AEGIS User Guide

Build regulator-ready evidence from the DITA source

Contents

Chapter 1: What AEGIS does..... 5

Chapter 2: Generate the authority and downstream packs..... 7

Chapter 3: Pick a reader and read the checks..... 9
Read a requirement and its evidence..... 10

Chapter 4: Propose a link with Gemini and review it..... 11
Record a decision as the compliance lead..... 12
Try the leak test..... 12

Chapter 5: Build and download a pack in the sandbox..... 13
Release and seal a pack..... 14

Chapter 6: Ask the evidence corpus..... 15

Appendix A: The four checks..... 17
Metrics from the design..... 18
Pack contents and the manifest..... 18

Appendix B: Problems and what to do..... 21

Appendix C: Glossary..... 23

Appendix D: Release notes: AEGIS 0.1.0..... 25

Chapter 1

What AEGIS does

AEGIS builds an evidence pack from a DITA source and proves that each requirement is covered, current and approved, or names it as a gap.

Regulators, downstream providers and compliance leads read model documentation as evidence. They need each statement with its version, its source and the person who approved it. A DITA source in Git already holds all three. AEGIS reads the Gemma evidence set in dita-corpus: requirement topics, the relationship table that links each requirement to its evidence, each topic's prolog dates and keys, and the pull request that approved each file.

Four checks decide whether a pack can be released. They are plain code and give the same answer every time. One tool calls a model: when a requirement has no evidence, Gemini 3.8 Flash proposes a link, and a person approves or rejects it. The model never approves a link, releases a pack or edits a topic.

Important: AEGIS is a documentation tool. It isn't legal advice, and it doesn't decide whether an obligation applies or whether evidence is legally enough.

The quickest way to see it work is the AEGIS generate page. Pick a system and it builds the authority pack for the AI Office and a downstream pack for providers who build on the model, then measures both against the design's metrics. See Generate the authority and downstream packs on page 7.

On the AEGIS sandbox you act in a private sandbox. Your links, decisions and planted leaks are stored for 7 days and never change the source.

Chapter 2

Generate the authority and downstream packs

Pick an AI system and AEGIS builds two evidence packs from its DITA source, then measures them.

Start on the AEGIS generate page. The page runs the four checks and builds both packs in one step. You don't need an account, and nothing you do changes the source.

1. Under **Pick a system**, choose a system.

   Only a system with an evidence set can be chosen. Today that is Gemma 4. A system without one is listed but can't be selected.
2. Select **Generate packs**.

   The first run after a quiet period can take about 15 seconds while the service starts. Later runs take about a second.
3. Read each pack card.

   The **Authority pack** is for the AI Office (Annex XI, Section 1). The **Downstream pack** is for providers who build on the model (Annex XII). Each card shows a status, the four check tiles (coverage, currency, approval, consistency), the requirements that have no evidence, and its findings.
4. Select **Download** on a card to get that pack as a zip.

   The card shows the manifest hash before you download. The download carries the same hash.
5. Read the **Metrics from the design** table under the packs.

   Each row says whether the number was measured on this run, modeled in the design only, or is not part of the MVP. See Metrics from the design on page 18.

You have two zips. Each holds the DITA-OT HTML5 and PDF output built with that reader's profile, a manifest, a report, a coverage table and SHA-256 hashes. These are demonstration packs: they are hashed, but never released or sealed.

To act on a gap, such as asking Gemini 3.8 Flash to propose a link or recording a decision, use the AEGIS sandbox.

Chapter 3

Pick a reader and read the checks

Topics:

- Read a requirement and its evidence

Each reader gets a pack built from the same source with its own DITAVAL profile and requirement set.

1. Pick **AI Office**, **Downstream provider** or **Public**.

   The AI Office gets the authority pack (Annex XI, Section 1). A downstream provider gets the downstream pack (Annex XII). The public gets the model card, checked against its own sections.
2. Read the status badge and the four check tiles.

   Each tile shows how many items passed out of how many were checked.
3. Read the source line under the badge.

   It names the corpus commit, the DITAVAL profile, the model version date and the number of topics in the build.

The status is **Ready to release**, **Ready, with recorded decisions**, **Needs decisions** or **Release blocked**.

Read a requirement and its evidence

Each row traces one requirement to the topics that answer it, with the commit, date and approver of each.

1. Find the requirement in the left column.

   Select its ID to open the requirement topic in dita-corpus.

2. Read each evidence topic on the right.

   The badge is the coverage value from the relationship table: stated, partly, linked or conflicting. Then the topic's revised date, the commit that last changed it, and the code owner who approved its pull request.

3. Look for rows marked **gap**.

   A gap is a requirement with no evidence topic for this model version. You can propose a link or record a decision.

Chapter 4

Propose a link with Gemini and review it

Topics:

- Record a decision as the compliance lead
- Try the leak test

For a requirement with no evidence, Gemini 3.8 Flash proposes one topic, and you approve or reject it as the information architect.

The model gets only the requirement text and the titles and short descriptions of the candidate topics, never their bodies. Its answer must name a topic that exists in this pack, or `none`. The public sandbox allows 20 proposals a day.

1. In a gap row, select **Ask Gemini to propose a link**.
2. Read the proposal: the topic, the reason and the confidence.

   A proposal under 0.6, or `none`, becomes a gap note instead.
3. Select **Approve as information architect** or **Reject**.

An approved link counts for `check_coverage` at once. It is labelled as a sandbox link in the table, the manifest and the report. In the real flow it would be a pull request to the evidence map.

Record a decision as the compliance lead

A gap, a stale topic, a missing approval or a conflicting statement can ship with a recorded decision and a reason. A leak can't.

1. Select **Record a decision** on a finding, or **Record one decision for all** for a check.
2. Choose **Release with this recorded** or **Hold the pack**.
3. Type a reason, then select **Save as compliance lead**.

   A reason is required. It goes into the manifest with the date.

When every finding has a decision and none blocks, the status becomes **Ready, with recorded decisions**.

Try the leak test

Put an authority-only key into an evidence topic in your sandbox, and watch `check_consistency` block the public and downstream packs.

The authority-only keys hold placeholders, not real figures. In a real source they would hold facts that only the AI Office may receive, such as training compute.

1. Under **Leak test**, choose a topic and an authority-only key.
2. Select **Plant the key**.
3. Pick the **Public** or **Downstream provider** reader.

   The status becomes **Release blocked** and the build is turned off. The authority pack isn't blocked, because it may contain authority-only keys.
4. Select **Remove it** to end the test.

Chapter 5

Build and download a pack in the sandbox

Topics:

- Release and seal a pack

The pack is a zip with the DITA-OT output, the manifest, a report, a coverage table and SHA-256 hashes.

1. Select **Build the pack**.

   The button is off while a leak blocks release.
2. Open the zip and read `report.html`.
3. Check the hashes: run `sha256sum -c SHA256SUMS` in the pack folder.

The `documentation` folder holds the HTML5 and PDF output that DITA-OT built in CI from the corpus commit with this reader's DITAVAL profile. A sandbox pack is hashed but never sealed.

Release and seal a pack

Only a code owner of the AEGIS repository can release a pack. Release seals it in Cloud Storage and writes a register row.

1. Open an Evidence request issue and choose the reader.

   A workflow runs the four checks, builds the pack with DITA-OT and posts the report on the issue.
2. Review the report.
3. As the compliance lead, comment `/release` on the issue.

   The workflow refuses a release from anyone who isn't a code owner, and refuses a blocked pack.

The pack is uploaded to the evidence bucket with a retention policy, its manifest records who released it, and the register shows it on the sandbox page. The proof of concept uses a short, unlocked retention policy, because a locked policy can't be removed or shortened.

Chapter 6

Ask the evidence corpus

Ask a question in plain words. Each point of the answer cites a topic with its commit, date, approver and the requirements it answers.

1. On the Ask page, type a question or pick a sample, then select **Ask**.
2. Read each point and its quote.

   The quote is copied word for word from the cited topic. A point whose quote or numbers aren't in the topic is dropped and listed.
3. Read the sources.

   An evidence topic lists the requirements it answers in the trace. A requirement topic lists its evidence, or says it is a gap.

Appendix
A

The four checks

Topics:

- Metrics from the design
- Pack contents and the manifest

What each check reads, when it passes and what it writes.

Check	Passes when	Reads
`check_coverage`	Every requirement has at least one evidence topic that survives the pack's profile.	The relationship table in `gemma.ditamap` and the DITAVAL profile.
`check_currency`	Each evidence topic was revised on or after its model version date, or holds only key references whose keys changed in an approved commit after that date.	The prolog `<critdates>` and `model-version`, and the version dates in `packs/models.yaml`.
`check_approval`	The last change to each topic merged through a pull request that a code owner approved and didn't author. The agent's account never counts.	The approval index built from Git and GitHub at build time, and `CODEOWNERS` in dita-corpus.
`check_consistency`	Every value resolves from a key, no topic states a fact two ways, and no authority-only key resolves in a public or downstream build.	Key references, key definitions and their audiences, and coverage values marked conflicting.

What can take a decision

A coverage gap, a stale topic, a missing approval, a literal value and a conflicting statement can each ship with a recorded decision. A leak and an unresolved key can't: they block release until the source is fixed.

Metrics from the design

What each metric on the generate page counts, how it is obtained, and what the design aims for.

The two packs are measured together. A **measured** number comes from this run. A **modeled** number appears in the design only. **Not in the MVP** means the capability isn't built into this page.

Metric	What it counts	Design target	Basis
Requirements covered, current and approved	Requirements, across both packs, that have evidence, whose evidence topics are current for the model version, and whose last change was approved in Git.	100%, or a recorded decision for each gap	Measured
Evidence items with a named human approver from Git	Evidence items whose last commit merged through a pull request approved by a code owner who isn't the author or an agent.	100%	Measured
Values in the packs that resolve from a key	Key references that resolve, checked by `check_consistency`.	100%	Measured
Authority content found in downstream output	Matches for authority-only topic ids, titles and key values in the built downstream HTML. The same search runs on the authority pack as a control, so a result of 0 means the search can see the content.	0	Measured
Time from request to a pack ready for release	Seconds for the checks and both pack builds on the server. A person still has to review the findings.	Under 1 working day	Measured
People hours per pack	Not measured. The design models under 4 hours against 32 by hand.	Under 4	Modeled
Packs sealed with a locked policy and a manifest hash	Packs here carry a manifest hash but are never sealed. Sealing needs a human release and a locked bucket.	100%	Not in the MVP

Result labels

Meets target	The measured value reaches the design target.
Open	The value is short of the target. The row's note says why, for example gaps with no recorded decision.
Fails	The value is worse than allowed. For the leak metric, any match is a failure.

Pack contents and the manifest

The files in a pack, and the fields of `manifest.json`.

Item	What it holds
`documentation/`	DITA-OT 4.4 HTML5 and PDF for this reader's profile.
`manifest.json`	The request, the source commit and profile, the status, each check's result, one item per requirement with topic, revised date, commit, approver and pull request, every

Item	What it holds
	finding with its decision, and a SHA-256 hash for every file.
`report.html`	The same content for a person to read.
`coverage.csv`	One row per requirement and evidence topic.
`SHA256SUMS`	Hashes of every file, including the manifest, in the format `sha256sum -c` reads.

Labels

A link that the model proposed carries `proposed_by: propose_mapping` and the person who approved it. A sandbox pack carries `sandbox: true` and is never sealed.

Appendix
B

Problems and what to do

Messages you may see on the AEGIS pages.	
The page says the service couldn't be reached, or the first action is slow.	
The service scales to zero between visits and is starting.	
Wait about 15 seconds and try again.	
The sandbox has used its 20 proposals for today.	
Record a decision for the gap instead, or try again tomorrow.	
Every topic shows **not approved**.	
Read the reason on each finding.	
A topic whose last change was pushed directly, not merged through a reviewed pull request, fails `check_approval` until a reviewed change merges.	
On the generate page, **Approvals read 0** appears and the approval metrics show 0.	
Merge the review pull requests in dita-corpus.	
AEGIS trusts Git for approval. Until a change to each evidence topic merges through a pull request approved by a code owner, no topic counts as approved. The page reports this honestly. It isn't an error.	
The metric for authority content in downstream output says **not measured**.	
Check that the service has the DITA-OT output.	
The scan needs the built HTML for both packs. Without it the metric can't run. It also isn't reported as a pass unless the control search on the authority pack finds the content.	
After a download, the message says the manifest hash differs from the one shown.	
Select **Generate packs** again, then download.	
This can happen if the service restarted onto a new corpus commit between the two steps.	

Appendix
C

Glossary

Terms used on the AEGIS pages.

Evidence pack	The files built for one reader: documentation, manifest, report and hashes.
Authority pack	The pack for the AI Office and national authorities: Annex XI, Section 1.
Downstream pack	The pack for providers who build on the model: Annex XII.
Manifest hash	The SHA-256 hash of `manifest.json`. The same hash is shown on the page and in the download.
Requirement topic	A DITA reference topic that states one requirement, its legal source and a key such as `aia.xi-2d`.
Evidence topic	A documentation topic, such as a model card section, that answers a requirement.
Relationship table	The DITA `<reltable>` that links each requirement to its evidence.
DITAVAL profile	A file that keeps or removes content by `@audience` and `@product` at build time.
Gap	A requirement with no evidence topic in the pack.
Leak	Authority-only content in a build for a reader who must not get it.
Sandbox	Your private overlay of links, decisions and tests, kept for 7 days. It never changes the source.

Appendix
D

Release notes: AEGIS 0.1.0

The first release that anyone can try, with a one-step page to generate and measure two packs.

- The generate page: pick a system, get the authority pack and a downstream pack, and see the design's metrics measured on the run.
- Three readers from one source: authority, downstream and public packs.
- The four checks, with tests for each, including the leak test.
- A private sandbox: propose a link, approve or reject it, record decisions, plant a leak, simulate a model update.
- Pack download with the DITA-OT output, manifest, report, coverage table and SHA-256 hashes.
- Release and seal by issue comment, for a code owner.
- The Ask page: answers from the evidence corpus with checked quotes.

Not in this release

Uploading your own requirement set, the NIST AI RMF pack, a BigQuery register and dashboards.