation; it shows that the elements already defined compose into a single governed behaviour — capability through service, transition through gate, control through evidence, evidence through traceability, traceability through conformance — and that this behaviour preserves, at every step and every assurance class, the invariants that make the object trustworthy. It is the systems view that the series needed before it could be inspected, and the map the first vertical slice will follow.

KF-3007 · Architectural Conformance Framework

How the Knowledge Foundry Reference Architecture is demonstrated to conform to its own architectural commitments

Document Class: Reference Architecture · Parent Programme: Knowledge Foundry · Parent Document: KF-3000 · Status: Draft for Experiment · Version: 0.2

```
Principle (KF-1000) → Lifecycle (KF-2001) → Materiality (KF-2003)
        → Reference Architecture (KF-3000)
              → Architectural Integration Model (KF-3006)     ← how
the architecture works
              → ▸ Architectural Conformance Framework (KF-3007) ← how
we know it is working
        → Implementation (KF-4000)
```

Up to KF-3006 the programme has been *designing* the architecture. KF-3007 is where the architecture becomes *demonstrable* — where it can show, from its own operation, that it is working as intended. It answers the question that distinguishes Knowledge Foundry from a document store, a personal knowledge tool or an AI answer engine: not *does the system hold knowledge*, but **how can an institution demonstrate — rather than merely assert — that its knowledge architecture is functioning as intended?**

A conformant architecture is not one that is believed to be correct, nor one that merely claims to satisfy its design intent. It is one whose architectural commitments can be demonstrated from observable evidence. The architecture has, in effect, begun to distinguish *truth* from *proof*: the purpose of this framework is not to establish trust by declaration, but to make trust continuously demonstrable.

Purpose

The Architectural Conformance Framework defines how the Knowledge Foundry Reference Architecture is evaluated against its own architectural commitments. It establishes the principles, criteria, evidence requirements and assessment methods through which architectural conformance may be **demonstrated, monitored and reconstructed** over time.

The verb is deliberate. Conformance here is not *checked*, *audited* or *inspected* as an external act imposed on a finished system. It is **demonstrated** — shown to hold from evidence the architecture already produced by operating. The whole framework rests on one sentence:

> **Architectural conformance is demonstrated when every material architectural claim can be reconstructed from observable evidence through explicit traceability.**

That sentence links, without adding a new conceptual layer, everything the programme has built: Canonical Principles become architectural obligations; obligations are discharged through the lifecycle, services and controls; discharge emits evidence; materiality sets its depth; traceability makes it reconstructable; and reconstruction is what “demonstrated” means. The rest of this document is that sentence, unfolded.

Figure 1 · The Conformance Chain

The framework assesses along one logical chain — and the chain matters, because **the assessment never evaluates an obligation directly.** It evaluates whether the *claim* that an obligation has been discharged is supported by evidence.

```
Architectural Obligation
        | asserted as
        ▼
Conformance Claim
        | supported by
        ▼
Evidence            (event-native records)
        | evaluated through
        ▼
Assessment          (the four tests, graded by materiality)
        | produces
        ▼
Finding
```

Each chapter that follows populates one link of this chain: the obligation (§5), the claim (§9), the evidence (§10), the assessment (§7), the finding (§13). The Conformance Claim is the bridge the framework turns on: obligations are architectural facts, but conformance is always a statement *about a claim*, supported or not by what the architecture recorded while it worked.

Scope and the Conformance Admission Test

KF-3007 is technology-neutral. It defines *how conformance is demonstrated*, not the tools that store or compute the evidence (KF-4000), and not the obligations themselves (those are defined in their home documents). A statement belongs in KF-3007 only if:

1. it concerns how architectural conformance is demonstrated, evidenced, judged or reconstructed;
2. it can be stated without naming a specific technology, product or deployment; and
3. it references obligations defined elsewhere in the programme without redefining them.

A statement that creates a new obligation belongs in the document that owns that obligation; a statement that names a technology belongs in KF-4000.

---

1 · Introduction

> **Worked example.** KF-5001 §4 demonstrates these tests, and a reconstructed Conformance Claim, on a single governed object.

1.1 Purpose

To make the architecture demonstrable: to define the obligations against which an implementation is assessed, the dimensions and tests through which assessment proceeds, the

evidence on which it draws, and the findings it yields — all proportionate to the reliance placed on the knowledge governed.

1.2 Scope

In scope: the philosophy and principles of conformance; the definition of architectural conformance and of an architectural obligation; the conformance dimensions; the four-test assessment model; the conformance criteria; the Conformance Claim; the evidence model; the assessment process; findings and exceptions; architectural review; and continuous conformance.

Out of scope: the definition of any obligation (its home document); organisational compliance, HR or process audit; technology, tooling and deployment (KF-4000); and the *judgement calls* of a particular assessment, which this framework structures but does not pre-decide.

1.3 Relationship to Other Documents

- **KF-1000 · Canon** — the principles from which every architectural obligation ultimately derives; conformance preserves architectural intent, and intent originates here.
- **KF-2001 · Lifecycle** — the gates, transitions and invariants whose preservation is the substance of behavioural and lifecycle conformance.
- **KF-2003 · Materiality and Assurance** — the assurance classes that grade how deep evidence, assessment and review must go (§11).
- **KF-3006 · Architectural Integration Model** — describes how the architecture *operates*; KF-3007 determines *whether that operation conforms*. The traceability layers of KF-3006 §10 are exactly what this framework reconstructs.
- **KF-3001–3005, 3008, 3009** — the domains whose obligations are assessed and whose records supply the evidence.

KF-3006 makes the machine run; KF-3007 inspects the run. The two are built to meet.

---

2 · Philosophy of Conformance

Seven convictions set the tone of everything that follows.

1. **Architecture is judged by behaviour, not by documentation.** A conformant architecture is one that *behaves* as intended on real objects, not one that merely describes itself well. Assessment reads behaviour, not intentions.
2. **Conformance is evidence-based.** A claim of conformance stands only on observable evidence. Absent evidence, an obligation is *not assessable*, which is a finding in itself — never a silent pass.
3. **Conformance is proportional.** The depth of assurance demanded matches the reliance placed on the object (KF-2003). Exploratory work is not held to the standard of a board decision, and a board decision is not held to less. Proportionality applies to the depth of evidence, review and assessment — **not to the architectural obligations themselves.** The Canon and the lifecycle invariants remain universal; only the level of

demonstration varies. Low assurance never means "no provenance"; it means less depth above an unchanged invariant floor.

4. **Conformance is reconstructable.** It is not enough that the system behaved correctly; a reviewer must be able to reconstruct *that* it did, after the fact, from the record.
5. **Assessment minimises administrative burden.** Because evidence is event-native (KF-3001 IC-24), assessment reads the records the architecture already produced. Conformance does not commission a parallel paperwork exercise; that would be a second knowledge-management problem.
6. **Conformance emerges from normal operation.** A well-integrated architecture is *continuously* demonstrating conformance simply by working. Assessment formalises and inspects that stream; it does not create it.
7. **Assessment strengthens the architecture.** Findings are not merely reports on implementation quality; they are architectural feedback that informs future revisions of the Reference Architecture itself. The architecture is a learning system, not a static specification — and its conformance framework is one of the instruments through which it learns.

---

3 · What Is Architectural Conformance?

> **Architectural conformance is the observable correspondence between the intended behaviour of the Knowledge Foundry Reference Architecture and the observable behaviour of an implemented system.**

The operative word is **correspondence**, not *compliance*. Compliance suggests obedience to an external rulebook; correspondence asserts that what the architecture *intends* and what the system *does* are the same, and that the sameness can be shown. An implementation is free to differ from the reference architecture in structure, technology and arrangement (KF-3000 Part VIII); it is not free to differ in behaviour where that behaviour discharges an obligation. Conformance is the demonstrable claim that, behaviourally, it does not.

---

4 · Conformance Principles

Every assessment conducted under this framework shall:

- **preserve architectural intent** — assess against the obligations the architecture actually imposes, not against a reviewer's preference;
- **remain technology-neutral** — judge behaviour and evidence, never the choice of tools;
- **rely on observable evidence** — ground every finding in a record that exists, not in inference or reputation;
- **distinguish objective finding from judgement** — separate *what the evidence shows* from *what the assessor concludes from it*, and record both;
- **remain reconstructable** — leave its own trail, so that an assessment can itself be reviewed;

- **respect assurance class** — demand exactly the depth KF-2003 fixes for the object, no more and no less;
- **preserve traceability** — never break, in the act of assessing, the traceability it depends on.

---

5 · Architectural Obligations

The Canon establishes principles. The architecture translates them into **obligations**. The framework assesses whether those obligations are satisfied. The obligation is therefore the unit of conformance — the thing an assessment is *of*.

5.1 Where obligations come from

Obligations are not invented here; they are gathered from where they already live: Canonical Principles (via the handoff of KF-1301), lifecycle gates and the six invariants (KF-2001), engineering practices (KF-2002), the architectural principles, invariants and interaction constraints (KF-3001 AP / INV / IC), the six Governance Primitives (KF-1104), and the proportionality rules of KF-2003. The authoritative enumeration and its upstream tracing are held by KF-3009 (Crosswalks); this framework defines how any obligation is specified and assessed.

5.2 Obligation specification

Each architectural obligation is specified through five fields:

- **Purpose** — the architectural intent it protects.
- **Applicable elements** — the capabilities, services, gates, controls or records it binds.
- **Evidence expected** — the records whose presence demonstrates discharge (event-native wherever possible).
- **Assessment method** — how correspondence is shown: record inspection, cross-service reconciliation, control testing, reconstruction, or independent review.
- **Possible exceptions** — the bounded, attributable deviations that may be accepted without breaching an invariant (§14).

Illustrative obligation. **OBL — Justified Transition.** *Purpose:* no state advances without recorded warrant (Principle VIII; invariant 3). *Applicable elements:* every transition; SVC-08; GATE-JUS-01 and each gate. *Evidence expected:* the transition record and its justification reference, emitted by the act. *Assessment method:* reconstruct any transition and confirm a justification is recoverable. *Exceptions:* none — this is invariant-protecting and ungraded.

Illustrative obligation. **OBL — Sufficient Acceptance Authority.** *Purpose:* reliance is authorised by an identifiable human authority (Principle III). *Applicable elements:* Accept and Release transitions; SVC-06; GATE-ACC-01. *Evidence expected:* the acceptance record naming authority, scope and date. *Assessment method:* confirm the accepting authority held a valid mandate for the object's assurance class. *Exceptions:* a time-limited governance exception at one class below, recorded and visible (§14).

---

6 · Conformance Dimensions

Conformance is not one score. Reducing it to a single pass/fail hides where an architecture is strong and where it is thin. Assessment evaluates eight dimensions, each answering a different question:

Dimension	Asks
Structural	Do the required elements and relationships exist and remain distinct?
Behavioural	Do transitions, gates and controls behave as intended on real objects?
Lifecycle	Is the staged progression, with its gates and returns, honoured?
Governance	Are authority, stewardship, acceptance and review exercised as required?
Evidence	Is the required evidence present, event-native and proportionate?
Traceability	Can the structural, behavioural and evidential trails be reconstructed?
Authority	Does every consequential act resolve to an identifiable human authority?
Information integrity	Are records protected, single-owned and append-only where required?

A dimension may be strong while another is weak; the assessment reports the profile, not an average. Each dimension is evaluated through the four tests of §7, to the depth §11 requires.

---

7 · Architectural Completeness and the Assessment Model

7.1 Architectural completeness precedes conformance

Architectural conformance assumes architectural completeness. Before behaviour can be assessed, the architecture itself must demonstrate that every normative obligation established by the Canon possesses an identifiable realisation path, governance mechanism and evidential outcome. Completeness therefore evaluates the integrity of the *architecture*; conformance evaluates the integrity of its *implementation*.

The two answer different questions:

- **Completeness** — *is the architecture internally coherent?* Does every obligation have somewhere to be discharged, governed and evidenced?
- **Conformance** — *does reality correspond to the architecture?* Does the implemented system actually discharge them?

An incomplete architecture cannot be conformant, because there would be obligations with nowhere to be shown satisfied. Completeness is established once per architectural revision and re-checked whenever a domain document changes (§15); conformance is demonstrated continuously and per object (§16).

7.2 The four tests

The dimensions of §6 are *what* is assessed. The four tests below are *how* — the coverage logic through which any dimension is judged. They replace an open-ended list of disconnected questions with four graded tests, applied in order.

1. **Constitutional coverage** — every Canonical Principle produces at least one enforceable obligation; every obligation has at least one permissible realisation path; nothing below the line contradicts the Canon. *Ungraded.*
2. **Operational coverage** — every capability has an identified realisation mechanism; every transition has trigger, mechanism, authority and recorded outcome; every gate evaluates explicit exit criteria. *Existence ungraded; depth graded.*
3. **Integrity coverage** — every material control addresses an explicit risk, obligation, policy or invariant; every material obligation generates proportionate, event-native evidence; the six invariants hold at full strength. *Graded.*
4. **Reconstructability** — material status and conformance claims can be reconstructed to the depth their reliance requires; a reviewer can establish what happened, why, by whom and on what evidence. *Graded.*

Coverage, not correspondence: an obligation is conformant when *covered* by one or more mechanisms, never on condition that it maps one-to-one to a dedicated element. Single-owner accountability (one accountable service per capability, one authoritative maintainer per record family) is retained and is a different thing from realisation, which may be shared.

---

8 · Assessment Criteria

The four tests are applied through a practical checklist — the criteria against which evidence is read. The architecture *defines* these criteria; the framework *applies* them, which is why they are assessment criteria rather than a separate conformance standard. They are the review questions the programme has used throughout, now seated inside the tests and graded by materiality rather than posed as an absolute proof:

- Does every Canonical Principle produce one or more architectural obligations? *(Test 1)*
- Does every obligation possess at least one realisation pathway? *(Test 1)*
- Does every capability possess one or more realisation mechanisms? *(Test 2)*
- Does every lifecycle transition have trigger, authority, criteria and recorded outcome? *(Test 2)*
- Does every gate evaluate explicit, inspectable exit criteria? *(Test 2)*
- Does every material control protect an explicit obligation, risk, policy or invariant? *(Test 3)*
- Does every material obligation generate proportionate, event-native evidence? *(Test 3)*

- Do the six lifecycle invariants hold for every object, at full strength? *(Test 3)*
- Can every accepted object be reconstructed to the depth its class requires? *(Test 4)*
- Can every material conformance claim be supported from evidence through traceability? *(Test 4)*

A criterion answered "no" is not automatically a Non-Conformance; it is a finding (§13) whose kind depends on materiality and on whether an invariant is breached.

---

9 · The Conformance Claim

The framework introduces one formal concept, and only one.

> **A Conformance Claim is an explicit assertion that a defined architectural obligation has been satisfied for a defined object, transition or scope.**

Examples: *"this lifecycle transition preserved provenance"*; *"this acceptance possessed sufficient authority"*; *"this object satisfied its Consequential-class evidence obligations"*; *"this traceability chain is complete."* A claim is specific: it names the obligation, the subject and, where graded, the assurance class it is claimed against.

The claim is the logical hinge of the whole framework. The chain below elaborates Figure 1, making the traceability step explicit:

```
Conformance Claim
      | supported by
Evidence            (event-native records — §10)
      | read through
Traceability        (structural · behavioural · evidential — KF-3006
§10)
      | enables
Assessment          (the four tests — §7)
      | yields
Finding             (§13)
```

A claim that cannot be supported from observable evidence through explicit traceability is not thereby false — but it is *not demonstrated*, and an undemonstrated claim carries no conformance weight, however plausible. This is the operational meaning of the framework's central sentence: demonstration *is* the reconstruction of a claim from evidence through traceability, to the depth its materiality requires.

---

10 · The Evidence Model

10.1 What counts as evidence

Evidence is the observable record that an obligation was discharged. The framework draws on the records the architecture already maintains (KF-3004):

- **event records** — the emitted architectural events (KF-3006 §8);
- **decision records** — gate decisions and their composed control results;
- **approval / acceptance records** — authority, scope of reliance and refusals;
- **traceability and provenance records** — origin, derivation, custody;
- **version and lineage history** — versions, changes, supersession;
- **lifecycle transition records** — every governed move;
- **relationship graphs** — the structural links among objects, versions and claims.

10.2 Evidence is event-native

Wherever possible, evidence is emitted by the governed act, not authored separately (KF-3001 IC-24; KF-3004 §5.2): the acceptance record *is* the evidence of acceptance, the transition record *is* the event record of the act, the evaluation verdict *is* the test result. A distinct evidence artefact is created only where the act's own records cannot carry the required proof. This is what makes assessment affordable and continuous rather than periodic and burdensome.

10.3 Independence of evidence

A conformance judgement may not rest solely on the self-assertion of the service or actor being assessed (KF-3001 IC-18). Each material judgement cites at least one independently generated source — record inspection, cross-service reconciliation, control testing or external review — and the assurance function reads records without altering them (IC-17). The Conformance and Audit Service (SVC-18) is the architectural means by which evidence is gathered and kept current; it is read-only over everything it assesses.

---

11 · Materiality and Assurance

This framework does not redefine materiality; it consumes it. Assurance flows as one graded quantity, entering at the scope of reliance recorded at acceptance and release:

```
Materiality (of the intended reliance)          ── KF-2003
      ↓ determines
Assurance Class   (A0 … A4, + overlays)
      ↓ sets
Evidence Depth        (how much, how independent)
      ↓ sets
Assessment Frequency  (how often conformance is re-demonstrated)
      ↓ sets
Required Review       (who must review, and how deep)
```

An A0 Exploratory object demonstrates conformance shallowly and rarely; an A4 Critical object demonstrates it comprehensively, independently and often. **The invariants are never graded:** identity, attribution, justification, inspectability, provenance and lineage are demonstrated at full strength at every class. Only the depth above that floor scales. Assurance class therefore parameterises every dimension (§6), every test (§7) and every review cadence (§15) at once, from a single recorded value.

12 · The Assessment Process

An assessment is technology-neutral and proceeds in seven steps:

```
Preparation      define scope, obligations in view, and assurance
classes
      ↓
Evidence         gather the event-native records for the scope
Collection
      ↓
Analysis         apply the four tests across the applicable dimensions
      ↓
Findings         record each as Observation / Recommendation /
Exception / Non-Conformance
      ↓
Exceptions       record accepted deviations, waivers and compensating
controls
      ↓
Recommendations  identify improvements; assign owners
      ↓
Reassessment     re-demonstrate after remediation or at the next
cadence
```

The process is itself governed: an assessment records its own scope, evidence, findings and reasoning, so that the assessment can be reconstructed (Conformance Principle: assessment remains reconstructable).

---

13 · Findings

13.1 The four kinds of finding

An individual finding is one of four kinds — mirroring mature systems-engineering and quality practice without importing its terminology wholesale:

- **Observation** — something noted as interesting or contextual, not itself a problem.
- **Recommendation** — an improvement opportunity; conformance is not at risk, but could be strengthened.
- **Exception** — an intentional, attributable, bounded deviation, accepted as risk with no invariant breached (§14).
- **Non-Conformance** — an architectural obligation is not satisfied; where an invariant is breached, the Non-Conformance is architectural, not merely operational.

13.2 The overall assessment outcome

Findings roll up into one assessment outcome, deliberately not “pass / fail”:

- **Conformant** — obligations in scope are satisfied and demonstrated.

- **Conformant with observations** — satisfied; observations or recommendations recorded.
- **Partially Conformant** — some obligations satisfied; specific, bounded gaps recorded with remediation.
- **Non-Conformant** — one or more material obligations unsatisfied, or an invariant breached.
- **Not Assessable** — evidence is insufficient to demonstrate the claim. This is a real outcome, never silently converted to Conformant, and never to Non-Conformant either: it obliges either evidence or escalation.

Consistent with KF-3005, a Not-Assessable / Indeterminate condition is never treated as Pass, and an exception never converts an invariant breach into conformance.

13.3 Confidence in Findings

Conformance and *confidence in the finding* are not the same thing, and the framework keeps them distinct. Two objects may both be found Conformant while warranting very different confidence: one with complete, event-native evidence, unbroken traceability and strong provenance; another with sparse evidence, partially reconstructed history and thin traceability. Both may satisfy their obligations; our confidence that they do differs.

Every material finding therefore carries a **confidence qualifier** reflecting the completeness, independence and directness of the evidence it rests on — for example *high* (complete, event-native, independently corroborated), *moderate* (adequate but partly reconstructed) or *low* (sparse, or reliant on the assessed party's own assertion). Confidence never changes the finding, but it travels with it: a Conformant-at-low-confidence result is a signal to strengthen evidence, not a defect in the object.

Confidence interacts with assurance class. A Consequential or Critical object found conformant only at low confidence is itself worth a recommendation (§13.1), because at high reliance the *demonstrability* of conformance matters as much as conformance — an obligation discharged but poorly evidenced has not been fully demonstrated. Confidence is thus how the framework surfaces the gap between *being* conformant and being *able to show it*.

---

14 · Exceptions

Architecture is never perfect, and a framework that cannot record imperfection drives it into hiding. Exceptions make deviation visible and governed rather than silent. The framework recognises:

- **accepted deviations** — a difference from the reference architecture that satisfies the obligation by another means (a conforming variation);
- **temporary waivers** — a time-limited governance exception permitting operation one class below the norm, accepted as risk by a named authority;
- **known limitations** — an acknowledged gap not yet remediated, recorded with its consequence;

- **compensating controls** — an alternative control that discharges the obligation where the standard mechanism cannot;
- **future remediation** — the recorded plan and owner for closing a gap.

Every exception is attributable, bounded and **visible until closed, formally accepted as risk, or rendered irrelevant by retirement** (KF-3001 IC-19). No exception may waive an invariant: an invariant breach is a Non-Conformance, not an exception.

---

15 · Architectural Review

This chapter institutionalises architectural governance and is the home of the "completeness" discipline — recast, as elsewhere, from an absolute proof into a periodic, materiality-graded review. Review occurs at defined moments:

- **Continuous review** — the event-native checks of §16, running as the architecture operates.
- **Vertical-slice review** — assessment of a real object taken end to end (the first being the MacroLab Singapore assessment); the primary means by which this framework and KF-2003 are themselves validated before being frozen.
- **Major-release review** — before a release enters consequential reliance.
- **Periodic architecture review** — at a cadence set by the highest assurance class in scope.
- **Reference-architecture revision** — when a domain document changes, to confirm coverage still holds.
- **Independent review** — conducted without sole reliance on the assessed parties (IC-18), for A4 and regulated overlays.

Completeness is demonstrated here not by showing that every element maps to another, but by showing that **every material obligation is discharged by the integrated behaviour of the architecture, to the depth its class requires** — the spine of KF-3006 Figure 1 running unbroken, tested against real objects.

---

16 · Continuous Conformance

The architectural invariant of *continuous conformance* (KF-3000 §3) means conformance is maintained over time, not verified once. Because evidence is event-native, much of it can be demonstrated continuously by detective controls that watch the record stream and raise findings automatically:

- missing or incomplete provenance;
- broken or unreconstructable traceability;
- invalid or illegal transitions;
- expired or withdrawn acceptance still being relied upon;
- transitions lacking required authority;

- evidence gaps against the object's assurance class;
- stale review or breached currency conditions.

Each detected condition is raised as an event and recorded as a finding, exactly as a manual assessment would record it. Continuous conformance is periodic assessment's complement, not its replacement: the stream catches drift as it happens; the reviews (§15) judge the whole.

---

17 · Relationship with Implementation

An implementation is **free to innovate** — to choose technologies, combine or distribute services, and arrange components as it sees fit — **provided the architectural obligations remain satisfied and demonstrable.** KF-3007 never dictates *how* an obligation is met; it requires only that its being met can be shown from evidence through traceability. Two structurally dissimilar implementations may both be fully conformant. Conformance constrains behaviour and its demonstration; it does not constrain design.

---

18 · Summary

Architectural conformance is **not the objective** of the Knowledge Foundry architecture. It is the **means** by which the architecture preserves trust, accountability and institutional reliance throughout its evolution. The framework:

- translates principles into **obligations** (§5);
- assesses them across **dimensions** (§6) through four graded **tests** (§7) and a practical **checklist** (§8);
- turns each into a **Conformance Claim** (§9) supported by **event-native evidence** (§10);
- grades depth by **materiality** (§11);
- runs a reconstructable **process** (§12) yielding four kinds of **finding** and five **outcomes** (§13);
- makes imperfection **visible** through exceptions (§14);
- institutionalises **review** (§15) and **continuous** demonstration (§16);
- and leaves **implementation** free to innovate within satisfied obligations (§17).

All of it collapses back to one sentence: *architectural conformance is demonstrated when every material architectural claim can be reconstructed from observable evidence through explicit traceability.*

Relationship to Subsequent Documents

- **KF-3008 · Architectural Authority Framework** defines the authority whose sufficiency the Authority dimension assesses.

- **KF-3009 · Architectural Crosswalks and Traceability** holds the authoritative obligation enumeration and upstream tracing this framework assesses against.
- **KF-4000 · Knowledge Operating System** instantiates the evidence sources and continuous checks in a particular deployment.

Closing Statement

Across the KF-3000 series a pattern has completed: KF-3000–3005 answer *what exists*; KF-3006 answers *how it works*; KF-3007 answers *how we know*. Structure, then behaviour, then assurance. KF-3007 is therefore not an audit checklist appended to the architecture — it is the culmination of the architectural argument, the point at which a governed knowledge system can turn to those who rely on it and *demonstrate*, from its own ordinary operation, that their reliance is warranted.

Status and Next Step

This document is **Draft for Experiment**. Its dimensions, tests, criteria, evidence model and the Conformance Claim are hypotheses the first vertical slice is designed to test: whether evidence arises event-natively at sufficient depth, whether the four tests apply cleanly to a real object, whether the findings taxonomy is workable, and which criteria matter most in practice. It advances to *Validated with Exceptions* after the slice, and to *Approved* only after the resulting revisions and a formal architecture review. Finalising it ahead of the slice would repeat the error it corrects — deepening assurance in advance of proven use.

KF-3008 · Architectural Authority Framework

How authority is established, exercised, delegated and evidenced within a governed knowledge system

Document Class: Reference Architecture · Parent Programme: Knowledge Foundry · Parent Document: KF-3000 · Status: Draft · Version: 0.1

```
Principle (KF-1000) → Lifecycle (KF-2001) → Materiality (KF-2003)
        → Reference Architecture (KF-3000)
              → Architectural Integration Model (KF-3006)      ← how
the architecture behaves
              → Architectural Conformance Framework (KF-3007)  ← how
that behaviour is evaluated
              → ▸ Architectural Authority Framework (KF-3008)  ← who
may make it consequential
        → Implementation (KF-4000)
```

Throughout the architecture the programme has used words that presuppose authority — *actor*, *reviewer*, *approver*, *owner*, *authority*, *acceptance*, *delegation*, *accountability* — without ever defining authority as an architectural construct. That omission is the last remaining gap in the conceptual core, and it matters because Knowledge Foundry is fundamentally about **institutional** knowledge, and institutional knowledge cannot exist without institutional authority.

This document is therefore not about organisational hierarchy, and it is emphatically not an access-control document. It answers one question:

> *Who possesses the architectural authority to create, evaluate, accept, modify, supersede and retire Operational Knowledge Objects — and how is that authority established, exercised and evidenced?*

It is titled a **Framework**, not a Model, because it establishes governing rules, not merely a description of relationships.

Purpose

The Architectural Authority Framework defines how authority is **represented, exercised, delegated and evidenced** within the Knowledge Foundry Reference Architecture. It establishes the architectural rules governing who may perform consequential lifecycle actions, under what conditions those actions are legitimate, and how authority itself becomes observable through traceable evidence.

The document is about *authority itself* — not organisational charts, and not the technologies that authenticate people or enforce permissions. Where KF-3006 asked *how the architecture behaves* and KF-3007 asked *how that behaviour is evaluated*, KF-3008 asks *who possesses the legitimate authority to make that behaviour consequential.* Without it, acceptance — the act at the centre of the lifecycle — remains an abstract concept with no defined source.

Read together, these three documents answer three complementary questions — which is why they are three documents and not one long chapter:

Question	Document
How does the architecture behave?	KF-3006 · Integration
How do we know it behaved correctly?	KF-3007 · Conformance
Who had the legitimate right to make it behave that way?	KF-3008 · Authority

Figure 1 · Why Authority Belongs in the Architecture

> **Worked example.** KF-5001 §5 shows a real acceptance — who held the authority, and how it resolves from decision back to institution.

```
Institution
     ▼ constitutes
Governance
     ▼ defines
Role
     ▼ carries
Authority
     ▼ legitimises
Decision
     ▼ effects
Lifecycle Transition
     ▼ emits
Evidence
     ▼ warrants
Institutional Reliance
```

Read top to bottom, the figure explains authority's architectural purpose in a single line: an institution constitutes governance, which defines roles, which carry authority, which legitimises the decisions that effect lifecycle transitions, which emit the evidence that warrants an institution in relying on what it knows. Authority is the link without which reliance has no legitimate source.

Scope and the Authority Admission Test

KF-3008 is technology-neutral and institution-neutral. It defines architectural authority; it does not name the mechanisms that implement it. A statement belongs in KF-3008 only if:

1. it concerns the establishment, exercise, delegation, scope, failure or evidence of architectural authority;
2. it can be stated without naming a specific technology, product, directory or protocol; and
3. it references elements defined elsewhere in the programme without redefining them.

Deliberately out of scope, because it belongs to implementation (KF-4000): Active Directory, IAM, OAuth, RBAC schemes, authentication, credential and permission stores. Also distinct is *technical permission* — the architectural ability to execute an action — which is owned by the

Access and Integrity Service (SVC-16). This framework governs institutional **authority** (the *right*); SVC-16 governs **permission** (the *ability*); KF-4000 implements both. Authority is not permission (KF-3001 IC-13), and this document never conflates them.

---

1 · Introduction

1.1 Purpose

To define authority as a first-class architectural construct: what it is, who holds it, how it is bounded and delegated, how it fails, and how it is made observable — so that every consequential act in the lifecycle resolves to a legitimate, attributable and reconstructable authority.

1.2 Scope

In scope: the philosophy and principles of authority; the distinction of authority from responsibility, accountability, ownership and permission; the authority model; authority types, scope and lifecycle mapping; delegation; authority evidence; separation of authority; the interaction of authority with materiality; authority failures; continuous authority; and the concept of institutional reliance.

Out of scope: organisational design and staffing; access-control and identity technology (KF-4000); the substantive content of governance policy; and the definition of any lifecycle stage, control or record (their home documents).

1.3 Relationship to the Canon

Authority is the architectural realisation of **Principle III** — *knowledge becomes enduring capability only through collective authorisation* — and of the **human-accountability invariant**: labour may be distributed across people and machines, but accountability may not be discharged into a substrate (KF-1103; KF-1050). This framework fixes where that irreducible human authority sits and how it is exercised.

1.4 Relationship to Lifecycle

The lifecycle names an **Authority** role and an **Accept** transition and insists that acceptance is an act of authority *distinct from* justification (KF-2001). KF-3008 defines the authority that the Accept transition — and every other consequential transition — requires. KF-3005 identifies these needs abstractly, as *Required Authority Class*, *Required Decision Role* and *Required Separation Condition* in its Transition-to-Authority Crosswalk (§60); KF-3008 defines who may occupy those roles.

1.5 Relationship to Conformance

Authority is itself assessable. KF-3007 lists an **Authority** conformance dimension; this framework supplies what that dimension tests — *was authority valid, was delegation valid, was acceptance legitimate, can authority be reconstructed?* (§15). Because authority is evidenced

event-natively (§10), it is demonstrable on the same terms as everything else the architecture does.

---

2 • Philosophy of Authority

Before the principles, one distinction frames them. **Authority establishes *who* may perform a consequential act; legitimacy concerns *whether* that act satisfies the architectural conditions governing its exercise.** A decision may originate from a valid authority yet still fail to achieve architectural legitimacy if required obligations, controls or evidence are absent — just as a technically sound recommendation from an unauthorised source never acquires authority at all. Authority is the entitlement; legitimacy is the entitlement properly exercised. This framework defines authority; legitimacy is what the Conformance Framework (KF-3007) later assesses, which is why an "authority valid" finding and an "act legitimate" finding are distinct there.

Authority in Knowledge Foundry rests on nine convictions.

1. **Authority exists to protect institutional reliance.** It is not an administrative convenience; it is the mechanism by which an institution may legitimately base action on what it knows. Authority exists *because* institutions rely — not the other way around.
2. **Authority is vested in institutional roles, not individuals.** A role carries authority; individuals exercise it only by virtue of occupying the role. People join, move and leave; the authority persists in the role and the institution.
3. **Authority is attributable.** Every exercise of authority resolves to an identified role and, through it, an identified person or accountable body. No consequential act is anonymous.
4. **Authority is bounded.** No authority is universal. Every authority has a scope beyond which it does not reach (§7).
5. **Authority is delegable — under rule.** Authority can be extended to others through governed delegation, without losing attribution or accountability (§9).
6. **Authority is observable.** Authority that cannot be seen cannot be relied upon; it must leave evidence of its mandate, its exercise and its bounds (§10).
7. **Authority is revocable.** Authority can be withdrawn, and withdrawal must not retroactively invalidate acts legitimately taken while it held (§14).
8. **Authority is proportional to materiality.** The authority required to make an act consequential rises with the reliance placed on it (§12). The invariant that *some* human authority is required never varies; the *level* of authority does.
9. **Authority is persistent, not personal.** Because institutional knowledge must survive personnel change, authority belongs to the institution through defined governance structures rather than to particular people. An individual's departure ends their *exercise* of authority; it does not remove the authority, which remains vested in the role and reconstructable in the record (§14).

---

3 · What Is Architectural Authority?

Architectural authority is the **legitimate institutional right to make a knowledge act consequential** — to cause a lifecycle transition on which the institution will rely. It is distinct from five concepts with which it is routinely confused. The architecture keeps all six apart.

Concept	Definition	Distinct because
Authority	the right to make a consequential decision (accept, release, retire …)	it *legitimises* an act
Responsibility	the obligation to perform or maintain work	one may be responsible for work one has no authority to accept
Accountability	answerability for an outcome	accountability may rest with a body that did not itself act
Ownership	stewardship of an object over time	an owner (Steward) may lack the authority to accept a new version
Permission	the technical ability to execute an action	permission may exist without authority, and must not substitute for it
Expertise	the ability to form an informed judgement	expertise informs a decision but does not entitle one to make it consequential

The confusions this dissolves are load-bearing: **ownership does not imply authority** (a Steward maintains an object but may not be the accepting Authority); **permission does not imply accountability** (a service may be permitted to execute a transition an agent proposed, while accountability remains with a human); **responsibility does not imply acceptance authority** (an Evaluator is responsible for the verdict but does not thereby accept the object); and **expertise does not imply authority** (an expert may hold no acceptance authority, while an executive may hold acceptance authority without being the technical expert). The architecture keeps informed judgement separate from the entitlement to act on it. Acceptance is the exclusive province of Authority.

---

4 · Authority Principles

Every authority in the architecture shall be:

- **explicit** — recorded as a mandate, never assumed from position or context;
- **attributable** — resolvable to an identified role and person or accountable body;
- **bounded** — scoped by domain, stage, object type, materiality, organisation, jurisdiction and time (§7);

- **proportional** — sufficient for the object's assurance class, and no broader than needed (§12);
- **reviewable** — subject to periodic confirmation and re-validation;
- **traceable** — evidenced such that any exercise can be reconstructed (§10);
- **durable across organisational change** — vested in roles and the institution, so that departures and reorganisations do not orphan authority or void its history (§14);
- **technology-independent** — defined without reference to any authentication or permission technology.

---

5 · The Authority Model

The heart of the framework. Authority flows through a chain from the institution to a specific lifecycle action:

```
Institution
     ▼ constitutes
Governance Structure
     ▼ defines
Role
     ▼ carries
Authority
     ▼ permits
Permitted Decisions
     ▼ effect
Lifecycle Actions
```

The decisive property is stated once and holds throughout: **authority is vested in institutional roles, not people; individuals exercise it only by virtue of occupying those roles.** An institution constitutes a governance structure; the governance structure defines roles; roles carry authority; authority permits particular decisions; those decisions effect lifecycle actions. Individuals *occupy* roles and thereby *exercise* — but never *own* — the authority a role carries. This is what lets a governed knowledge system survive personnel change without losing the legitimacy of its past decisions: the person who accepted an object may leave, but the role that accepted it, and the institution behind the role, remain.

An **accountable body** (a board, a committee, a named office) is modelled as a role held collectively. Collective authority is still attributable — to the body, and to the members who acted — and still resolves to identifiable humans, satisfying Principle III.

5.1 Authority Resolution

Every consequential act must be able to answer one architectural question: *from where did this authority originate?* **Authority resolution is the authority equivalent of traceability** — the reverse reading of the model chain, from an act back to the institution that ultimately warranted it.

```
Decision
     ▲ resolves to
```

```
Authority
     ▲ carried by
Role
     ▲ defined by
Governance Structure
     ▲ constituted by
Institution
```

Where traceability reconstructs *what happened and why*, authority resolution reconstructs *by what right*. A consequential act whose authority cannot be resolved to a role, a governance structure and an institution is not legitimately authorised, however competent the actor — and the failure is detectable (§13: ambiguous or orphaned authority). Resolution is graded like everything else: the depth to which authority must be resolvable rises with assurance class (§12), and it is demonstrated from the mandate, delegation and decision records the act already emitted (§10).

---

6 · Authority Types

"Approver" is too coarse. The architecture distinguishes authority by the consequential act it legitimises; each type has a distinct architectural meaning and attaches to a distinct transition.

Authority type	Legitimises	Principal transition / gate
Creation	admitting a candidate as a governed object	Capture / GATE-REG-01
Evaluation	certifying a quality or justification verdict	Evaluate, Justify
Acceptance	authorising reliance on a justified version	**Accept / GATE-ACC-01**
Publication (Release)	making an accepted version operationally eligible in a scope	Release / GATE-REL-01
Revision	authorising a controlled change to a new version	Evolve / GATE-EVO-01
Supersession	replacing a version or object with a successor	Supersede
Retirement	ending active operational life	Retire / GATE-RET-01
Delegation	conferring authority on another role, within bounds	(meta-authority; §9)

Acceptance, Publication, Supersession and Retirement authorities are the consequential ones: they must resolve to an identifiable human authority and may never be discharged into a substrate (KF-3006 §5.3). Creation and Evaluation may be agent-assisted under a named role. **Delegation authority is meta-authority** — the authority to confer authority — and is the most tightly bounded of all.

6.1 Authority Relationships

Authorities do not act in isolation; they stand in defined relationships to one another. Naming these relationships is the bridge from *what* authorities exist to *how far* each reaches (§7):

- **authority may delegate authority** — conferring a bounded portion on another role (§9);
- **authority may require concurrence** — a decision needing two or more authorities acting together before it takes effect;
- **authority may require independent review** — an act valid only once reviewed by an authority outside the deciding role (§11);
- **authority may supersede authority** — a higher or later mandate displacing an earlier one, with lineage preserved;
- **authority may be constrained by policy** — a governance rule narrowing what an otherwise-valid authority may do;
- **authority may be suspended** — temporarily rendered unexercisable pending review, without being revoked (§13).

These relationships are themselves governed and evidenced: a concurrence, a required review, a supersession or a suspension is recorded as it happens, on the same event-native terms as any other authority act (§10). They are the vocabulary in which the scope rules of §7 and the separation pattern of §11 are expressed.

---

7 · Authority Scope

No authority is universal; scope is what makes it bounded. An authority is defined over one or more dimensions, and an exercise outside its scope is not weak authority but *no* authority.

Scope dimension	Bounds authority by
Knowledge domain	the subject matter it may govern
Lifecycle stage	the transitions it may effect
Object type	the kinds of object it applies to
Materiality / assurance class	the highest class it may accept (§12)
Organisation	the organisational unit it acts within
Jurisdiction	the legal or regulatory context it covers
Time	the effective period over which it holds

Scope is recorded with the mandate and checked at every gate. An acceptance authority valid for A2 Routine objects in one domain does not thereby authorise an A3 Consequential object, an object of another type, or an object outside its effective period. This is how the architecture prevents unlimited authority without enumerating every permitted act.

---

8 · Authority and the Lifecycle

Authority maps onto the lifecycle transition by transition. Each consequential move is gated by the authority type it requires:

```
Candidate ──(Creation Authority)──▶ admitted
     ▼
Evaluated / Justified ──(Evaluation Authority)──▶ verdict certified
     ▼
Accepted ──(Acceptance Authority)──▶ reliance authorised
     ▼
Operational ──(Publication Authority)──▶ eligible in scope
     ▼
Next Version ──(Revision Authority)──▶ change authorised, warrant re-
earned
     ▼
Superseded ──(Supersession Authority)──▶ successor installed, lineage
preserved
     ▼
Historical ──(Retirement Authority)──▶ active life ended, history
intact
```

This is the concrete filling of KF-3005's abstract Transition-to-Authority Crosswalk (§60). A transition presented without the authority its gate requires does not proceed; the gate returns *Fail* or *Not Assessable*, never a silent Pass (KF-3005; KF-3007 §13).

---

9 · Delegation

Delegation is how authority extends beyond a single role-holder without loss of attribution or accountability. The architecture recognises seven distinct constructs, and conflating them is a common source of authority failure:

- **Delegation** — a role-holder confers a bounded portion of its authority on another role, for a defined scope and period; accountability remains with the delegator.
- **Substitution** — a temporary stand-in exercises a role's authority in the holder's absence, under the same bounds.
- **Escalation** — an act requiring authority beyond the current holder's scope is routed upward to a role that holds it.
- **Revocation** — conferred authority is withdrawn before its natural expiry, by an attributable act (§14).
- **Expiry** — conferred or temporary authority lapses at the end of its effective period without further action.
- **Temporary authority** — authority granted for a bounded period or purpose, expiring automatically.
- **Emergency authority** — an exceptional, time-boxed grant permitting action under urgent conditions, recorded as a governance exception and visible until reviewed (KF-3001 IC-19); it may never waive an invariant.

Every delegation is itself an exercise of **Delegation authority** (§6), bounded by scope (§7) and evidenced (§10). Delegated authority can never exceed the delegator's own scope: one cannot confer what one does not hold.

---

10 · Authority Evidence

The framework's connection to KF-3007: **authority itself becomes evidence.** Because authority must be observable (Principle 6), every mandate, delegation and exercise leaves a record — not because any particular technology requires it, but because the architecture requires authority to be reconstructable.

Authority is evidenced through records the architecture already maintains (KF-3004): the **Authority Mandate Record** (AUT-01) and **Delegation Record** (AUT-02), maintained by SVC-06; the **Acceptance Record** and **Scope-of-Reliance Record**; role-assignment records; and the decision and transition records emitted by every gated act. In implementation these may take the form of an approval record, a signature, a delegation entry, a role assignment, an acceptance event or a decision log — but the architecture requires only that the mandate, its bounds and its exercise be recoverable.

This evidence is **event-native** (KF-3001 IC-24; KF-3004 §5.2): the acceptance record *is* the evidence that acceptance authority was exercised; the delegation record *is* the evidence that delegation was valid. Authority does not commission a separate register of its own legitimacy; the exercise of authority generates its evidence as a natural consequence of the architectural event rather than through a parallel administrative process (KF-3006 §9), exactly as the rest of the architecture does.

---

11 · Separation of Authority

Some acts should not be authorised by the same role that produced the work. The architecture recognises separation as a **pattern**, applied where materiality warrants rather than mandated everywhere:

- **Evaluator ≠ Approver** — the role that certifies a verdict is not the role that accepts reliance on it;
- **Author ≠ Reviewer** — the role that produced content is not the role that reviews it;
- **Publisher ≠ Accepting Authority** — the role that releases into operation is not the role that accepted the version.

Whether separation is *required* is a function of assurance class (§12): Exploratory and Working objects may collapse these roles; Consequential and Critical objects increasingly require them, enforced by the Separation-of-Duties control (CTRL-SEG-01). Separation is thus proportional, like everything graded in the architecture — a floor that rises with reliance, not a uniform burden.

---

12 · Authority and Materiality

Authority is one of the quantities materiality grades. The bridge to KF-2003 runs:

```
Materiality (of the intended reliance)      ── KF-2003
     ▼ determines
Assurance Class   (A0 … A4, + overlays)
     ▼ sets
Required Authority   (which type, whose confirmation, what separation)
     ▼ produces
Evidence   (mandate, delegation, acceptance — event-native)
     ▼ enables
Conformance   (the Authority dimension — KF-3007)
```

Concretely: the object **owner proposes** an assurance class; for **A3 and A4** the **governance authority confirms** it (KF-2003 §7). Higher classes demand higher and, where the pattern applies, *separated* authority; A4 Critical typically demands a formally constituted accountable body and independent review. The universal floor never moves — *some* identifiable human authority is always required — but its level, formality and separation rise with reliance. Automation may recommend a class or propose a transition, but may not hold the confirming authority for consequential or critical objects (KF-3001 IC-16).

Authority scales with consequence, not seniority. Higher materiality does not necessarily demand a more *senior* individual; it demands authority *appropriate to the consequence and scope of reliance*. Seniority and authority often coincide in practice, but the architecture equates required authority with the reliance at stake, never with rank: an A4 acceptance needs an authority mandated for A4 consequence in that domain and scope, which may or may not be the most senior person available. Equating authority with hierarchy is a category error the scope rules of §7 exist to prevent.

---

13 · Authority Failures

Treating authority architecturally means naming how it fails. Each failure is a detectable condition with an architectural consequence, not merely an operational inconvenience.

Failure	Condition	Architectural consequence
Missing authority	a consequential act has no mandate behind it	transition blocked; Non-Conformance
Expired authority	the mandate's effective period has lapsed	act refused; escalation required
Conflicting authority	two mandates assert incompatible rights over one act	act suspended pending resolution
Competing authority	two authorities each claim the right to make the same act	act suspended; precedence resolved before proceeding

Failure	Condition	Architectural consequence
Ambiguous authority	the responsible authority cannot be resolved (§5.1)	act *Not Assessable*; not a silent pass
Orphaned authority	the role that held the authority was removed with no successor	acts requiring it blocked until reassigned; history preserved
Delegation exceeded	conferred authority is used beyond its bounds	act invalid; delegation reviewed
Materiality threshold exceeded	authority is used on an object above its permitted assurance class	act invalid; escalation to sufficient authority (§12)
Authority unavailable	the required authority cannot act in time	escalation or emergency authority (§9), recorded
Authority withdrawn	the mandate is revoked	future acts blocked; past acts remain valid (§14)

None of these is resolved by proceeding. An authority failure yields a governed outcome — block, refuse, suspend, escalate or record an exception — and never a silent transition.

---

14 · Continuous Authority

Authority is dynamic. Organisations restructure; roles are created and dissolved; people are appointed and depart; delegations are granted and revoked. The architecture treats authority as a living state, subject to continuous validation — and it holds one rule above all the churn:

> **A change in authority never invalidates historical evidence.**

An acceptance made by a role that later ceased to exist remains a valid acceptance: it was legitimate at the time it was made, and the record carries the effective time at which the authority held (KF-3004 time semantics). Revocation, departure and reorganisation act **forward**, on future exercises of authority; they do not reach back to void past acts or erase the lineage of decisions already taken (the lineage invariant, KF-2001). This is what allows an institution to reconstruct, years later, that a decision was properly authorised *when it was made*, even though every person and role involved has since changed. Continuous validation keeps present authority current; append-only history keeps past authority true.

---

15 · Relationship with Conformance

Authority is assessable on exactly the terms KF-3007 defines. The Authority conformance dimension reconstructs, from event-native evidence, the answers to four questions for any consequential act:

- **Was authority valid?** — did a mandate exist, in scope and in its effective period?

- **Was delegation valid?** — did any delegation stay within the delegator's bounds?
- **Was acceptance legitimate?** — did the accepting authority hold sufficient authority for the object's assurance class?
- **Can authority be reconstructed?** — to the depth the class requires, can the whole chain from institution to act be recovered?

A "no" to any is a finding (KF-3007 §13), graded by materiality. Because authority is evidenced natively, these questions are answered from the record the act already produced — assessment reads authority, it does not re-interview it.

---

16 · Summary

Authority is **not an administrative convenience.** It is one of the architectural mechanisms through which institutional trust becomes possible. This framework:

- defines authority as the legitimate right to make a knowledge act consequential (§3), distinct from responsibility, accountability, ownership and permission;
- vests it in **roles, not people** (§5), so it survives organisational change;
- distinguishes authority **types** (§6), **scope** (§7) and **failures** (§13);
- maps it onto the **lifecycle** (§8) and governs its **delegation** (§9);
- makes it **observable and event-native** (§10), **separable** where materiality warrants (§11), and **proportional** to assurance class (§12);
- keeps it **continuous** without letting change rewrite history (§14);
- and renders it **assessable** on the same terms as every other architectural behaviour (§15).

Above all it defines the concept that gives authority its purpose: **institutional reliance**. *Institutional reliance is the explicit willingness of an institution to base decisions, actions or obligations upon an Operational Knowledge Object.* Authority exists because institutions rely — and this framework is how that reliance acquires a legitimate, attributable, reconstructable source.

Relationship to Subsequent Documents

- **KF-3009 · Architectural Crosswalks and Traceability** carries the authoritative tracing from Canonical Principle III and the human-accountability invariant through the authority types, scopes and controls named here.
- **KF-4000 · Knowledge Operating System** implements authority — identity, authentication, permission and delegation technology — within the architectural rules this framework fixes, and never in place of them.

Closing Statement

Across the KF-3000 series a three-layer structure has completed. KF-3000–3005 are the **structural architecture** — *what exists*. KF-3006 and KF-3007 are the **behavioural**

architecture — *how it behaves and how we know*. KF-3008 is the **institutional architecture** — *who may legitimately make that behaviour consequential*. A knowledge architecture that can only represent and process knowledge is a sophisticated knowledge system; it becomes a framework for **governed institutional knowledge** only when it can establish who has the legitimate authority to create, evaluate, accept and rely upon what it holds. That is the layer KF-3008 supplies, and with it the conceptual core of Knowledge Foundry is complete.

Status and Next Step

This document is **Draft · Version 0.1**. Its authority types, scope dimensions, delegation constructs and failure modes are exercised directly by the first vertical slice, whose A3 Consequential object requires a confirmed acceptance authority and, under §11, the beginnings of separation. The slice is the occasion on which this framework — like KF-2003 and KF-3007 — is first tested against a real object rather than asserted, and revised before the series is baselined.

KF-3009 · Architectural Crosswalks and Traceability

The navigation layer of the Knowledge Foundry Reference Architecture — how the whole fits together

Document Class: Reference Architecture (Navigational) · Parent Programme: Knowledge Foundry · Parent Document: KF-3000 · Status: Draft · Version: 0.1

```
Principle (KF-1000) → Methodology (KF-2000) → Reference Architecture
(KF-3000)
        → Domains (KF-3001–3005) → Integration (KF-3006) → Conformance
(KF-3007) → Authority (KF-3008)
        → ▸ Crosswalks and Traceability (KF-3009) → Implementation
(KF-4000)
```

The architecture is complete at KF-3008. Principles, lifecycle, materiality, reference architecture, integration, conformance and authority together provide everything essential. This document therefore **introduces no new architectural domain and no new first-order concept.** It performs a different role: it is the architecture's *synthesis* — the atlas that accompanies the constitutional text. The constitution defines the rules; the atlas helps readers see where everything sits and how the parts relate, without creating new law.

Purpose

The Architectural Crosswalks and Traceability document provides the **navigational layer** of the Knowledge Foundry Reference Architecture. It maps architectural concepts, responsibilities and relationships across the architecture, clarifies document ownership, and demonstrates how the architecture forms a coherent, navigable and traceable system whose architectural integrity can be understood **without duplicating its normative content**.

This is *navigation*, not architecture. Every entry below points to a home document that owns the concept; KF-3009 owns only the *relationships between* the documents — the map, the crosswalks, the traceability matrices and the relationship catalogue. Where a crosswalk here and a home document appear to differ, the home document governs and this document is corrected.

Navigational Principle

> **Every architectural concept has one authoritative home. Relationships may be described elsewhere, but normative ownership is never duplicated. KF-3009 exists to preserve that discipline.**

That single rule governs the whole document. It is why KF-3009 can map, cross-reference and trace freely without ever becoming a competing source of truth: it describes where things live and how they relate, and defines only the relationships themselves.

Figure 1 · The Master Traceability

The single diagram that explains the entire architecture — the programme's architectural signature. It threads every layer, from the obligations the Canon imposes to the reliance the institution is finally warranted in placing:

```
Canonical Principles
        ▼ obligate
Architectural Obligations
        ▼ realised by
Capabilities
        ▼ delivered through
Services
        ▼ operate over
Lifecycle
        ▼ made consequential by
Authority
        ▼ enforced through
Controls
        ▼ emit
Evidence
        ▼ accumulates as
Traceability
        ▼ reconstructed as
Conformance
        ▼ warrants
Institutional Reliance
```

This figure is referenced throughout the document, and §11 reads it in full.

What this document does and does not own

KF-3009 defines **no new ontology, no new lifecycle, no new authority and no new governance mechanism.** It is the architecture's index, atlas and navigation layer. What it *does* own — because these are relationships, not concepts — are: the Concept-Ownership register (§4), the Relationship Catalogue (§10), and the master traceability matrices (§§6–9). These were already promised to KF-3009 by the closing sections of KF-3002–3005 ("KF-3009 preserves traceability from principle through capability, service, record and control").

---

1 · Introduction

Scope. The reading guide, the architecture map, concept ownership, the relationship catalogue, and the principle / obligation / lifecycle / materiality crosswalks, plus reference indexes and reserved extensions.

Audience. Every reader of the Reference Architecture, especially those arriving after KF-3000 who need to orient before going deep, and maintainers who must keep the series coherent as it evolves.

Relationship to the Reference Architecture. KF-3009 sits beside KF-3000 as its companion: KF-3000 opens the architecture, KF-3009 helps navigate it. It is the document every future reader should read *second*.

---

2 · How to Read the Architecture

Different readers enter at different places. This is the shortest path to what each needs:

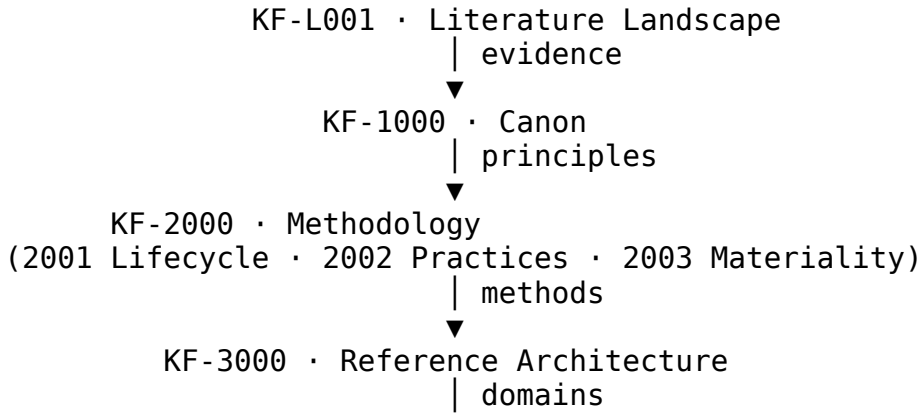
Reader	Enter at	Then read
Architect	KF-3000 Reference Architecture	KF-3001–3005, then KF-3006
Engineer	KF-3006 Integration Model	KF-3003 Services, KF-3005 Control
Reviewer / Assessor	KF-3007 Conformance Framework	KF-3004 Records, KF-3006 §10
Governance lead	KF-3008 Authority Framework	KF-2003 Materiality, KF-2001 Lifecycle
Implementation team	KF-4000 Knowledge Operating System	KF-3006, KF-3007, KF-3008
Executive / sponsor	KF-0001 Architecture Map, then this §14	KF-1000 Canon (purpose)
Researcher	KF-L001 Literature Landscape	KF-1000 Canon

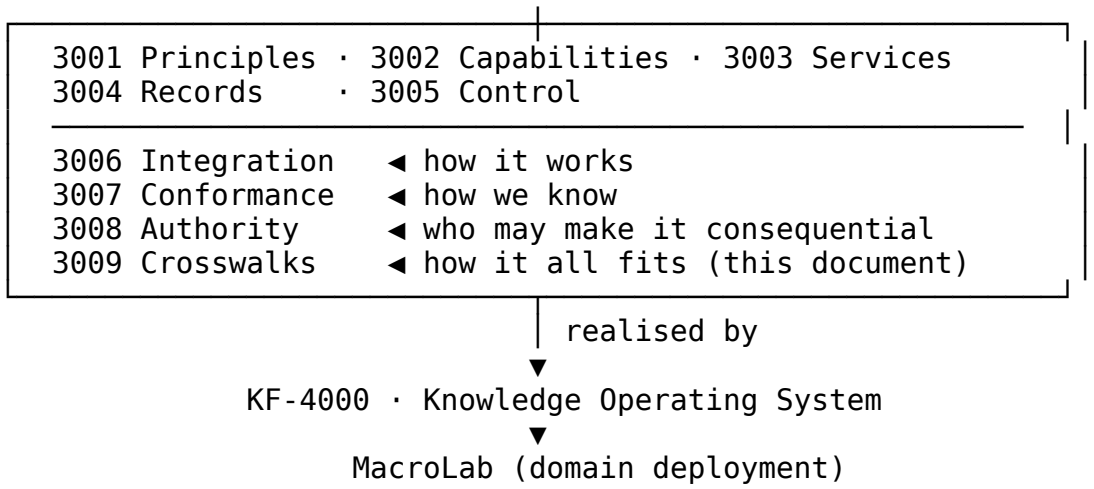
No reader should have to read the series front to back to become productive; each path above is sufficient to act, and the crosswalks below fill in the rest on demand.

---

3 · The Architecture Map

The authoritative one-page orientation. (Its standalone pull-out is **KF-0001 · Architecture Map**; this is the same map, seated in its atlas.)





Programme-wide navigation: KF-0001 Map · KF-0002 Controlled Vocabulary

Read by architectural layer, the same series stacks into four bands:

Constitution	KF-1000
Method	KF-2000 (2001–2003)
Architecture	KF-3000–3009
Implementation	KF-4000

---

4 · Concept Ownership

The rule that keeps the architecture free of duplicated definitions: **every concept has exactly one primary home.** When a concept appears in several documents, one owns it and the others reference it.

Concept	Primary home	Secondary references
Canonical Principle · Constitutional Invariant · Governance Primitive	KF-1000	KF-3001, KF-3006, KF-3009
Operational Knowledge Object *(defined KF-1101; architectural unit KF-3000 §2)*	KF-1000	KF-3000, KF-3004
Lifecycle — stages, transitions, lifecycle invariants	KF-2001	KF-3005, KF-3006, KF-3008
Engineering Practice	KF-2002	KF-3002, KF-3000 §22

Concept	Primary home	Secondary references
Materiality · Assurance Class · Governance Overlay	KF-2003	KF-3005, KF-3007, KF-3008
Architectural Principle · System Invariant · Interaction Constraint	KF-3001	KF-3005, KF-3007
Capability	KF-3002	KF-3003, KF-3000 §23
Service	KF-3003	KF-3004, KF-3005, KF-3006
Information / Record Object · Event-native evidence *(record model)*	KF-3004	KF-3001 (IC-24), KF-3007
Gate · Control · Transition pattern	KF-3005	KF-3006, KF-3007
Integration — relationships, behaviour, patterns, events	KF-3006	KF-3007, KF-3009
Conformance · Conformance Claim · Findings · the four tests	KF-3007	KF-3006, KF-3008, KF-4000
Authority · Delegation · Institutional Reliance	KF-3008	KF-2001, KF-2003, KF-3007, KF-3009
Crosswalks · Traceability matrices · Relationship Catalogue	KF-3009	KF-3006 §4, KF-3001 §24
Controlled Vocabulary · Required Distinctions · user-facing terms	KF-0002	KF-3009 §13.3
Architecture Map (one-page)	KF-0001	KF-3009 §3
Implementation — components, technology, deployment	KF-4000	KF-3006, KF-3007, KF-3008
Worked examples · canonical slice · case studies	KF-5000 / KF-5001	KF-2001, KF-3006, KF-3007, KF-3008

Where two documents touch one concept, one holds the authoritative treatment and the other holds an adjacent or derived view — the same single-owner discipline the record model applies to data (KF-3004 §7).

---

5 · Concept Relationship Crosswalk

For each core concept: what it depends on, what it supports, and where it is referenced. This is the “if I change this, what else is affected?” table.

Concept	Depends upon	Supports	Most significant consumers
Lifecycle	Canon (II, X)	Authority, Conformance, Control	KF-3005, KF-3006, KF-3008
Materiality	Canon; Lifecycle	Authority depth, Evidence depth, Review cadence	KF-3005, KF-3007, KF-3008
Capability	Canon; Lifecycle; Practices	Services	KF-3003, KF-3000 §23
Service	Capability	Records, Transitions, Controls	KF-3004, KF-3005, KF-3006
Control	Invariants; Obligations	Gates	KF-3005, KF-3007
Authority	Lifecycle (Accept); Canon III	Acceptance, Release, Retirement	KF-3007 (Authority dimension), KF-2003
Evidence (event-native)	Transitions; Records	Traceability, Conformance	KF-3006 §9, KF-3007 §10
Conformance	Obligations; Evidence; Traceability	Institutional reliance	KF-3008, KF-4000
Institutional Reliance	Authority; Conformance	(the purpose the architecture serves)	KF-1204, KF-2003

6 · Principle Traceability *(constitutional matrix)*

Every Canonical Principle is realised somewhere in the architecture. This matrix is the constitutional traceability the Canon requires (KF-1301): a principle with no realisation would be an aspiration, not an obligation.

Canonical Principle	Principal architectural realisation
I — governs OKOs, not knowledge	OKO as architectural unit (KF-3000 §2); Object Registry (SVC-01); single-identity invariant
II — engineered as a lifecycle	Lifecycle (KF-2001); Lifecycle & Control Architecture (KF-3005)
III — collective authorisation	Authority Framework (KF-3008); Acceptance (SVC-06); GATE-ACC-01
IV — governance is constitutive	Governance Primitives across services; Controls (KF-3005); Conformance (KF-3007)
V — modality redistributes labour, not obligation	Modality-independence invariant (KF-3000 §3); human-accountability boundary (KF-

Canonical Principle	Principal architectural realisation
	3006 §5.3)
VI — engineered, collective, governed at once	Architectural Integration Model (KF-3006)
VII — trust is a property of transitions over time	Provenance & lineage records (KF-3004); Temporal Continuity (SVC-19); continuous conformance (KF-3007 §16)
VIII — inspectable justification per transition	Justification (SVC-05); justified-transition invariant; event-native evidence (KF-3004 §5.2)
IX — explainable across modality	Explainability control (CTRL-EXP-01); Application & Outcome (SVC-13)
X — designed for governed evolution	Version & Lineage (SVC-09); Evolve gate (GATE-EVO-01); the lifecycle loop

7 · Obligation Traceability *(complete lineage)*

One level below principles. Each material obligation can be traced along a single lineage from constitutional source to demonstrated conformance — the chain KF-3007 reconstructs:

```
Canonical Principle
      ▼ handed to method (KF-1301)
Architectural Obligation
      ▼ realised by
Architecture   (capability · service · gate · control · record)
      ▼ instantiated in
Implementation (KF-4000)
      ▼ emits
Evidence   (event-native)
      ▼ reconstructed as
Conformance
```

Worked lineage — justified transition. Principle VIII → obligation "no state advances without recorded warrant" → realised by Justification Service (SVC-05), GATE-JUS-01 and the justified-transition invariant → instantiated by a deployment's justification store → emits the justification and transition records → reconstructed by KF-3007 Test 4 as the claim *"this transition carried its warrant."* No link in this chain is optional for a material obligation; the depth of each is graded by assurance class (KF-2003).

This lineage is the **minimum reconstruction path** required for any material conformance claim. Additional evidence may exist, and higher assurance classes require more of it, but no conforming implementation may omit one of these architectural stages — a claim with a broken link in the chain is *Not Assessable* (KF-3007 §13).

8 • Lifecycle Crosswalk

Each lifecycle stage, read across its architectural participants. This is the single-page operational reference for "what does this stage require?"

Stage	Authority (KF-3008)	Principal service	Principal gate	Principal control	Evidence emitted	Primary conformance dimension (KF-3007 §6)
Capture	Creation	SVC-01 Object Registry	GATE-REG-01	CTRL-ID-01/02	Identity, Intake, Provenance Seed	Structural
Structure	—	SVC-02 Representation	GATE-STR-01	CTRL-STA-01	Structured Representation, Metadata	Structural
Evaluate	Evaluation	SVC-04 Evaluation	GATE-EVA-01	CTRL-EVA-01	Quality Assessment, Verdict	Evidence
Justify	Evaluation	SVC-05 Justification	GATE-JUS-01	CTRL-JUS-01	Justification, Assumptions, Rationale	Evidence
Accept	**Acceptance**	SVC-06 Authority & Acceptance	GATE-ACC-01	CTRL-AUT-01, CTRL-ACC-01	Acceptance, Scope-of-Reliance	Authority
Operationalise / Release	Publication	SVC-12 / SVC-10	GATE-REL-01	CTRL-REL-01, CTRL-DEP-01	Actionable Form, Release	Governance
Govern	Stewardship	SVC-07 Stewardship & Review	GATE-REV-01	CTRL-STW-01, CTRL-REV-01	Review, Stewardship records	Governance
Apply	— (use under permission)	SVC-13 Application & Outcome	GATE-APP-01	CTRL-EXP-01, CTRL-OUT-01	Application, Explanation, Outcome	Behavioural
Evolve	Revision	SVC-09 Version &	GATE-EVO-01	CTRL-VER-01, CTRL-	Version, Change,	Lifecycle

Stage	Authority (KF-3008)	Principal service	Principal gate	Principal control	Evidence emitted	Primary conformance dimension (KF-3007 §6)
		Lineage		LIN-01	Lineage	
Retire	Retirement	SVC-14 Retirement & Preservation	GATE-RET-01	CTRL-LIN-01	Retirement, Historical Object	Traceability

Every row's evidence is emitted by the act (event-native), and its conformance is assessed on the KF-3007 Authority, Lifecycle and Evidence dimensions.

---

9 • Materiality Crosswalk

How assurance class propagates through the architecture — the single value that grades authority, evidence, review, monitoring and conformance at once (KF-2003; KF-3007 §11; KF-3008 §12).

Class	Required authority	Evidence depth	Review	Monitoring	Conformance depth
A0 Exploratory	none (provenance only)	minimal, event-native	none required	none	shallow, rare
A1 Working	named owner	basic + versioning	collaborative	on change	light
A2 Routine	authorised acceptance role	defined scope of use	periodic	currency checks	standard
A3 Consequential	confirmed acceptance authority; separation begins	independent, counter-evidence	independent review	active	deep
A4 Critical	accountable body; segregation of duties	comprehensive, independent	formal + independent	continuous	full, reconstructable end-to-end

The six lifecycle invariants hold identically across every row; only the depth above that floor scales.

10 · The Relationship Catalogue

The permanent home of the canonical relationship vocabulary first stated in KF-3006 §4.1. **The relationships defined below constitute the canonical architectural vocabulary for expressing relationships throughout the Knowledge Foundry programme. New relationship types shall not be introduced unless adopted into this catalogue.** Every architectural diagram in the programme draws its arrows from here, so that a relationship always means one thing. These are *collected*, not invented here.

Relationship	Category	Reads as
constrains	normative	Principle / interaction constraint → element
realises	implementation	Capability → Service
invokes	behavioural	Transition → Service
consumes	information	Service → Information object (read)
produces	information	Service / Transition → Information object
evaluates	governance	Gate → Control
protects	assurance	Control → Obligation / Invariant
generates	evidential	Transition / Obligation → Evidence
traces	provenance	derived state → source; successor → predecessor
delegates	authority	Authority → Authority (bounded)

Three structural relationships support composite decisions and grading: **references** (decision → authoritative record, without copying it), **attributed-to** (transition → actor / authority), and **graded-by** (obligation depth → assurance class). *Supersession* and *governance oversight* are read as instances of **traces** (lineage) and **constrains** (policy) respectively, not as separate primitives.

11 · Reading the Master Diagram

Figure 1 (near the front of this document) is the single diagram that explains the entire architecture: it threads every layer, from the obligations the Canon imposes to the reliance the institution is finally warranted in placing. Every earlier diagram in the series is a *segment* of it — KF-3006 Figure 1 is its Principles-to-Conformance span; KF-3008 Figure 1 is its Authority-to-Reliance span. Read whole, it is the architectural argument of Knowledge Foundry in a single

column, and the reason the series ends here, with synthesis rather than another framework. It is the programme's architectural signature — the figure future presentations, papers and talks should reach for first.

---

12 · Reading Paths *(recommended order by role)*

- **Executive / sponsor** — KF-0001 → KF-1000 (purpose) → Figure 1 → KF-3009 §14.
- **Reader (general)** — KF-0001 → KF-3000 → KF-3006 → this document.
- **Architect** — KF-3000 → KF-3001 → KF-3002 → KF-3003 → KF-3004 → KF-3005 → KF-3006.
- **Implementer** — KF-3006 → KF-3003 → KF-3005 → KF-3007 → KF-3008 → KF-4000.
- **Governance officer** — KF-1000 → KF-2001 → KF-2003 → KF-3008 → KF-3007.
- **Researcher** — KF-L001 → KF-1000 → KF-3000 → KF-3006.

---

13 · Reference Tables

13.1 Document Index

ID	Title	Role
KF-L001	Literature Landscape	evidence
KF-1000	Canon	principles
KF-2001	Operational Knowledge Object Lifecycle	method
KF-2002	Engineering Practices	method
KF-2003	Materiality and Assurance	method
KF-3000	Reference Architecture	architecture (umbrella)
KF-3001	Architectural Principles and Invariants	domain
KF-3002	Capability Model	domain
KF-3003	Service Architecture	domain
KF-3004	Information and Record Model	domain
KF-3005	Lifecycle and Control Architecture	domain
KF-3006	Architectural Integration Model	behaviour
KF-3007	Architectural Conformance Framework	assurance
KF-3008	Architectural Authority Framework	institutional
KF-3009	Architectural Crosswalks and	navigation (this)

ID	Title	Role
	Traceability	
KF-4000	Knowledge Operating System	implementation
KF-5000	Exemplars & Case Studies	illustrative (series note)
KF-5001	Canonical Vertical Slice (four-day-week Brief)	illustrative exemplar
KF-0001	Architecture Map	navigation (one-page)
KF-0002	Controlled Vocabulary	navigation (terminology)

13.2 Figure Index

Figure	Home	Shows
Integration Spine	KF-3006 Fig. 1	Principles → Conformance
Conformance Chain	KF-3007 Fig. 1	Obligation → Claim → Evidence → Assessment → Finding
Why Authority Belongs	KF-3008 Fig. 1	Institution → … → Institutional Reliance
Master Traceability	KF-3009 Fig. 1	the full architectural column

13.3 Concept Index and Glossary

For any term's authoritative definition, see the primary home in §4; for controlled definitions, required distinctions and the user-facing vocabulary, see **KF-0002 · Controlled Vocabulary**. KF-3009 does not redefine terms; it locates them.

---

14 · Architectural Evolution

The architecture is designed to change without losing coherence. Two distinct governance modes keep evolution disciplined — and the distinction between *adding* architecture and *changing* it is itself load-bearing.

14.1 Controlled Extension *(new concepts)*

Adding to the architecture. The numbering space is reserved rather than filled ad hoc:

- **KF-3010+** — reserved for future architectural domains, admitted only if they pass the KF-3000 Architectural Admission Test and own a concept no existing document owns.
- **Deeper treatments** — a dedicated temporal or dependency model, say — would extend, not revise, the existing domains, and would be traced here.
- **Discipline** — no new document may duplicate a concept owned in §4; a proposed addition that would must instead amend the owning document. This is the Navigational Principle applied to growth.

14.2 Controlled Revision *(existing concepts)*

Changing what already exists. Revision follows the inheritance rule the Canon fixes: a lower level may change how it satisfies a higher one, but may not silently revise the higher one.

- **Amend at the home** — a concept is revised only in its owning document (§4); every secondary reference then inherits the change.
- **Constitutional amendments** — a change to a Canonical Principle is made against the Landscape's evidence and propagated downward (KF-1000); it is never overridden quietly in a lower document.
- **Re-trace after revision** — when any home document changes, the crosswalks here and the conformance completeness check (KF-3007 §7.1) are re-run to confirm coverage still holds.

The series is considered conceptually complete at KF-3008; anything beyond elaborates, operationalises or validates that core. Extension adds a new home; revision changes an existing one; neither duplicates a concept already owned.

---

15 · Summary

The Knowledge Foundry Reference Architecture is intentionally modular. Architectural understanding therefore depends not only on the individual documents but on the relationships that bind them together. This document provides those relationships — the map, the ownership boundaries, the crosswalks, the relationship catalogue and the master traceability — ensuring that the architecture may continue to evolve while remaining coherent, navigable and traceable.

The series answers six questions in order: *what is the architecture* (KF-3000), *what are its domains* (KF-3001–3005), *how do they operate together* (KF-3006), *how is its integrity demonstrated* (KF-3007), *who may legitimately make it consequential* (KF-3008), and *how does the whole fit together* (KF-3009). Ending with synthesis rather than another specialised framework is the point: the last document is the one that makes all the others easier to understand and maintain.

Together these documents establish a complete architectural lineage from constitutional principle to institutional reliance. KF-3009 closes the Reference Architecture by making that lineage explicit, navigable and maintainable.

Closing Statement

KF-3009 is the atlas to the constitution. It creates no law and owns no concept save the relationships among the rest. Its success is measured not by what it adds but by what it makes navigable: an architect, an implementer, a reviewer or a governance officer should be able to enter the architecture anywhere, find where every concept lives, follow any obligation from principle to conformance, and trust that the whole remains one coherent, traceable system. With it, the Reference Architecture is not merely a set of documents — it is a map of a single governed knowledge system, and a means of keeping it coherent as it evolves.

KF-0002 · Controlled Vocabulary and Architectural Distinctions

The authoritative terms of the Knowledge Foundry programme, the distinctions they must preserve, and their user-facing projection

Document Class: Programme-Wide Reference · Parent Programme: Knowledge Foundry · Status: Draft for Experiment · Version: 0.1

Purpose

This document is the single authoritative home for Knowledge Foundry terminology. It exists so that a term means the same thing in the Canon, the methodology, the architecture, the implementation, the interface and the training material — and so that distinctions the programme depends on are not quietly collapsed by casual synonymy.

It does not re-cut the architecture's concepts; those are defined in their home documents. It **collects** them, records the distinctions that must never be treated as synonyms, and defines the small user-facing vocabulary that ordinary users meet in place of the full ontology. Where a definition here and a definition in a home document appear to differ, the home document governs the concept and this document is corrected.

> **Namespace note.** This document is numbered **KF-0002** on the assumption that the **KF-0000 series is reserved for programme-wide constitutional and navigational artefacts** (glossary, programme map, document register). It is deliberately *not* numbered KF-1001, which is already occupied by the Canon's opening chapter ("The Nature of Governed Knowledge Systems"). If the KF-0000 series is designated otherwise, this document is renumbered without changing its content.

The document has four sections: **authoritative definitions**, **required distinctions**, **permitted user-facing language**, and **mapping rules**.

Section 1 · Authoritative Definitions

Terms are grouped by the layer that owns them. Each definition is a compact restatement of the governing document; the citation is the authority.

Constitutional (KF-1000)

- **Governed knowledge system** — a socio-technical system that holds operational knowledge objects and governs their transition from one justified state to another across modalities (KF-1001).
- **Operational Knowledge Object (OKO)** — the unit of design: a versioned representation together with its attached evidence, justification and acceptance-state (KF-1101). The system governs the object, never "knowledge itself".
- **Canonical Principle** — an enduring obligation true of any governed knowledge system, admitted only if it survives the two-question test (KF-1000 Part III).
- **Governance Primitive** — one of the six operations by which warrant is conferred: evidence, authority, provenance, validation, versioning, reversibility (KF-1104).

- **Constitutional invariant** — one of the five properties whose loss would make a system cease to be a governed knowledge system: inspectability, justification, provenance, governed evolution, human accountability (KF-1050).

Lifecycle and method (KF-2000)

- **Lifecycle stage** — a distinct engineering responsibility with an entry condition and an exit gate (KF-2001).
- **Transition** — a governed move between states; always attributable and justified (KF-2001).
- **Event** — an occurrence that triggers a transition; may be raised by a human or an agent, but the transition it triggers remains governed (KF-2001).
- **Gate** — a justification checkpoint; the point at which an object may advance only if the obligations of its stage are discharged (KF-2001; KF-3005).
- **Lifecycle invariant** — one of the six continuously checkable properties of an object at every instant: single identity, attributable transitions, justified transitions, inspectable transitions, provenance always present, lineage preserved (KF-2001).
- **Materiality** — the degree to which an object, claim, decision or transition may affect consequential action, reliance, obligation, exposure, representation or preservation (KF-2003).
- **Assurance class** — the ordinal grade (A0 Exploratory · A1 Working · A2 Routine · A3 Consequential · A4 Critical) that sets the depth of assurance required; attaches to the release / scope of reliance (KF-2003).
- **Governance overlay** — an orthogonal attribute (regulated, confidential, public, client-facing, legally-privileged) adding obligations without changing class (KF-2003).

Architecture (KF-3000 series)

- **Capability** — an enduring ability the architecture must possess, independent of workflow, role, service, technology or modality (KF-3002).
- **Service** — a logical mechanism with defined responsibility that realises capabilities and maintains records (KF-3003).
- **Component** — the technical element that implements part of a service (KF-4000).
- **Practice** — the disciplined human or socio-technical method by which work is performed (KF-2002).
- **Control** — a reusable mechanism that establishes whether one required condition holds; protects an obligation, risk, policy or invariant (KF-3005).
- **Information / Record object** — a structured unit of governed information with one authoritative maintaining service (KF-3004).
- **Evidence record** — the durable record that a material obligation or decision was discharged; wherever possible emitted by the governed act itself (KF-3004 §5.2).
- **Conformance claim** — an assertion that an object is governed, accepted, current or reliable, reconstructable to the depth its reliance requires (KF-3007).
- **Release** — a version authorised for a defined scope of operational reliance; the anchor to which assurance class attaches (KF-3000 §2).

Navigation (KF-3009, KF-0001)

- **Relationship** — a semantic connection between architectural elements; the canonical relationship types are fixed in KF-3009 §10.
- **Dependency** — a relationship in which one element requires another for correct operation. Every dependency is a relationship; not every relationship is a dependency.
- **Traceability** — the ability to reconstruct lineage across one or more relationships using observable evidence (KF-3009; KF-3006 §10). Traceability is what makes a relationship *demonstrable* rather than merely asserted.

Section 2 · Required Distinctions

These pairs carry the programme's intellectual weight. They **must not** be treated as synonyms in any document, interface or procedure. Collapsing any one of them re-introduces the ambiguity the discipline exists to resolve.

Distinction	The difference that must be preserved
Evidence vs. justification	Evidence is what supports a claim; justification is the reasoned case — including counter-evidence, assumptions and rationale — for why it may be believed.
Justification vs. acceptance	Justification establishes *why* an object may be believed; acceptance establishes that an identified *authority will rely* on it. Neither substitutes for the other.
Principle vs. invariant	A principle governs *progression* (what a transition must earn); an invariant governs *conformance* (what must remain true at every instant).
Capability vs. service	A capability is *what* must be possible; a service is *one mechanism* by which it is delivered. A capability need not reduce to a single service.
Object vs. representation	The object is the enduring governed identity; a representation is one form of its content. One object may have several representations.
Identity vs. version	Identity is immutable and singular; a version is one immutable state of the object's content under that identity.
Provenance vs. lineage	Provenance is origin and derivation of content; lineage is the historical relationship between objects and versions (branch, supersede, consolidate).
Transition vs. event	An event is what *happened*; a transition is

Distinction	The difference that must be preserved
	the governed *move* it triggers. An event never bypasses a gate.
Control vs. gate	A control tests one condition; a gate *orchestrates* several controls into one decision (KF-3005 §29).
Evidence record vs. conformance claim	An evidence record is the durable proof an obligation was discharged; a conformance claim is an assertion reconstructable *from* such records.
Assurance class vs. maturity profile	A class grades how much rigour a *particular object* is governed to (KF-2003); a maturity profile describes how much *architecture* an implementation has stood up (KF-3000 §20).
Relationship vs. dependency vs. traceability	A relationship is any semantic connection between elements; a dependency is a relationship of *required operation*; traceability is the *reconstruction* of lineage across relationships from observable evidence (KF-3009).

Section 3 · Permitted User-Facing Language

Ordinary users experience governance; they do not study its ontology. Interfaces and operating procedures may use the following **five-state vocabulary** in place of the internal lifecycle states. These terms are **controlled projections** of the architecture — simplifications sanctioned here — not replacements for the formal vocabulary, which continues to govern beneath them.

User-facing term	What the user understands
Draft	Being worked on; not yet approved to rely on
Under Review	Being checked; approval not yet complete
Approved	Approved to rely on, within a stated scope
Needs Updating	Something has changed; its currency is in question
Superseded	Replaced by a newer object or version

No other user-facing state names are sanctioned without amendment to this document. Assurance class may be surfaced to users in plain terms (e.g. “for significant decisions”) but the class codes A0–A4 are architectural, not user-facing.

Section 4 · Mapping Rules

The user-facing terms map to internal architecture as follows. The mapping is the contract: an interface showing a user term asserts that the corresponding internal condition holds.

User-facing term	Internal meaning
Draft	Object exists but is not accepted for operational reliance (pre-Acceptance lifecycle states)
Under Review	One or more required evaluation or justification activities are incomplete (Evaluate / Justify stages, or a returned object)
Approved	Accepted for a defined scope of reliance (Acceptance recorded; a valid Release within scope)
Needs Updating	A currency or validity condition has been breached or questioned (a Continuing-Review or dependency event has fired)
Superseded	A successor object or version has replaced the object for defined use (Supersession Link recorded; lineage preserved)

Rules of use. 1. A user term may be shown only when its internal condition is satisfied; the interface does not assert states the architecture has not reached. 2. The mapping is many-to-one downward: several internal states may present as one user term (e.g. Evaluate and Justify both present as *Under Review*). It is never one-to-many: a single internal condition does not present as two contradictory user terms. 3. The internal state remains the source of truth for every governed decision. User terms are for comprehension, not for authority.

Status and Maintenance

This document is **Draft for Experiment**. The five-state projection and the required distinctions are to be tested against the first vertical slice: does the five-state vocabulary suffice for a real user of a Consequential object, and do the distinctions hold up when one object is taken end to end? It moves to *Approved* after that evidence. The glossary is maintained as a controlled vocabulary in its own right — changes are governed, attributable and versioned, exactly as the architecture requires of any governed record.