MANUAL

IKINGAI™

Metadata Governance Manual

Consent-based and semantic workflows,
content access control, and data protection
in AI-assisted work

Execute Your Intentions, LLC

What this manual covers

How to govern a body of work that an AI system can read, route, and act on: what the machine may touch, what it may never touch, how content is classified before it moves, how access is granted and revoked, and how a decision made inside the system stays traceable afterward.

It is written for anyone who has pointed an AI at their own material and realized nothing was stopping it.

1. The failure this prevents

An AI system given access to a body of work will do three things by default: read everything, write anywhere, and remember nothing about why.

The result is not usually dramatic. It is a file quietly overwritten, a private note pulled into a public draft, a claim asserted as verified that nobody checked, a source deleted after its value was assumed captured. Each is small. They compound, and the record stops being trustworthy at a point nobody can identify afterward.

The fix is not more caution from the operator. It is structure the system carries itself.

Four properties make a body of work governable:

- Every object declares what it is and who may act on it.
- Action is gated on consent, not on capability.
- Nothing is destroyed before its value is captured.
- Every claim carries the check that proves it.

The rest of this manual is how to get those four.

2. Consent before action

The governing rule is that capability is not permission. An AI system that *can* write to a file must still be told, per class of file, whether it *may*.

Set the default to read-only. Then grant outward in three tiers.

Allowed by default. Listing, reading, searching, summarizing, extracting candidate meaning, recommending where something should go, drafting a proposed change. All of these produce output without altering the record.

Allowed only if separately configured. Creating new artifacts in named locations. Capturing raw material in the intake folder. Writing drafts to an approved draft location. Each of these adds to the record without changing what is already there.

Not allowed without explicit approval, delivered as a proposed patch. Overwriting, renaming, moving, modifying governance files, modifying canonical files, promoting anything into an authoritative state. These change what the record says.

Never allowed at all. Deleting. Emptying a deletion holding area. Moving material into one. Deletion sits outside the tool path entirely and is performed by a person, by hand.

That last tier is what makes the rest safe to relax. A system that cannot delete can be given generous read access without the operator watching it.

Ethical decision-making as an enforced process

Consent gates decide whether an action may proceed. Something has to decide what the gate is protecting.

Most organizations write their values into a document and check against them never. The IKINGAI™ approach codifies them as rules the system evaluates at the moment of action. Values become law in the operational sense: a declaration that changes what the machine will do, not a statement of aspiration.

Three properties make an ethical decision-making process real rather than declared:

The rule is machine-readable. "Private material is never used in public output" is a policy. A `vault` flag the drafting process checks before including anything is a rule. Only the second one holds when nobody is watching.

A violation stops the action rather than logging it. A process that records that it did something prohibited has not been governed. It has been observed.

The values are versioned like anything else. They change deliberately, with a date and a reason, and the change is reviewable. A value that can be quietly edited is not governing anything.

The practical form is a short document of codified rules, each expressed as a condition the system can evaluate, each linked to the metadata field it checks. Ten enforced rules govern

more than a hundred stated principles.

3. Metadata is the governance layer

Governance rules that live in a document get read once. Governance rules that live in metadata get enforced on every access.

Attach a semantic metadata object to every governed item. Semantic means the tags describe what the object *means* and what it is *for*, not what format it is in. A tag that says "pdf" is filing. A tag that says "supersedes the prior pricing decision" is governance.

At minimum the object declares four things:

```
{
  "tags": ["#loop", "#reflection"],
  "execution_rights": "cleo",
  "linked_files": ["goal.md", "status.md"],
  "vault": true
}
```

tags classify the object so routing can be automatic rather than manual.

execution_rights names which agent or role may act on it. Not who may read it. Who may *act*.

linked_files declares the object's relationships, so the system can traverse from any object to its neighbours.

vault flags material that never leaves, regardless of what a downstream process requests.

The schema matters less than the discipline. What matters is that the rule travels with the object rather than sitting in a policy document the machine never opens.

Write the metadata before the object moves, not after. Metadata applied retroactively describes where a file ended up. Metadata applied at intake decides where it goes.

4. Defining data relationships

Two rules keep a body of work navigable as it grows.

One concept, one home. No fact is authoritative in two places. Everything else references it. When the same claim appears in two files, that is a defect to reconcile, not a state to maintain: one home wins and the other becomes a pointer. Duplication is how a record starts disagreeing

with itself.

Every file links to its neighbours and back to the index. Relationships are declared, not inferred. An AI system traversing declared links behaves predictably. One inferring relationships from similarity does not.

The practical test: pick any object at random. Can the system name what governs it, what it supersedes, and what it belongs to, without searching? If not, the relationships are not defined; they are guessed.

5. Controlling access to content

Access control in AI-assisted work has to answer a question ordinary file permissions do not: not just who may open this, but what may be *derived* from it and where that derivative may go.

Three mechanisms, in order of strength.

Protected areas. Name the paths that may never be written directly, only through a reviewed change. Identity material, authoritative reference, external-facing standards, and the frozen archive belong here. The list is short and explicit. A path is protected or it is not; there is no partial state.

Execution rights per object. The metadata field above. Read access and action rights are separate grants. Most material can be widely readable and narrowly actionable, and that combination is usually what is wanted.

The vault flag. Some material is readable by the operator and by nothing else. It is not classified by sensitivity level, which invites judgment calls at the moment of retrieval. It carries a flag that downstream processes check before including it in anything.

Retrieval is not approval. This is the rule that gets violated most. An AI system that successfully retrieved a document has demonstrated that it could reach it, and nothing more. Whether the content may be used in the output being drafted is a separate decision, made against the object's own declarations.

6. Content governance in an autonomous system

An autonomous system acts without a person in the loop for each step. Governance is what makes that acceptable rather than reckless.

Route semantically, not by pattern. Every incoming item is read for meaning and intent before it is filed. Pattern rules match on filename, extension, and folder, and they file things that look correct and are wrong. Semantic routing determines what a thing *is* and what it is *for*, then sends it to the one home that owns that concept. This is the difference between a filing system and a governance system.

Consent-based workflows gate on the item, not the actor. A trusted process still stops at an object whose declarations do not permit the action. Trust is granted per object, per action, and it is checked at the moment of the action rather than at the start of the session.

Separate content from system changes. Incoming material is one of two kinds. Content is a fact, insight, or reference: route it to its home. A system change alters how the system itself works, a new rule, a new location, a new behavior, a naming change. System changes go through a different path with a higher bar, because they change everything downstream of them.

Draft first, always. Material compounds into its home automatically, as a draft. It becomes authoritative only when a person acknowledges it. The gate is a review of a digest, not approval of each item, which keeps the human in control without making them the bottleneck.

Some categories always require an explicit yes. Anything touching public positioning, external voice, pricing, or commercial terms does not lock on silence. Silence approves the routine. It never approves the consequential.

7. Data protection

Local by default. Material lives on storage the operator controls. Nothing is pushed to a remote by default. Remotes are granted per surface, deliberately, for the specific surfaces that require them.

Know what each tool does with an input. Any tool in the path that trains on inputs by default receives no governed material unless a documented opt-out is confirmed engaged. A tool with no configurable opt-out receives none at all. This is checked before the tool enters the workflow, not after material has passed through it.

The most-restrictive link governs the whole chain. When an artifact passes through more than one tool before it ships, every link must permit the intended use. If one link claims rights over outputs or requires training use of inputs, the composite carries that limit no matter how permissive the other links were. A tool whose terms have not been read counts as the most restrictive link until they are.

Never destroy before value is captured. Extract what a source is worth into its home first. Preserve the source. Only then may anything recommend it for review. The recommendation is the ceiling of what an automated process may do.

8. Provenance and evidence

Every authoritative claim carries where it came from and how confident it is. A recorded gap beats false certainty. "Unknown" is a usable input to a decision. A confident wrong answer is not.

Verified means checkable. The words *verified*, *done*, *resolved*, and *confirmed* may not be written about anything unless the claim carries the check that proves it: a command to re-run, a file and line, a commit, or a link to the source. A claim that cannot be reduced to a re-runnable check is written as *reported*.

This rule exists because of a specific failure mode. A process asserts something is verified. The next process trusts the word instead of the source. The falsehood propagates and is now load-bearing. Requiring the check breaks the chain at the first link.

Point, do not restate. An index that restates a fact can go stale and false. An index that points at where the fact lives cannot. Prefer "run this to check" over "this is confirmed."

When a claim is wrong, replace it. Delete it and write what is true. Do not annotate it, quote the old wording, or leave it standing under a correction header. Version control holds the history. Superseded text reads as live to anything scanning the file, so each correction layer adds surface for the next misread.

9. The outbound boundary

Intake governs what comes in. Something has to govern what goes out.

Before any derived material leaves the system, declare five things:

1. **What crosses.** The specific item, named. Nothing implicit.
2. **What never crosses.** Identity material, raw authoritative internals, internal standards, and anything flagged as held. The gate refuses these by default rather than on request.
3. **Provenance.** The item states what it derived from and that it is a derived view, not the source. Traceable back, never mistaken for the original.

4. **Scope boundary.** What the recipient may rely on it for, and what they may not.
5. **Approval.** Anything carrying positioning, voice, or commercial terms needs an explicit yes before it ships.

Then log it. Date, recipient role rather than personal data, what went, what it derived from, how it physically left, and whether approval was given. The log is the answer to "what has left, and to whom," which is a question that becomes unanswerable fast without one.

10. Putting it in place

A working order that does not require rebuilding anything first.

1. **Write the protected list.** The paths nothing may write directly. Do this before granting any access, because it is the boundary everything else is defined against.
2. **Set the default to read-only.** Grant outward from there, tier by tier, as specific needs appear.
3. **Close the deletion path.** No automated process deletes. Deletion is manual and outside the tool path. This one change makes generous read access safe.
4. **Add metadata at intake.** Tags, execution rights, links, vault flag. Applied when material arrives, not later.
5. **Route by meaning, draft first.** Classify content against system change. Everything lands as a draft.
6. **Require the check on every claim of verification.** This is a writing rule, and it is the one that keeps the record honest as it grows.
7. **Build the outbound gate last**, once there is something worth sending.

Nothing here requires a particular tool. It requires that the rules live where the machine reads them, and that the boundary is declared before the access is granted.

Published by Execute Your Intentions, LLC. Version 1.0, July 2026. executeyourintentions.com