N O R M

Reference Architecture

How a tool of this suite is built — the norm from which every tool architecture is derived.

v0.4 — draft for correction · as of 2026-08-14 · thirteen principles and ten prohibitions, each with identifier and declared anchoring

Kaspar Brönnimann

1 · Purpose, scope, demarcation

This norm states how a tool of the suite must be built. It is the input of the architecture agent: from it — together with the manifestos, the capability catalogue and the operating norm — the architecture of a concrete tool is derived.

It is harvested, not invented. Every principle stems either from a manifesto sentence or from a decision that has already been taken and proven in a running product. Where a principle has no such origin, this is declared.

WHAT THIS NORM REGULATES	WHAT THIS NORM DOES NOT REGULATE
The construction of a tool: layers, core objects, states, tenant separation, evidence.	What a tool does substantively. That belongs in the method norm of the product.
Where decisions may be made and where they may not.	What the surface looks like. That belongs in appearance and operation.
Which building blocks are used and how they interact.	What the building blocks are able to do in detail. That belongs in the capability catalogue.
What evidence a tool must produce.	How review takes place. That belongs in the test concept.

2 · The principles, their identifiers and their anchoring

Thirteen principles carry the architecture. Each has a fixed identifier so that later documents can refer to them instead of repeating them — a conformity statement writes "fulfils RA-09" and no longer "no direct model contact".

The origin of a principle falls into three items of information which so far stood in one column and do different things. In this version they are kept separate; what the three mean is stated in section 2.2.

IDENTIFIER	PRINCIPLE	ANCHORING	DERIVATION	PROVEN IN
RA-01	The core decides, the model proposes, the human takes responsibility	PH-03, supported by PH-01	—	Technology Radar (E4)
RA-02	No authoritative facts from the model	PH-10, sharpened by WA-02	architecture concept, fact boundary	invariant test PIA
RA-03	Every core object carries its governance mixin	PH-02	test concept	—
RA-04	Draft and confirmation are separate fields	PH-03	—	divergence log (E9), arguments not values (E14)
RA-05	Release is a gate; a change invalidates it	PH-03	test concept ADR-T04, knowledge life cycle	—
RA-06	Changes are transitions, not breaks	PH-08 (indirect)	test concept	versioned assessments (E3, E9)
RA-07	Configuration before programming	none, technical rule	test concept ADR-T05	operating and maintenance costs per branch
RA-08	Tenant separation is structure, not filter	outstanding — see 2.2	—	—
RA-09	No direct model contact	PH-03, executed by SC-01, SC-02	framework, model portability	—
RA-10	Every result	PH-02,	—	—

	carries its provenance	sharpened by WA-02		
RA-11	Uncertainty is a field, not a formulation	PH-04, supported by WA-03	—	certainty grades (E16)
RA-12	Controlled vocabulary	none, technical rule	RA-05, RA-06	category thicket (E7, E12)
RA-13	A tool produces evidence, not only results	PH-02	test concept	—

2.1 · The rule for the identifiers

An identifier is never redefined.

Identifiers are only worth something if "fulfils RA-09" means the same in five years as today. Therefore three rules apply:

- New principles are appended and receive the next free number. They are not inserted thematically, even if their content would belong elsewhere.
- If a principle is withdrawn, its number remains occupied and is kept as withdrawn. It is not reassigned.
- If the content of a principle changes substantially, that is a new principle with a new number — not the same number with different content. The old one refers to the new one.

The same three rules apply to the identifiers of the negative space (NR-01 ..., Chapter 9).

This version numbers for the first time; the order is therefore still chosen thematically. From now on it is frozen. That is the same discipline as with the invariants of the quality model and with the architectural decisions of the test concept.

2.2 · What anchoring means — and what it does not

Three items of information that do different things and therefore do not belong in one column:

- Anchoring — a manifesto sentence. It grounds why the rule also applies to a tool that does not yet exist. Only the top level applies to everything; if a principle rests exclusively on a domain manifesto, strictly speaking it applies only in that domain.
- Derivation — another norm of the suite. It shows what the rule follows from, and it carries within the order, not outward.
- Proven in — a decision that has been taken and tested in a product. It is evidence, not grounding, and may be empty: an empty cell means "not yet tested anywhere" and is a piece of information, not an embarrassment.

A piece of evidence is not an anchoring.

It does not answer the question why a rule applies to a tool that was not built at the time. If the grounding of a principle stood only in this column, it was not a norm of the family, but a generalised individual experience — perhaps correct, but not tenable before a client.

Anchoring is not deduction.

What is required is a supporting ground, not a formal-logical derivation — the same rule as with the manifesto identifiers: an identifier is named when it supports, not when it merely fits. Proximity therefore differs; indirect anchorings are marked as such in the table.

An implementation strategy is not a derivation.

The question is not whether a value fits the rule, but whether the rule necessarily follows from it or is merely one possible way to it. Configurability supports self-determination, but self-determination could also be produced otherwise; that is why RA-07 is kept as a technical rule and not anchored.

The chain is not one-to-one. A manifesto sentence can carry several principles, one principle can honour several values, and a technical rule can be necessary without possessing moral content. A set of rules in which every rule without exception follows from a value is constructed after the fact and not clean.

Two principles therefore expressly carry no anchoring: RA-07 and RA-12. One does not yet carry it: RA-08 — the manifesto sentence for it is formulated but deferred until it is settled who may designate knowledge for sharing and what happens to the knowledge derived from it upon revocation.

3 · The thirteen principles

RA-01 *The core decides, the model proposes, the human takes responsibility*

Three levels that never coincide. The model produces proposals. The deterministic core decides what is valid — states, counts, review results, releasability. The human decides what becomes binding.

What is guaranteed is enforced deterministically, not requested at the prompt level.

A commitment that stands only in the prompt is complied with in reformulation. That is why the core downgrades a release when the prerequisites are missing, and language only explains it. Downgrades are one-sided: a stricter judgement is never softened.

> ***Verifiable:*** *No path exists on which a model result becomes binding without core review.*

RA-02 *No authoritative facts from the model*

The model may classify, synthesise, formulate, route and interpret. It may not produce citations, numbers, legal statements or items of evidence. These come from verifiable sources or from tools.

> ***Verifiable:*** *Every value with evidentiary character carries a stored source; if it is missing, it cannot be output. Counts in the result come from the core, not from the text.*

Since 14 August 2026 this principle is anchored at the top level: PH-10 — the observed and the presumed are two different things; what appears as fact names its origin, and origin means where something comes from, not who said it. Before that, RA-02 rested solely on a domain-bound manifesto sentence and thus, strictly speaking, applied only where new material is brought in from outside.

Note: on this formulation there is an open need for clarification with the test concept (Chapter 13, Finding 1). This norm uses the broader formulation, because the narrower one would exclude the substantive review by a model.

RA-03 *Every core object carries its governance mixin*

Core objects are the things a tool leads and takes responsibility for — an order, a radar entry, an assessment, a protocol, a process model. Each carries the same fields.

`created_at · updated_at · version · created_by · status`

Thereby every object is datable, versionable and attributable to an author — human or model. The history arises as a by-product: whoever stores versioned assessments gets the question "where has the assessment moved to?" for free.

> ***Verifiable:*** *No core object without a complete mixin; created_by distinguishes human and model.*

RA-04 *Draft and confirmation are separate fields*

Not only separate states: separate fields. The model draft is retained alongside the human final version.

Fig. 1 — Structure of a core object. The draft does not disappear.

Two reasons. First, the difference between draft and final version is the divergence log — from it one learns one's own assessment grid and refines the guidance. Second, a human reviews a reasoning, not a number; that is why the draft contains arguments and not merely values.

> ***Verifiable:*** *The confirmation field for the reviewing human is mandatory; without it, the object is not binding. The draft is never overwritten.*

RA-05 *Release is a gate, and a change invalidates it*

Fig. 2 — A pattern that recurs in every product.

This pattern occurs in the suite at four places, and it is meant to work the same everywhere: at the release of knowledge, at the release of an order, at the gate between internal validation and customer test, at the extension of a controlled vocabulary.

Releases have three forms, not two: released · released with condition · not releasable. The middle one is the practically most important — it makes it possible to proceed without losing an open point, because the condition is documented with responsibility and deadline.

> ***Verifiable:*** *A change to a released object resets the status; there is no way to keep a release and change the content.*

RA-06 *Changes are transitions, not breaks*

The other principles describe what a tool looks like at a point in time. This one describes how it may change over time. Without it, the other twelve have an expiry date.

Whoever changes a norm, a vocabulary or a schema changes the meaning of data already stored.

Without recorded transitions, learning is impossible — neither for an organisation nor for a human. Whoever silently changes the meaning of stored values no longer has a history, but only a holding that looks like a history. That is the quietest way to destroy knowledge: no test flags it, and no one notices.

Proven at the Technology Radar: when the assessment grid there is sharpened without the transition being recorded, the same grade means something different before and after the change. The history — the actual value of the holding — is then silently falsified. It still looks good and is no longer true.

Four rules keep the timeline clean:

- Every stored object knows under which version of the norm, the vocabulary and the assessment grid it came into being. That is the extension of the version triple from the methods to the data.
- Every change to a norm delivers a transition with it: the existing data still apply, or there is a mapping onto the new version, or they are expressly marked as historical. A silent reinterpretation is none of these three possibilities.
- The assessment grid itself is a versioned object, not knowledge in the head. Its history belongs to the evidence.
- An issued piece of evidence is immutable. A correction is a new version, not an editing of the old.

A break remains permitted — not everything can be kept compatible in the long run. But it is explained, dated and applied to the existing data. What is forbidden is only the unnoticed break.

> ***Verifiable:*** *For every stored object it can be determined under which version it came into being. No change to a controlled vocabulary without a transition rule for existing data. An issued piece of evidence cannot be changed, only replaced.*

Relation to RA-05: both protect something from unnoticed change — RA-05 the state of an object, RA-06 the meaning of its values. And RA-06 is the condition under which RA-12 does not become a trap: a controlled vocabulary that is extended without a transition reinterprets its own past.

RA-07 *Configuration before programming*

What differs per tenant, per product or per case is not programmed but configured — as a versioned file alongside the code, not in a database and not in a foreign system.

WHAT IS CONFIGURED	WHY
Methods and procedures	The substantive HOW changes faster than the code.

Catalogues and controlled vocabularies	Extension is a deliberate act, not a side effect.
Review rules and tolerances	They are matters of substantive discussion, not technical ones.
Test cases	Shared tool, product-specific cases.
Appearance per tenant	Logo and colours are configuration, not custom build.
Assignments: function to role, cost rates	They differ per organisation.

> ***Verifiable:*** *A tenant request from this list requires no code change. All configuration files are versioned with the code and diffable.*

RA-08 *Tenant separation is structure, not filter*

Fig. 3 — Three levels; inheritance downwards, never sideways.

A separation that rests on a filter holds only as long as no one forgets the filter. That is why every tenant-related object structurally carries its affiliation, and what is shared is expressly marked — not recognisable by the absence of a specification.

Three levels: the shared base without tenant, the tenant with its deltas, the instance or project with its values. Inheritance is downwards; an access across tenant boundaries does not exist — not as a path, not as an authorisation.

> ***Verifiable:*** *An access without tenant reference delivers exclusively shared holdings. There is no query that sees two tenants at once.*

Anchoring outstanding. The manifesto sentence for this principle is formulated — entrusted knowledge remains separate; only what is expressly intended for sharing is shared —, but deferred. As long as it is not settled who may designate something for sharing and what happens upon revocation to the knowledge derived from it, the manifesto would contain a sentence that ongoing operation breaks. Until then, RA-08 is the only principle without anchoring, and it is declared as such.

RA-09 *No direct model contact*

No product speaks directly with a model provider. All calls — including embeddings — run through the capability AI call, which maintains an adapter per provider and pre-connects pseudonymisation.

> *A call without pseudonymisation is not technically possible, not merely forbidden.*

From this two properties follow for the price of one: the protective layer cannot be circumvented because there is no second path. And a model change becomes an adapter question rather than a re-development.

> ***Verifiable:*** *In the code there is exactly one place with provider contact. The call record retains provider, model and model version.*

RA-10 *Every result carries its provenance*

Provenance is a by-product of execution, not a feat of memory of the model. Whoever has the origin written together at the end gets a plausible narrative instead of an evidence.

For each section of a result the following is kept: the source types used, the core statements with their origin, the follow-up questions posed together with the rejected ones, the assumptions made and what has been left open.

> ***Verifiable:*** *The provenance can be reconstructed from the execution data without asking a model. Every displayed value is traceable back to its origin.*

RA-11 *Uncertainty is a field, not a formulation*

A tool that dresses its uncertainty in words makes it unevaluable. The certainty grade and age of a statement are structured fields — confirmed, probable, unconfirmed — and appear in the interface, not only in the record.

Verifiable: *Statements with evaluative character carry a certainty grade; assessments carry their date and are marked as overdue when they age.*

RA-12 *Controlled vocabulary*

Classification takes place exclusively into existing lists. A model classifies; it does not extend. New terms need a deliberate release along the pattern of RA-05 and a transition for the existing data along RA-06.

Without this rule, dozens of categories arise within months and the evaluability is lost — the data look complete and carry nothing more.

Verifiable: *A value outside the list is rejected, not silently created.*

RA-13 *A tool produces evidence, not only results*

To the question "is this reviewed?" no word answers, but a document. Five pieces of evidence arise in this suite; which ones a tool leads depends on what it does.

EVIDENCE	WHAT IT RECORDS	WHO PRODUCES IT
Provenance	Origin of the statements of a result.	the core, at execution
Review protocol	Result of a substantive review of a result.	the reviewing method
Test protocol	Which test types ran with which result.	the test runner
Call record	Provider, model and version per model call.	the capability AI call
Divergence log	Difference between model draft and human final version.	the core, at confirmation

Verifiable: *Every piece of evidence carries the governance mixin and is readable without the producing tool.*

4 · The layered structure

Fig. 4 — Five layers and one calling rule.

LAYER	WHAT LIVES HERE	WHAT MAY NOT BE HERE
Surface	Presentation and interaction according to the operating norm.	Substantive rules, decisions, direct calls to capabilities.
Methods and procedures	The substantive HOW: procedures, rubrics, catalogues, vocabularies — configured.	Technical building blocks, state management, counts.
Core	Domain model, states, releases, rule review, counts, provenance — and the execution of the methods.	Substantive judgement without a norm; provider-specific code.
Capabilities	Technical abilities according to catalogue contract.	Substantive knowledge of any kind. A capability knows HOW, not WHAT is good.
Persistence	Storage with structural tenant separation.	Substantive logic, state transitions.

The execution layer belongs in the core.

What manages follow-up loops, states, handoffs, abort criteria and provenance is not part of a method, but executes it. From this follows the rule of thumb for the partition: deterministically verifiable criteria belong in the core, substantively assessable ones in the method. This layer is listed in the capability catalogue as workflow and not yet built — it is the largest open place of this architecture.

5 · Core objects and states

A tool leads a manageable number of core objects. They carry the governance mixin, run through the release cycle of P5 and are the anchors on which provenance and evidence hang.

Two patterns have proven themselves and are binding:

- The tracked thing and its evidence are separated. An entry has its own identity; dated observations dock onto it. Only in this way can movement over time be shown.
- The assessment is a separate, versioned object with timestamp — not a field on the assessed thing. The history thereby arises free of charge.

Verifiable: *For every core object it can be answered who brought it into which state when and on what it rests.*

6 · Surface: what the operating norm demands of the architecture

The operating norm stands in a separate document. For the architecture it is decisive that several of its principles can only be honoured if the data model carries them. They are therefore listed here as requirements and not left to the surface.

PRINCIPLE OF OPERATION	WHAT THE ARCHITECTURE MUST LEAD FOR THIS
The human decides, the system proposes	Draft and confirmation fields separated (RA-04); a state that distinguishes draft from binding.
No statement without source	Origin per value retrievable (RA-10); source specification as a mandatory field, not as a text convention.
Make uncertainty visible	Certainty grade and date as fields (RA-11);

	overdueness computable.
The same across the family	Same status vocabulary in all products — that is, in the core, not per surface (RA-03, RA-12).
Trust and dignity	Pseudonymous identifiers instead of clear names in the display; disclosure as a deliberate step according to RA-05.

Conversely: the concrete review values of the operating norm — minimum sizes, contrasts, colour never as sole signal — belong in the rule review and not in the developer's diligence.

7 · Configuration and extension by tenants

The suite does not earn its money by building something different for every customer, but by the fact that the same can be configured differently. That is an architectural requirement, not a sales question.

LEVEL	WHAT LIES HERE	WHO MAINTAINS IT
Base	Methods, catalogues, vocabularies, reference holdings, review rules.	product responsibility
Tenant	Deltas to the base: own procedures, assignments, cost rates, appearance, own review rules.	advisory jointly with the tenant
Instance	Values of a concrete case.	user

Rules: a tenant delta overrides the base, it does not replace it. A delta without an assignment to its counterpart in the base is invalid. And a delta of a tenant is not visible to other tenants — not even as a template.

8 · Operation and environments

Four environments with different configuration: development with mocked interfaces, internal test environment, integration environment as an image of production with real interfaces, and productive operation without test operation.

For the architecture, one thing above all follows from this: a tool must be able to run against mocked or real surrounding systems without a code change. The interface mode is configuration and is held in the evidence, so that no illusory security arises.

Promotion runs stepwise and is never skipped. Every product occupies its own port range on the shared host; tools share the operating stack, not their data.

> ***Verifiable:*** *Change of the interface mode requires no new deployment. No product accesses the data holdings of another.*

9 · What a tool must not do

The negative space is the most verifiable part of an architecture, because violations are concrete. Every prohibition carries an identifier so that a finding can invoke it, instead of speaking of "basic principles" in general. For these identifiers the same rule applies as for the principles: they are never redefined.

IDENTIFIER	A TOOL MUST NOT …	RELATED PRINCIPLE
NR-01	make a decision that belongs to a human — not by default setting, silent automation or a pre-filled truth either.	RA-01
NR-02	produce a citation, a number or an item of evidence that does not come from a source.	RA-02
NR-03	address a model directly.	RA-09
NR-04	cross a tenant boundary — reading, writing or as a template.	RA-08
NR-05	retain a released state while the content changes.	RA-05
NR-06	extend a controlled vocabulary of its own accord.	RA-12
NR-07	output a result without provenance.	RA-10
NR-08	secure a guarantee solely by instruction to a model.	RA-01

NR-09	change the meaning of stored values without recording the transition.	RA-06
NR-10	subsequently edit an issued piece of evidence.	RA-06, RA-13

A finding that invokes the negative space names the identifier.

Without an identifier it is an assertion. Invocation of the negative space weighs heavily in several procedures of the suite — it turns a reservation into a must-finding —, and what weighs so heavily must be addressable.

10 · Conformance

A tool architecture is conformant if it declares for each of the thirteen principles how it honours it — or grounds why it is not applicable. The place of this declaration is the conformity statement from the framework.

Because the principles carry identifiers, list 1 of the statement is a short table instead of an essay — per row an identifier, the implementation and the reference. List 2 leads the deviations with grounding, list 3 the places where this norm remains silent; the third list is the feedback to this document.

```
konformitaet:
  - { prinzip: RA-01, erfuellt: ja,    fundstelle: "Core/Release,
Section 4.2" }
  - { prinzip: RA-06, erfuellt: ja,    fundstelle: "Schema version
per record" }
  - { prinzip: RA-08, erfuellt: nein,  begruendung: "single-tenant,
deliberate", liste: 2 }
  - { prinzip: RA-12, erfuellt: offen, hinweis: "vocabulary not yet
closed", liste: 3 }
```

Thereby the statement is machine-verifiable: a conformity checker can determine whether a row exists for every identifier — long before anyone reads its content.

Same norm does not mean same behaviour — the release applies per model and per surface. An implementation is released only when it passes the mandatory invariants and the ground truth on the deployed model. The evidence retains model and version.

11 · How a tool arises from this norm

1. Record purpose and demarcation of the tool — what it does and what it expressly does not.
2. Name core objects: what does the tool lead, what is merely attribute?
3. For each core object, determine the release cycle and specify the confirmation fields.

4. Select capabilities from the catalogue; declare what is missing as a catalogue request.
5. Draw the configuration boundary: what is base, what tenant delta, what instance value?
6. Determine evidence: which of the five does this tool produce?
7. Determine the timeline: which versions does each core object carry along, and what happens to existing data when the vocabulary or the assessment grid changes?
8. Translate applicable manifesto sentences or note them as not applicable.
9. Prepare the conformity statement over all thirteen identifiers.

12 · Open points of this norm

POINT	STATUS
The execution layer is not built.	The core executes methods — until now every product does that in its own way. Largest open place.
A survey of the running products is missing.	This norm describes how one builds. How far the existing products correspond to it is not yet measured.
The status vocabulary is not unified.	RA-03 requires the same states across the family; the concrete values have grown product by product. The unification is itself a case for RA-06.
Existing data do not yet know their version.	RA-06 requires that every stored object carries along its norm, vocabulary and grid version. In the running products this is only partly the case — the retrofit itself needs a transition.
Authorisations are not regulated.	This version regulates the tenant separation, not roles and rights within a tenant.
Relation to existing architecture decisions.	Existing decisions are worked in here, but not yet formally traced back to this norm.
RA-08 carries no anchoring yet.	The manifesto sentence is formulated and deferred until consent to sharing and revocation are regulated. Until then declared, not concealed.
Provenance is not transitive for derived knowledge.	RA-10 keeps one specification per value. If from A and B a C arises and from that an E, after a revocation of A it cannot be determined that E depends on it. Conceptually this requires a provenance graph. Noted, not built.

13 · Findings when deriving

FINDING	CONSEQUENCE
RA-08 has no manifesto sentence.	Done on 14/08/2026: the sentence is formulated — entrusted knowledge remains separate — and deferred until consent and revocation are regulated. RA-08 is until then kept without anchoring.
RA-06 has no manifesto sentence either.	Done: anchored in PH-08 (remembering as an enduring ability), expressly marked as indirect. The grounding has at the same time been formulated in a product-neutral way.
The fact boundary is formulated twice differently.	This norm uses the broader formulation. PH-10 now anchors the principle at the top level; the contradiction with the test concept is unaffected by this, and RA-02 remains under reservation in this respect.
Three concepts of evidence were not delimited.	Chapter 3.12 delimits them. They remain separate but carry the same governance mixin.
The operating norm places requirements on the data model.	Chapter 6 makes them explicit. Without this translation, several operating principles would not be honourable, but merely asserted.
The question of anchoring had only been asked twice.	When reviewing all thirteen identifiers it emerged: five anchored, two only domain-bound, six without an answer. After the decisions of 14/08/2026 it is ten anchored, two expressly technical, one outstanding.
Four of five manifestos are product manifestos.	Only the top level applies to everything. As long as the domain manifestos are cut according to tools, every principle resting on them inherits their domain boundary. The product-neutral formulations already lie in the identifier register, in the source documents not yet.