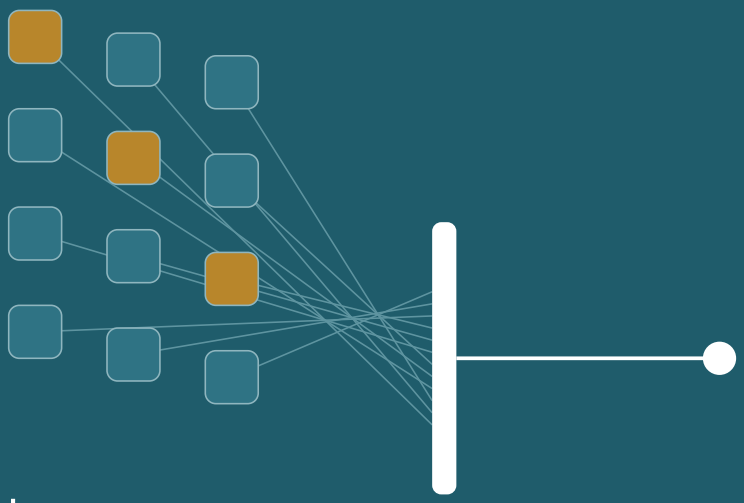
ZERYON

Report format · ZERYON Systems



AXYON Fleet
Whitepaper

Machine identity for agent fleets

Product	**AXYON Fleet**
Product version	**0.1.0 · core state v1.0.1 (983d6e6)**
Edition	**2.0**
As of	**2026-11-16**
Document ID	**AXYON-FLEET-0.1.0-WP-EN-2.0**
Vendor	**ZERYON Systems · Predrag Gasic**
Produced with	**reportlab · ZERYON Berichtsformat**

Scope
This paper describes AXYON Fleet version 0.1.0 as built and accepted. Extensions that are decided but not built are named as such and not described as features. It is neither an offer nor a proof of conformity.

Summary

The first AI agent in a company is a special case: someone knows it, someone knows what it is allowed to do. Once there are a handful of agents, something tips, and it does so quietly. Processes run that nobody started by hand, and several agents share service accounts because that was quicker at the time. The question "who did that?" then usually has only one honest answer: *something that had the right key*.

AXYON Fleet is a separate product alongside AXYON: the same control and evidence layer between AI and enterprise systems, for the situation in which several autonomous agents have to **identify themselves** to that layer. The agent identifies itself with a certificate, not with a secret. The access token is bound to the certificate of the connection it arrives on. The credential can be withdrawn at runtime — revocation is checked on **every request**. Identity events are written to the same append-only, hash-chained record as approval and execution. And approval stays human: an agent never approves an agent. That is a tested acceptance condition of the product, not a setting.

This white paper describes the situation, the mechanism, the architecture, the evidence runs and the limits. All statements were checked against the product state of 16 November 2026; the references are in the appendix.

01

The situation: from one agent to a fleet

In 2026, AI agents are no longer introduced one at a time. They appear in business departments, in automations, in pilot projects, and they stay. What grows faster than any other number in IT is the count of **non-human identities**: service accounts, API keys, automation users, agents. The industry term is *Non-Human Identities* (NHI).

For the human workforce the problem has been solved for decades: badges are issued, belong to exactly one person, and are collected when that person leaves — the gatehouse has the block list the same day. For agents, most organisations do something else: the "badge" is a secret. A token, a key, a certificate in a file.

A secret has three unpleasant properties:

	Secret token, key, certificate in a file	**Credential** issued, bound, revocable
Which question is answered?	“Do you know the word?” Whoever knows it gets in.	“Who are you?” Identity comes from an issued certificate.
Who is an action attributed to?	Several agents share one account. The originator stays unknown.	One certificate per machine; its fingerprint is in the record.
How is the right withdrawn?	Valid until expiry, if there is one.	Revocation at runtime; effective on the next request.

Figure 1 · Secret and credential compared

It answers the wrong question. Not *who are you*, only *do you know the word*. Whoever knows it gets in, and copying leaves no trace.

It is practically not attributable. Because managing secrets is tedious, several agents end up on the same account in practice. The question of who originated an action then cannot be answered at all.

It can hardly be withdrawn. A staff badge is collected. A key file is valid until its expiry date, if it has one. And unlike an employee who leaves, an agent that has gone astray has no interest in anyone noticing.

The debate about AI agents is almost always about reliability: is the model good enough? With several agents that is the **second** question. The first is less romantic:

> **Can you say afterwards which agent triggered which action — and can you take away its right to do so while the system is running?**

As long as the answer is "no", the best model does not help. You cannot let the agent do more, because you cannot attribute the result to anyone. And you cannot stop it, because there is no register it is listed in.

02
The requirement: issued, bound, revocable — and approved by a human

The difference between a password and a staff badge is not security. It is **manageability**. A badge is issued, belongs to exactly one holder, and is checked against a block list at every pass. If it is on the list, the turnstile does not open on the next pass.

Applied to machines, this gives four requirements:

1. **Issued:** identity comes from an issued certificate, not from a statement the agent makes about itself.
2. **Bound:** an access token only carries weight together with the certificate of the connection it is presented on. A token taken elsewhere does not.
3. **Revocable:** revocation takes effect while the system is running, on the next request — not at the next restart and not only at the expiry date.
4. **Approved by a human:** even with perfect machine identity, a machine does not approve a machine. Otherwise machine identity would only be a more elegant way of approving oneself.

The fourth requirement is regularly forgotten in identity discussions. It is the point at which an identity system becomes a **governance** layer.

03
What AXYON Fleet is

AXYON Fleet is a separate second product alongside AXYON — not an upgrade level, not a tier, not an add-on module. Both products share the same core source code, which AXYON Fleet consumes **pinned** to an accepted state; the dividing line is the deployment situation, not the price. Anyone running a single agent or a single-seat scenario does not need AXYON Fleet and loses nothing by that.

What AXYON does applies unchanged in AXYON Fleet, under identical governance rules: every AI action is checked, approved by a human where required, and recorded in an append-only, hash-chained chain **before** the reply is sent. AXYON Fleet adds machine identity — for the situation in which the counterpart of the control layer is not a single application but a fleet.



Figure 2 · AXYON Fleet between the agent fleet and enterprise systems

AXYON Fleet is delivered as a container appliance for your own data centre. There is deliberately no notarised macOS shell for AXYON Fleet, as AXYON additionally offers one: an agent fleet is not a single-seat scenario. AXYON Fleet is purchased, not subscribed to.

04
The mechanism in detail

4.1 A credential instead of a password

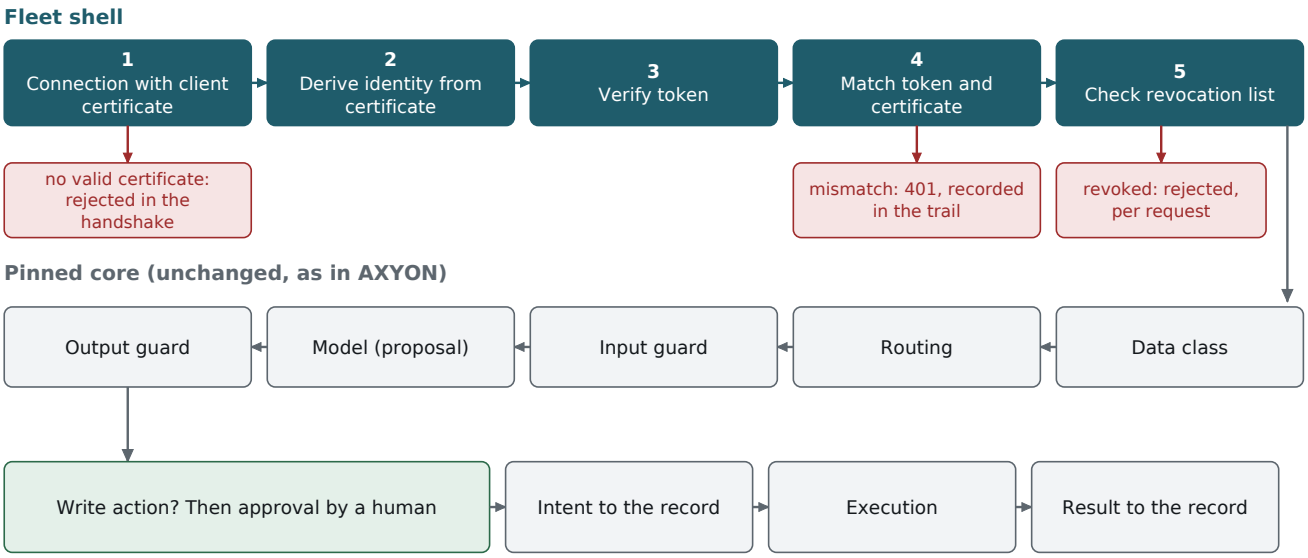


Figure 3 · Path of a request: five steps in the Fleet shell, then the pinned core

Every connection to AXYON Fleet is verified on both sides (mTLS). The agent presents a client certificate that is checked against the certificate root set by the operator; without a valid certificate no connection is established — rejection happens in the handshake, before a single request reaches the product.

The **agent's identity comes from the certificate itself**: from a name field (SAN URI in the delivery configuration; alternatively SAN DNS or CN, set by the operator). It is not a statement the agent makes about itself and not a header set by an upstream component. TLS termination at a reverse proxy that passes the identity on in a header was explicitly rejected in the architecture decision: where identity comes from a header, it is a claim again.

4.2 A token alone is no longer enough

Access tokens remain necessary — they carry roles, tenants and permissions. But in AXYON Fleet the token is **bound** to the certificate of the connection it is presented on: the identifier in the token must match the identifier in the certificate, and the certificate is authoritative. A mismatch leads to rejection.

The consequence is the one that matters: **a token that was taken and is presented elsewhere does not work.** The classic theft of a bearer token — from a log, an environment variable, a compromised process — loses its value, because the thief lacks the bound certificate and its private key.

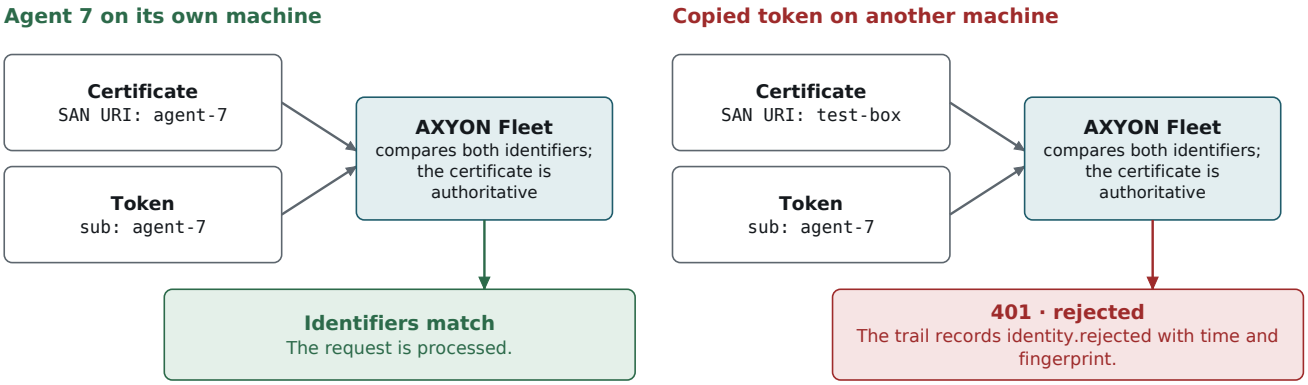


Figure 4 · Binding of token and certificate: a copied token does not work

This binding can be switched off, but switching it off is a documented change to the commitment made to third parties — not a silent operating option.

4.3 Revocation at runtime, on every request

The difference between a certificate file and a machine identity is revocation. Without it, a compromised credential is valid until its expiry date.

AXYON Fleet checks the revocation list **on every request** — not at connection setup, not at restart. A revoked agent is rejected on its next request, even on a connection that is already open. Withdrawing a revocation is equally possible and equally recorded; revocation and its withdrawal are deliberately secured asymmetrically, so that withdrawal of rights is fast and reinstatement is a conscious act. Administration runs through dedicated routes that are subject to the same authentication boundary as everything else.

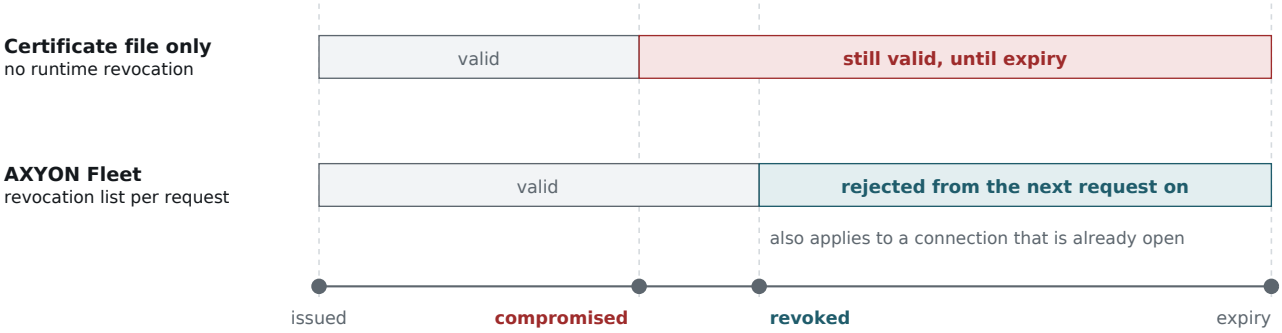


Figure 5 · Effect of revocation over time

4.4 Identity is in the same record as the action

Identified, rejected, revoked — these are entries in the **same** chain in which approval and execution are recorded. There is no second log beside it that would have to be evaluated as well, and no format an auditor has to learn anew: an AXYON Fleet record is the same chain known from AXYON and is recalculated with the same tool.

For the chain itself, what applies to AXYON applies here: every routing decision, every approval, every executed action and every rejection is written before the reply is sent — append-only and hash-chained. A later change to an entry shows up when the chain is recalculated. In the delivery configuration the trail is encrypted at rest (AES-256-GCM per entry, Argon2id key derivation); the operating compose file refuses to start without a key. The chain heads are signed with an Ed25519 key and kept in an anchor file, and an auditor can check the chain with `axyon-verify` — for identical content and for nothing having been cut off at the end. Without a running AXYON Fleet and without the operator's key.

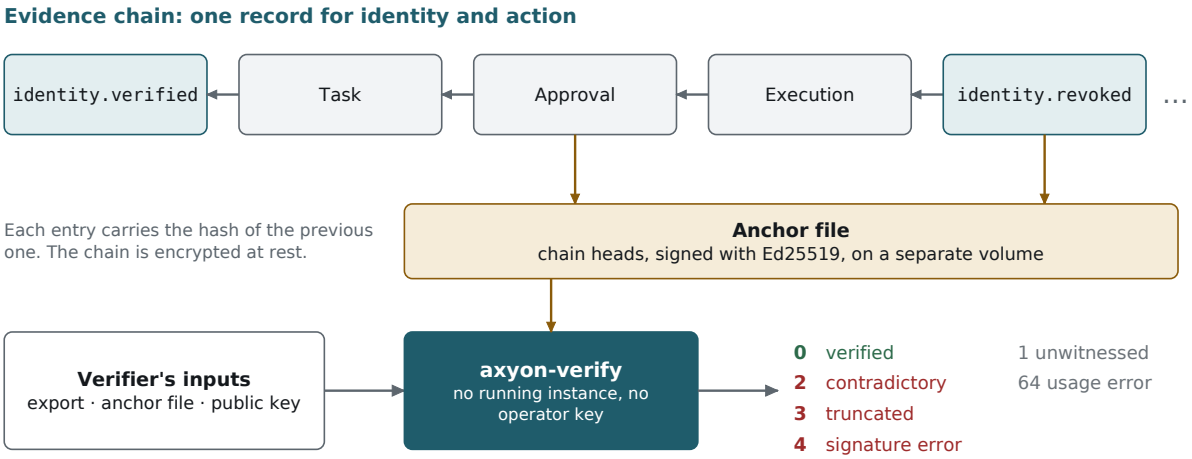


Figure 6 · Evidence chain, anchors and third-party verification

4.5 Approval stays human — and that is tested

The hardest acceptance condition of AXYON Fleet is worded inconspicuously: **an agent never approves an agent.**

How seriously this is meant is shown by the test case with which the condition was demonstrated on the built artefact on 2 September 2026. An AI agent was given everything that technically entitles to an approval: a signed access token with the role "approver" and a client certificate that belongs exactly to this token. Connection in order, binding satisfied, role present. The approval request was rejected (the product answers in German):

```
HTTP 403 – „Freigabe erfordert ein menschliches Subjekt" (approval requires a human subject)
```

The counter-test, without which this finding would be worth nothing: the same setup with a human subject behind the token — approval granted, recorded, in the chain. The distinction depends on the subject kind stated in the token, not on a convention; it is not an operating setting and not a prompt a model could override.

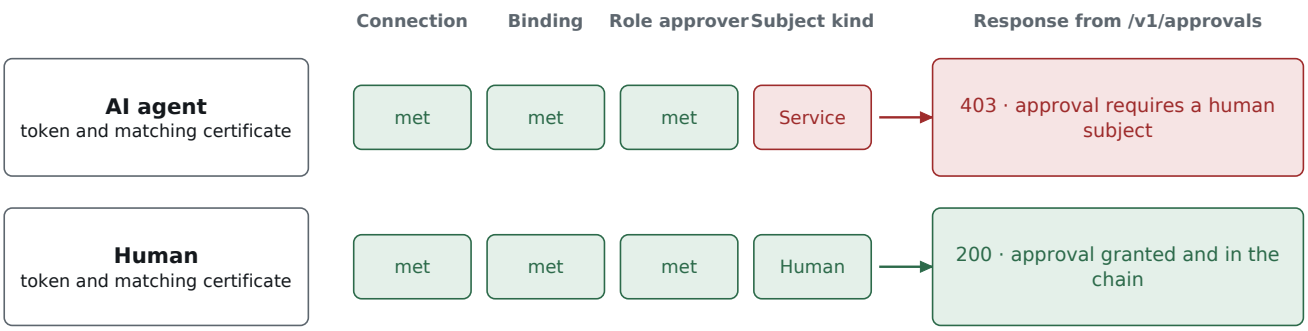


Figure 7 · Test case of 2 September 2026: same role, same binding, different subject kind

The second rule of the same family belongs here: **self-approval is not permitted.** Whoever requested an action cannot approve it — not even with the approver role.

4.6 Approvals are bound — and consumed

An approval in AXYON Fleet is bound to the specific task **and its content** and is valid exactly once:

1. A write action without approval is rejected — fail-closed.
2. A human with the approver role approves the specific task.
3. The same action goes through and is executed — verifiably, with effect.
4. The same action again: rejected. **The approval has been consumed.**

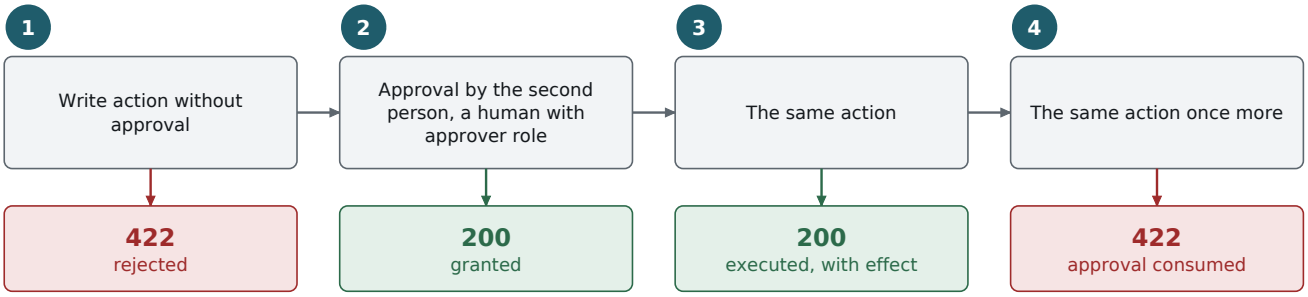


Figure 8 · Life of an approval, with the responses from the evidence run of 27 October 2026

Step 4 is the one that could be left out, and the one that completes the commitment. An approval that lives on is a standing authorisation, and that is what must not exist.

05
Architecture: shell and pinned core

AXYON Fleet is built as its own **shell** around the same core that also carries AXYON. All identity, certificate and revocation logic lives in the shell; the core stays free of input/output, time and configuration and is consumed pinned to an accepted state — **consumed, not copied**, no fork.

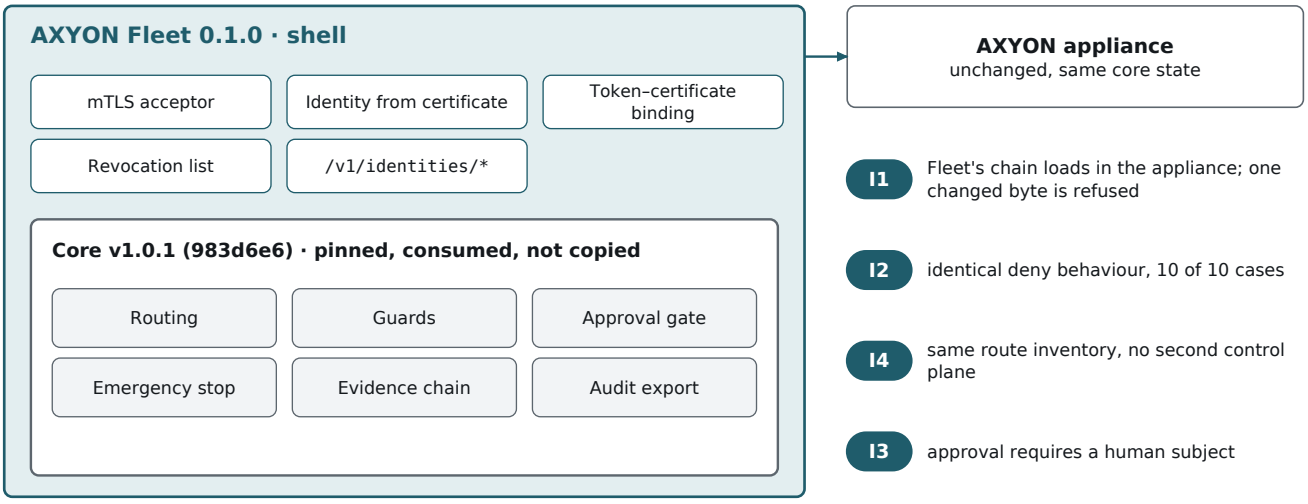


Figure 9 · Shell and pinned core, with the four invariants

Four invariants secure the relationship between the two products, and all four are acceptance conditions with an evidence run on the built artefact:

Invariant	Content	Evidence (2026-09-02, on the built image)
I1	A record written by AXYON Fleet has the same format: it loads in an **unchanged** AXYON appliance of the pinned state	Trail written by Fleet, loaded and verified by the appliance; counter-test: one changed byte, and the same appliance refuses to start
I2	AXYON Fleet relaxes no check: identical deny behaviour	Deny matrix against both shells, 10 of 10 cases identical
I3	Approval requires a human subject	Section 4.5
I4	No second control plane beside the existing one: matching route inventory, Fleet's own routes only behind the auth boundary	Route inventory matches in both directions

The artefact itself is built to minimise attack surface: distroless (no shell, no package manager), process without root rights, file system read-only in operation, all capabilities dropped; the operator console of the

existing product is not contained in AXYON Fleet, and switching it on is answered with a refusal to start. Refusal to start is the basic pattern: half an mTLS configuration, an unknown identity source, a binding rule without mTLS, a revocation list without mTLS — each of these ends the start instead of producing an unclear operating state.

06

Operation

Delivery. AXYON Fleet is handed over as a single-architecture container archive with a SHA-256 checksum — no registry, no registry account at the customer. Version 0.1.0 was accepted on the image of 2 September 2026. On Intel (`linux/amd64`) the same version runs as the reference instance; a customer archive for Intel with full acceptance does not exist yet. The limit that goes with it and belongs in every offer: the checksum shows that the file was not changed in transit. It does **not** show its origin. A signature chain and SLSA provenance for the image are open for AXYON and for AXYON Fleet alike (planned delivery stage 2).

Identities. Authority over issuance lies with the customer: which body in the organisation issues a certificate to an agent is the operator's decision — AXYON Fleet hardens that authority, it does not create it. If an agent can obtain a new certificate for itself, no mechanism helps; that is the first question of commissioning, and it is an uncomfortable one.

Authentication paths. For access tokens, AXYON Fleet supports OIDC with four demonstrated success paths (HS256, RS256 with PEM key, pinned JWKS, live JWKS over HTTP), each with an expired token as the counter-test. On the delivered image itself, the HS256 path has been tested; the other three are demonstrated against the program the image is built from. A dev-token procedure exists as a documented bootstrap for reference and demonstration operation, not as a production path.

Multi-node, with a condition. Approvals and the emergency stop apply across all instances once the shared approval and stop register is set up; without it, the emergency stop acts on the local node only. In the evidence run, an approval was granted on node 1 and the action executed on node 2; the emergency stop on node 1 also stopped node 2, and releasing it required the approver role.

Key rotations. Both rotations — encryption key and chain key — are demonstrated on existing data, including the counter-test: a wrong old key leads to a refusal to start, not to a silent fresh start.

Operating duties. The deployment document lists ten items that must be set before production — from the keys and the certificate root to storing the anchor file on a different volume.

Backup. On the reference instance the backup is demonstrated with a restore test (27 October 2026): a second instance was built from the backed-up files; it loaded and verified all 21 entries of the chain, with the same key identifier and the same anchor key. The restore test is part of operation at the customer in the same way.

07

Evidence runs

AXYON Fleet is built on the principle that evidence produced once by hand is a log; it only becomes a test when it is repeatable. All following runs are script-driven, repeatable and fail-closed; a skipped case does not count as passed.

Date	Run	Result
2026-09-02	Gates: format, static analysis without exceptions, 31 tests, supply-chain check with SBOM	green
2026-09-02	mTLS integration evidence: identity, binding, revocation/withdrawal, seven refusal-to-start variants	green
2026-09-02	Operating evidence: two nodes, both rotations with counter-test, four OIDC success paths with expired token as counter-test, self-approval, I3, full chain with consumed approval	green
2026-09-02	**Acceptance on the built artefact**: I1 with byte counter-test, I2 (10/10), I4, self-approval, full chain, rotations, OIDC and I3 against the image	fully green

Date	Run	Result
2026-09-02	First reference instance (since replaced): commissioning with ten test cases in live operation	10/10
2026-10-27	Security update of the dependencies; gates afterwards: format, static analysis, 31 tests, supply-chain check, SBOM	green
2026-10-27	amd64 build on the Intel build host: format, static analysis, 31 tests there; smoke test on the running container via mTLS	19/19
2026-10-27	Reference instance on Intel: evidence run N1–N10 (handshake without certificate rejected, someone else's token 401, write without approval 422, self-approval 403, approval by the second person, action executed, approval consumed)	10/10
2026-10-27	Backup of the reference instance with restore test	21 of 21 entries verified

On the rows of 27 October: the full acceptance with invariants I1, I2 and I4 was run on the image of 2 September. On the Intel image (amd64), the smoke run and the evidence run N1–N10 are demonstrated; the full acceptance has not been run there yet. The supply-chain check (vulnerabilities and licences) ran for the same dependency state on the development machine, not on the build host.

	Image of 2 Sept 2026 first acceptance	**Intel reference (amd64)** 27 October 2026
Gates (format, static analysis, 31 tests)	green	green
Supply-chain check (vulnerabilities, licences, SBOM)	green	green, on the development machine
Smoke test on the running container via mTLS	13 check lines	19/19
Full acceptance: I1 with counter-test, I2, I4	fully green	not run
I3: machine with approver role cannot approve	shown on the image	not run
Evidence run on the reference instance	10/10	N1–N10: 10/10
Backup with restore test	not shown	21 of 21 entries
Security update of 27 October included	no	yes
Customer archive with checksum	exists, older than the update	pending

Figure 10 · Evidence status of version 0.1.0: what has been tested where

Two features of this way of producing evidence are worth mentioning. First, the **counter-tests**: no success case counts without its failure case — the appliance that loads the unchanged trail must refuse the one changed by a byte; the valid token must stand next to the expired one. Second, the **honesty of the test tools themselves**: several findings of the acceptance runs were errors in the test scripts, not in the product, and were documented as such instead of being turned into product findings. A test tool whose reference point lies beside reality reliably reports findings that are none.

08

Regulatory framework

Human approval of write actions, the binding of the approval to task and content, its one-time character and its recording in a verifiable chain address the expectations of oversight and traceability that the EU AI Act (in particular human oversight under Art. 14) and frameworks such as NIS2 place on productive use of AI. AXYON Fleet comes with technical documentation following the pattern of Annex IV, an entry for the operator's AI inventory and a user manual that names the limits of the product. The Annex IV documentation is a draft (edition 0.1); open points are marked in it and not filled in.

For classification: AXYON Fleet is the control and evidence layer around the operator's models. In its own AI inventory the vendor lists the product as a separate entry as a precaution. How a deployment is classified under the EU AI Act is decided by the operator's use case, not by the tool; this paper does not make that classification. Nor does a tool create conformity. AXYON Fleet provides technical mechanisms and evidence on which the operator's argument can rest.

09

Honest limits

This list is in the manual and in the deployment document, and it belongs in every offer, because otherwise it comes back as disappointment.

No accumulation assessment. AXYON now assesses not only the single action but the sequence: several individually permitted actions of the same run are counted, and above a band AXYON requires a human approval to continue. AXYON Fleet 0.1.0 stands on a core state **before** this mechanism; twenty individually permitted actions do not add up to a threshold here. The follow-up has been decided since 6 November 2026. It is not built, and there is no date. Anyone who buys machine identity and expects accumulation protection does not get it in this version.

No signature chain for the image. The checksum shows the transfer, not the origin (section 6).

No customer archive for Intel yet. The reference instance runs on Intel (amd64). A customer archive with full acceptance does not exist for this architecture.

The accepted archive is older than the security update. It dates from 2 September 2026. The dependency update of 27 October is contained in the source state and in the Intel reference, not in this archive. A delivery build from the updated state is pending.

No user interface. Version 0.1.0 is operated through the HTTPS interface and the organisation's own operating tools. A console is not included.

No delegation chains. A machine identifies itself; it does not verifiably act *on behalf of* another. The architecture is open to this, it is not built — and without a counterpart at the customer that goes along, it would be worthless.

Connection frequency is not in the trail. The identity record is deliberately written once per identity and process run; a conspicuous connection pattern is therefore not visible. This keeps the trail readable and is a documented design decision.

Multi-node effect only with the register set up (section 6).

Issuance authority lies with the customer (section 6). AXYON Fleet hardens it, it does not create it.

10
AXYON and AXYON Fleet side by side

Why two products and not one large version? Because a tier would define the existing product by what it lacks. AXYON can do everything it can do. AXYON Fleet addresses a different situation: organisations with an agent fleet whose members have to identify themselves.

	AXYON	AXYON Fleet
Deployment situation	One AI application, one team, one workplace	Several autonomous agents facing one control layer
Identity of the counterpart	Access token	Certificate (mTLS) + bound token, revocation per request
Accumulation assessment (chain gate)	present	not today (section 9)
Delivery	Container appliance or native macOS app	Container appliance
Record format	the chain	the same chain, same format (I1)
External verification	`axyon-verify`	the same tool
Operation	Operator console	HTTPS interface, no user interface
Contract model	see the AXYON documents	Purchase, no subscription

Both products run on the operator's premises and follow the same governance rules, with the one named exception of the accumulation assessment.

11
The test question

It has nothing to do with us and can be answered in ten minutes:

> **Count the non-human identities in your organisation — service accounts, API keys, automation users, agents. Then mark those whose right to act you could withdraw today. Not at the expiry date — today.**

The difference between the two numbers is your real topic. It grows faster than any other number in your IT, and it grows whether you use AI agents or not.

If that difference is large in your organisation and you want to talk about what follows from it: **office@zeryon-systems.com**

Appendix A
Terms

Term	Meaning in this paper
mTLS	TLS connection verified on both sides: the client also identifies itself with a certificate
SAN URI	name field in the certificate from which AXYON Fleet reads the machine identifier
Binding	the rule that token identifier and certificate identifier must match; the certificate is authoritative
Revocation	withdrawal of an identity at runtime, checked on every request
Chain / trail	append-only, hash-chained record of all decisions, written before the reply
Anchor	signed chain head (Ed25519) in a separate file, basis of external verification
`axyon-verify`	stand-alone verification tool: recalculates the chain without a running instance and without the operator's key
fail-closed	refuse when in doubt: unclear configuration prevents the start, an unchecked action prevents execution
NHI	Non-Human Identities: service accounts, API keys, agents

Appendix B

References

Every statement in this paper is assigned to an architecture decision (ADR), a build slice or an evidence run. Selection:

Statement	Evidence
Two products, not a tier; language rule	ADR-048
mTLS, identity from the certificate; reverse-proxy path rejected	SLICE-60; integration evidence S3
Binding token↔certificate, certificate authoritative	ADR-062 / SLICE-63
Revocation per request, withdrawal, asymmetry, own routes behind the auth boundary	SLICE-61 / ADR-064; S3 run, K61 series
Identity events in the same trail, recorded once per process run	ADR-063
I1–I4 as acceptance conditions, shown on the artefact	ADR-048; acceptance on the image 2026-09-02
I3 test case (agent token with approver role → 403, human control case → 200)	SLICE-60 criterion 6; operating evidence S4 and acceptance on the image, both 2026-09-02
Full chain with consumed approval	S4 (V1–V5); reference-instance commissioning (N4–N10)
Four OIDC success paths with counter-tests	S4, 2026-09-02
Multi-node and emergency stop across two nodes, with condition	ADR-037; S4 (A0–A6)
Both key rotations with counter-test	S4 (B/C series)
External verification `axyon-verify`, Ed25519 anchor	ADR-049
Pinned core, consumed not copied	ADR-048 / ADR-056
Delivery as archive with checksum; limit on origin	ADR-047 / ADR-075
Distroless, no root, read-only, console refused	ADR-053; artefact properties in the build run
Accumulation assessment missing in Fleet 0.1.0	ADR-075; DEPLOYMENT §8; manual ch. 21
No delegation chains	ADR-048
Connection frequency not in the trail	ADR-063
Reference instance on Intel, N1–N10, smoke 19/19	Execution note Fleet Intel reference, 2026-10-27
Backup with restore test, 21 entries	same note, package 3
Security update of the dependencies	Fleet commit `94295c7`, 2026-10-27
No customer archive for Intel; archive of 2026-09-02	`dist/` record for commit `fba8ccd`; ADR-079, inventory table
Follow-up of the accumulation assessment decided, not built	ADR-079 (accepted 2026-11-06, "nothing implemented")
OIDC: HS256 on the image, three paths against the program	Fleet risk log, final state 2026-09-02

The complete evidence logs and the technical documentation following the Annex IV pattern are disclosed to interested parties in conversation.

Limits of this document

- Checked against the Fleet repository (main `dccb7a7`, tag `v0.1.0`) and the AXYON vault as of 16 November 2026. Build host and reference instance were not touched that day; statements about them come from the execution note of 27 October 2026.
- No evidence run was repeated for this document. Every result carries the date of its run.
- The figures are schematic. They show no measurements other than the responses and counts documented in the text.
- This English edition is a translation of the German edition of the same date. Tool output and log messages stay in the language of the tool (German).
- The document does not replace the classification of a use case, legal advice or a production acceptance.