TECHNICAL WHITE PAPER

PAIL

An Authority and Evidence Layer for Physical AI Actions

Glowfly Identity Systems Co., Ltd.

Version 1.1 · November 2026

ABSTRACT

Robots increasingly execute motion that was planned by software no human reviewed: large language model planners, autonomous agents, fleet schedulers and remote consoles. The command interface of a typical robot accepts whatever arrives on it, and after an incident there is rarely a record that an independent party can use to establish who authorized a given motion and within what limits.

PAIL (Physical Action Identity Layer) addresses this gap with two functions. As an authority layer, it requires that every physical action be covered by a signed, narrowly scoped, time-limited *action token*, and it enforces that requirement at a gate placed in the command path on the robot. As an evidence layer, it records each authorization decision in a tamper-evident journal before the decision takes effect, and it lets third parties verify that journal against a published specification with published tools.

This paper describes the problem PAIL is designed for, its design principles, its architecture, the life of an authorized action, its behaviour under failure, its evidence model, and the boundary between what PAIL upholds and what remains the responsibility of the integrator. It also summarizes what has been measured to date and states the limits of those measurements. PAIL is not a safety system and does not replace one; it is the layer that decides whether a requested action is authorized and preserves the proof of that decision.

CONTENTS

1 Introduction

2 Problem Statement

3 Design Principles

4 Architecture

5 The Action Token

6 The Life of an Action

7 Behaviour under Failure

8 Evidence

9 Trust Boundary

10 Integration

11 Operations

12 Observation

13 Measured Results

14 Availability

15 Scope and Limitations

16 Conclusion

1 Introduction

The software stack of a mobile robot or manipulator has well-established layers for perception, planning, control and functional safety. It has, in most deployments, no layer that answers a narrower question: *was this particular action authorized, by whom, and under what bounds?*

For a long time the question did not need its own layer. Motion was requested by an operator or by a program that an engineer had written and tested, and responsibility followed the organization chart. That assumption no longer holds. A planner built on a language model may propose a motion that no engineer anticipated. A scheduling agent may dispatch tasks to robots from several vendors. A remote console may be operated from another site. In each case the entity that decides and the mechanism that moves are separated by a network, by an organizational boundary, or by a model whose output is not predictable in detail.

Two consequences follow. First, the robot needs a way to refuse a command that carries no proof of authorization, independently of where the command came from. Second, the organization needs a record of authorization decisions that remains credible when the organization itself is the party under scrutiny, for example in an incident investigation, an insurance claim or a regulatory inquiry.

PAIL is designed to provide both. It introduces an explicit unit of authority for physical actions, enforces it where commands reach the actuators, and produces evidence that can be checked without trusting the operator or the vendor.

2 Problem Statement

2.1 Commands without provenance

A velocity command on a ROS 2 topic, a motion request on a vendor API or a line on a control socket carries its parameters and nothing else. It does not state who requested it, which task it belongs to, how long it is valid, or what limits apply to it. The controller executes it because it arrived. If an unintended node publishes on the command topic, or a planner emits a motion far outside the intended task, the robot has no basis on which to distinguish that command from a legitimate one.

After the fact, the available records are usually application logs written by the same system whose behaviour is in question. Such logs can be incomplete, can be edited, and cannot by themselves show that a given entry existed at the time it claims.

2.2 What existing layers provide

Several existing mechanisms address adjacent concerns. PAIL is designed to work with them, not to replace them.

Layer	Question it answers	What it leaves open
Transport security (for example DDS Security as used by SROS2)	Which participant may publish or subscribe to which topic, and is the message authentic on the wire	Whether one specific command is within what was authorized for one specific task
Functional safety (safety controllers, protective stops, speed and separation monitoring)	Whether the machine state is physically safe	Who authorized the motion, and whether it belonged to an approved task
Fleet management and access control	Which user or system may log in and dispatch work	What happens between dispatch and the actuator, and what can be proven afterwards
Application logging	What the application chose to write down	Integrity, completeness and independence of the record

The gap is the space between "this participant may talk on this channel" and "this machine is in a safe state": the authorization of an individual action, bound to a task and to limits, enforced at the point of execution and recorded in a form others can verify.

2.3 Requirements

The design of PAIL follows from six requirements.

1. **Action-scoped authority.** Authorization must be expressed per action, with explicit bounds such as speed, duration and target, and with an explicit validity period.
2. **Enforcement in the command path.** The check must sit where commands reach the actuators, and absence of authorization must result in no motion.
3. **Third-party verifiable evidence.** Records of authorization decisions must be tamper-evident and verifiable by parties who trust neither the operator nor the vendor.
4. **Bounded behaviour under communication loss.** A robot that loses contact with the authorizing service must neither continue indefinitely nor depend on connectivity for every control cycle.
5. **Integration without replacing the robot stack.** The layer must attach to existing command interfaces and leave planning, control and safety functions in place.
6. **Separation of observation from authorization.** Monitoring data must never become an input to an authorization decision.

3 Design Principles

Authority is explicit, scoped and time-limited. An action is authorized by a token that names the robot, the plan and step it belongs to, the action type, the limits that apply, and a validity window. A token authorizes one step once.

The verifier checks the bytes that were signed. The gate verifies the signature over the exact byte sequence the issuer signed and does not re-serialize the message before checking it. A component that merely relays authority cannot alter or create the token.

The record precedes the effect. An authorization is appended to the journal before the issuing service returns it. If the record cannot be written, the authorization is refused. A request that passes the format checks is recorded even when it is refused.

The default is refusal. Missing, expired, consumed, malformed or out-of-scope authority results in a refusal with a named reason. Requests that exceed policy are denied, not silently reduced to fit.

The robot holds no issuing key. Signing keys remain with the issuing service. Components on the robot verify authority; they cannot create it.

Observation does not authorize. Telemetry, dashboards and notifications are derived from authorization records and robot reports. No element of the observation path feeds back into a decision.

Evidence is independently checkable. The formats of checkpoints, anchor records and context hashes are specified at the byte level and published with frozen test vectors, and published tools verify them. The tools that check anchor proofs and context hashes share no code with the components that produce the records.

---

4 Architecture

PAIL consists of an issuing service, a set of components on the robot, an evidence chain, and an observation path. Figure 1 shows the components and the two paths that connect them.

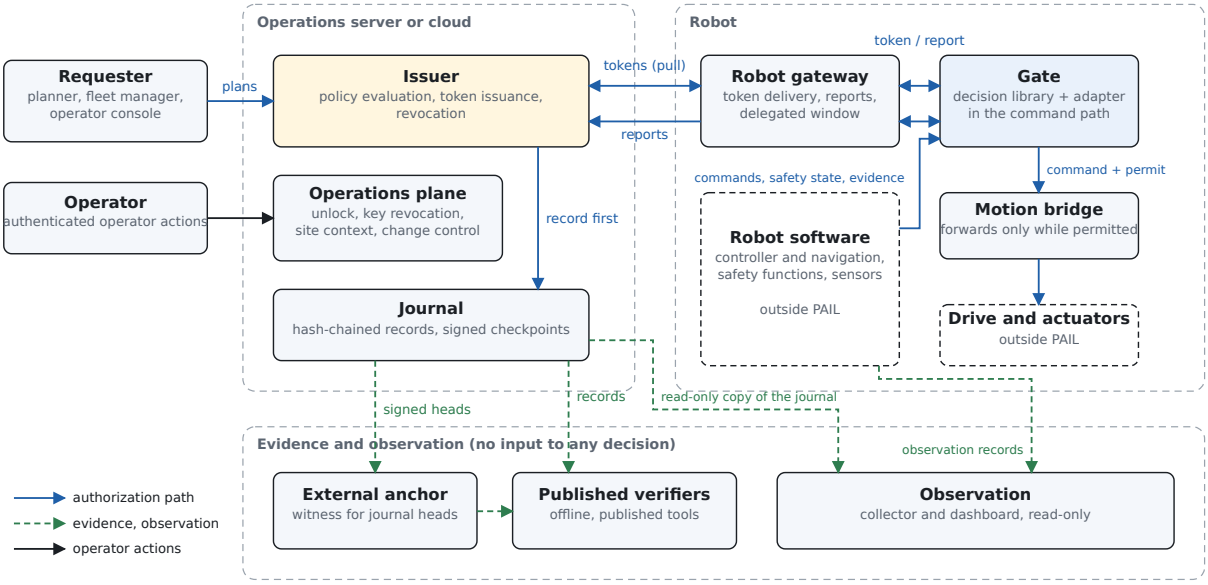

Figure 1. PAIL architecture. The authorization path (solid) carries authority from the Issuer to the gate in the command path. The observation path (dashed) is derived from records and reports and does not feed any decision.

4.1 Components

Component	Location	Role
Issuer	Operations server or cloud	Evaluates each requested plan against policy, skill definitions and the current site context; issues signed action tokens; writes the journal; applies revocation
Robot gateway	On the robot	Obtains tokens for the gate, maintains a bounded delegated window for periods without connectivity, delivers decision reports, reconciles state with the Issuer after disruption. Holds no issuing key
Gate	On the robot, in the command path	A decision library with a C ABI, plus an adapter for the robot's command interface. Evaluates every command against the current token and the robot's safety and evidence inputs
Motion bridge	On the robot, last hop before the drive	Forwards commands only while a valid permit signal is present; outputs a stop when the permit lapses
Journal	With the Issuer	An append-only, hash-chained record of authorization decisions, gate decisions and operator actions, with signed checkpoints
Anchor	External to the operator	An external reference point to which signed journal heads are submitted, so that the journal cannot be rewritten without contradiction
Verifiers	Anywhere, offline	Published tools that check journal integrity, checkpoint signatures, anchor proofs and token context hashes
Observation	Monitoring network	A collector and dashboard that display authorization state, freshness and uncertainty, read-only
Operations plane	Operations server	Operator functions: unlock after a safe stop, key revocation, site context events, maintenance lockout, and controlled software or policy change

4.2 Two paths

The **authorization path** runs from the requester to the Issuer, from the Issuer through the robot gateway to the gate, and from the gate through the motion bridge to the drive. Decision reports travel back along the same path to the journal. Every element on this path either creates authority (the Issuer), verifies it (the gate), or carries it without the ability to change the token (the gateway; signature verification at the gate prevents any change to the token's content). Whether a token is currently active, and the current context hash, do reach the gate through the gateway; trust in that part is bounded by the token's expiry.

The **observation path** is separate. The collector reads a copy of the journal and observation records produced on the robot, verifies what can be verified, and presents the result together with its provenance. Notifications emitted for observers are copies of events; the gate, bridge and adapters do

not read them. This separation is a structural property of the design: an observation component can fail, lag or be compromised without changing what a robot is permitted to do.

4.3 Reference implementations

The gate is provided as a decision library with a stable C interface so that it can be embedded in different command paths. The reference integration for ROS 2 consists of a gate node and a motion bridge placed in front of the base controller. A second, experimental adapter demonstrates the same decision library in front of a non-ROS control socket in a software-only preview. An adapter contract and a conformance suite define what any further adapter must do.

---

5 The Action Token

5.1 Contents

An action token is a compact signed message. Its fields fall into six groups.

Group	Fields	Purpose
Identity	token identifier, chain identifier, sequence index	Names the token and its position in the plan
Binding	robot, plan, step, target object	Ties the authority to one robot, one task step and, where applicable, one object
Validity	issued-at, not-before, not-after	Limits the token to a defined time window whose length is set by policy (one hour by default)
Action	action type, skill and slot	States what kind of action is authorized
Limits	maximum linear speed, maximum angular speed, maximum duration, guardrails	Bounds the execution; every command is checked against them
Context	policy hash, context snapshot hash, schema version	Ties the authority to the policy and to the site conditions under which it was granted

The token is signed with Ed25519. The gate verifies the signature over the original bytes and resolves the signing key through a signed key set that supports rotation and revocation. A revocation takes effect when the gate loads a key set that carries it.

5.2 Single use

A token authorizes one step once. The gate derives a consumption key from the verified token bytes and refuses a token it has already consumed. The Issuer, for its part, advances a plan only when the

completion it receives names the token of the step in progress and matches the skill's completion rule. Replay of a previously valid token is therefore refused at both ends.

5.3 Plans and chains

A requester submits a plan consisting of ordered steps. If the plan is accepted, the Issuer creates a *chain*: one token per step, served to the gate one at a time in order. The gate holds the token for the current step; when the step completes and the completion is recorded, the chain advances. A chain can end by completion, by cancellation from the requester, by replacement when a newer plan is accepted for the same robot, or by revocation.

5.4 Binding to context

Authorization depends on conditions that can change: a person entering a zone, a robot placed under maintenance, an area cordoned off. The Issuer maintains a model of the site and, at issuance, records a hash of the part of that model on which the authorization depends. The gate compares this value with the current one when it rechecks authority. If the conditions under which a token was granted no longer hold, the authority is withdrawn and the corresponding chain is revoked. The published specification defines how this hash is computed, and a public tool recomputes it from the site model and the journal.

---

6 The Life of an Action

Figure 2 follows one plan from request to record.

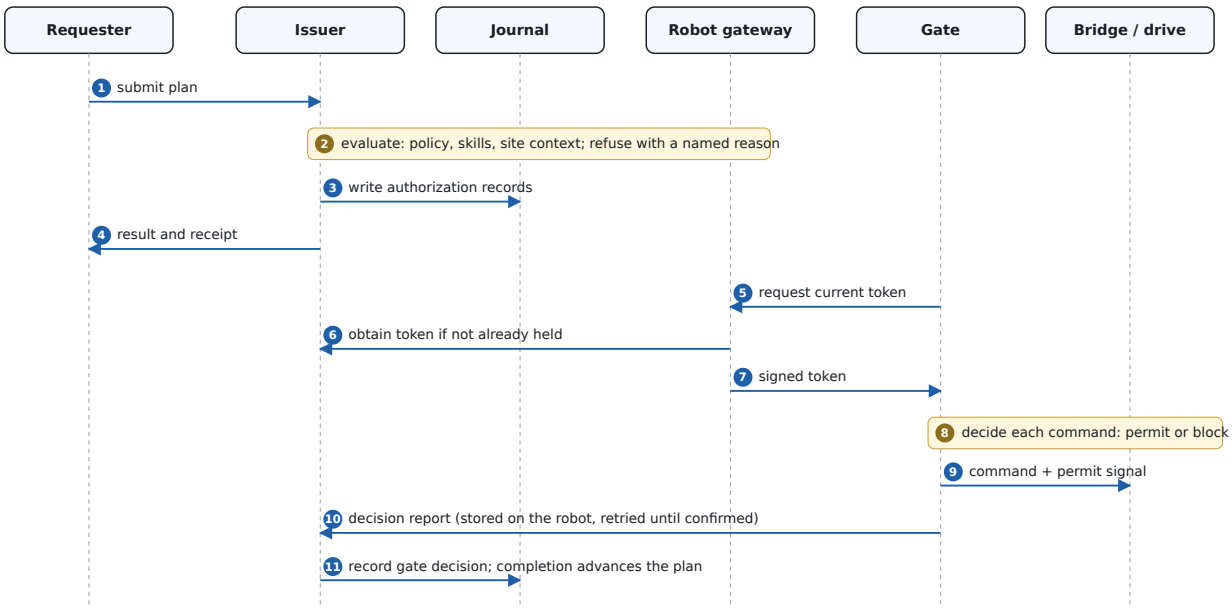

Figure 2. Life of an authorized action. Refusal at any stage produces a named reason and a journal record.

Request. A requester (a planner, a fleet manager or an operator console) submits a plan to the Issuer over an authenticated channel. The requester's identity is taken from the channel, not from the content of the request. Each state-changing request carries an operation identifier and receives a receipt, so that a retry after a lost response returns the original verdict and chain identifier instead of creating a second outcome.

Evaluation. The Issuer checks the plan against the requester's registered permissions, the policy envelope, the skill definitions for the robot class and the current site context. A plan that violates a bound is refused, and the refusal names the bound. The Issuer does not reduce a request to make it acceptable: a planner that asks for thirty metres where three are permitted receives a denial, not a three-metre motion it did not ask for.

Record, then issue. For an accepted plan the Issuer writes the authorization records to the journal and only then returns the result. A failure to write the journal is a failure to issue. Plans refused after the format checks are recorded too, so the journal shows attempts as well as grants.

Delivery. The gate requests the current token through the robot gateway. The gateway answers from the authority it already holds where it can, and otherwise asks the Issuer. Tokens are pulled by the robot side; the Issuer does not push authority to a robot.

Decision. For every incoming command the gate evaluates a fixed sequence of checks. The principal ones are, in order: the sanity of its clock, the robot's reported emergency-stop state, the presence of a token, the token's signature and validity window, the match between the token's context and the current context, single use, the match of the action type, the limits in the token against the parameters of the command, the freshness and consistency of the evidence the robot provides about

its target, and the freshness of the safety signal. The first failed check determines the outcome and its reason. The possible outcomes are a permit, a block, or an emergency-stop state.

Permit and forwarding. On a permit the gate forwards the command unchanged and emits a periodic permit signal. The motion bridge forwards commands to the drive only while that signal is current. If the signal stops, the bridge outputs a stop. In the reference implementation the gate evaluates every command against the active token, re-confirms the current authority with the gateway every 100 ms, and emits the permit signal at 20 Hz; the bridge applies a 100 ms deadline. These values are configuration defaults.

Report and completion. The gate reports its decisions to the Issuer through the gateway. Reports that change authority, such as step completion or a fatal condition, are written to durable storage on the robot before they are sent and are retried until the Issuer confirms them. The Issuer records the reported gate decisions in the journal, and a report that changes authority, such as a completion, is acted on only once its record is written; a completion that matches the completion rule defined for the skill advances the chain.

---

7 Behaviour under Failure

A layer that sits in the command path is defined as much by what it does when something fails as by its normal operation.

7.1 Refusal as the default

Each component is built so that its failure removes authority. A gate without a token blocks. A gate that cannot verify a token blocks. A bridge without a current permit signal stops. An Issuer that cannot write its journal does not issue. A deployment whose security configuration is incomplete refuses to start in the production profile.

7.2 Loss of connectivity

Robots operate over wireless links that drop. Requiring a round trip to the Issuer for every control cycle would make the link a single point of failure; allowing a disconnected robot to continue without limit would defeat the purpose of authorization.

PAIL bounds the problem. Policy may allow the Issuer to pre-authorize a limited window of upcoming steps, which the gateway holds as delegated authority. While the link is down, the gateway serves that window from local state, and the gate verifies each token exactly as it would otherwise. The depth of the window is the distance the robot can proceed without contact. When the window is exhausted and the Issuer cannot be reached, the gate blocks after a short grace on the permit it holds. Delegated authority is released when its tokens expire, when the gate reports expiry, when the Issuer signals revocation, and when reconciliation after reconnection shows that the chain has ended.

7.3 Restart and reconciliation

After a restart of the gateway or the gate, authority held before the restart is not assumed to be valid. The robot side declares what it holds and how far it has progressed, and the Issuer responds with a verdict before any token is served again. Chains that were delegated to an execution instance that no longer exists are cancelled. The Issuer and the gateway also confirm that they implement the same version of their communication contract; a mismatch at start-up prevents operation instead of proceeding on unshared assumptions.

7.4 Unsafe conditions

The gate accepts a safety-state input from the robot's own safety functions. When that input reports an unsafe condition, the gate stops issuing permits, latches the affected authority so that it cannot be reused, asserts a stop request toward the robot's safety layer, and reports the condition as fatal. The fatal report is retried until confirmed, and the Issuer revokes the robot's current chain when it arrives. The latch on the robot does not depend on the Issuer's response.

7.5 Change of plan and change of context

When a newer plan is accepted for a robot, the previous chain is superseded and its authority is withdrawn at the gateway. When the site context changes in a way that affects an active authorization, the corresponding chain is revoked. The remaining steps are re-evaluated against the changed context without returning to the planner; steps that no longer satisfy policy are not re-authorized.

7.6 Software integrity

The gate can report a measurement of the software it is running. In deployments that require it, the Issuer withholds issuance and serving for a robot whose gate has not presented a valid, current measurement. This mode is optional and is disabled by default.

---

8 Evidence

8.1 The journal

The journal is an append-only sequence of records in which each record is cryptographically linked to its predecessor. Deleting, inserting, reordering or altering a record breaks the chain and is detected on verification. Signed checkpoints are inserted at intervals, when writes pause and when the journal is closed. The Issuer verifies these signatures at start-up and does not start on a journal that fails the check. The journal contains accepted and refused plans, issued tokens, gate decisions including completions, revocations and their causes, operator actions, and changes to the site context.

Two properties matter for evidentiary use. First, because the record is written before the result is returned, there is no state in which authority was issued and no record exists; this is an ordering guarantee at the level of the issuing process and is stated together with its durability limits in the product's security documentation. Second, a refused request leaves a record of the attempt without a record of a grant, which distinguishes "attempted and refused" from "never requested".

8.2 The external anchor

A hash chain proves internal consistency, not that the chain presented today is the chain that existed yesterday. For that, the journal head is signed and submitted periodically to an anchor outside the operator's control. Submission is asynchronous and does not block issuance; a failed submission is queued and retried.

Glowfly operates a hosted anchor service that stores submitted heads in a transparency log and returns signed checkpoints together with inclusion and consistency proofs of the kind defined for certificate transparency. An evaluation deployment can use a local file anchor that exercises the same client path; a local anchor is not an independent witness, and the documentation says so.

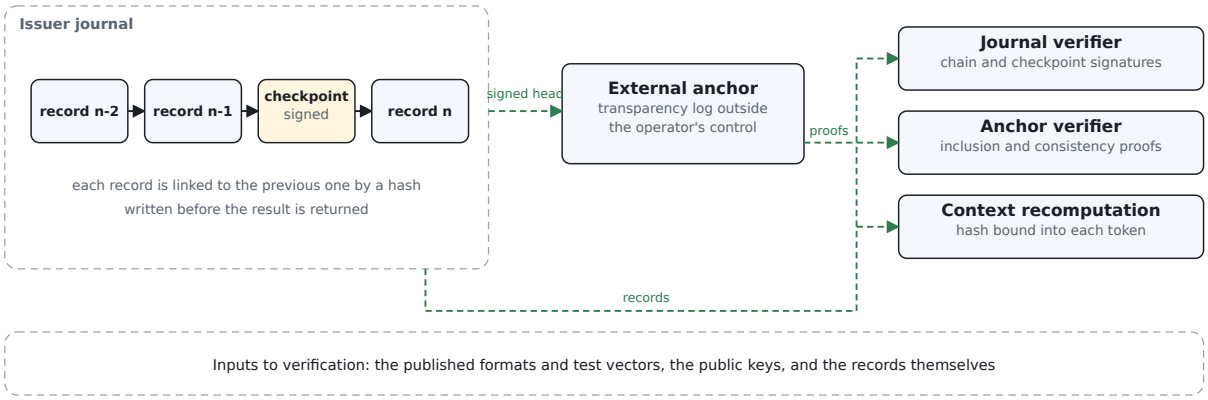

Figure 3. The evidence chain. Verification requires only published formats, public keys and the records themselves.

8.3 Independent verification

The canonical byte encodings and hash constructions of checkpoints, anchor records, Merkle proofs and context hashes are published as a specification with a named edition and a frozen set of test vectors. There are three verification tools. One checks the journal's chain and checkpoint signatures; given the Issuer's public key, it passes a journal only when every record is under a signed checkpoint. Another checks anchor checkpoints and their inclusion and consistency proofs. A third recomputes the context hash bound into a token from the site model and the journal. The latter two share no code with the components that write the records, and a party who wishes to build a second implementation can do so against the vectors.

8.4 What the evidence does and does not show

The journal shows who requested an action, what was authorized and within what bounds, what the gate decided for each command, and in what order these events occurred, in a form that cannot be altered without detection. It does not show that a physical stop occurred, that the robot's sensors were correct, or that an authorized action was the right action to take. PAIL records authority and decisions; physical outcomes are established by the robot's own systems.

---

9 Trust Boundary

PAIL states what it upholds as claims that can be refuted, and states separately what it delegates.

9.1 Four claims

Claim	Statement	It is false if
Raw-byte signature verification	The gate verifies the signature over the exact bytes the Issuer signed, without re-serialization	Any field can be changed and the token still verifies
Issuer-first write order	The authorization record is appended to the journal before the issuance response	A token is issued while journal writes are made to fail
Single use	A token authorizes one step once	A consumed token is accepted again
Fail-closed propagation	The default verdict is denial: an input that was not verified cannot evaluate to a permit, and the gate re-verifies the active token on every command	An unverified input is judged as permitted

9.2 Delegated responsibilities

Certain properties cannot be provided by an authorization layer and are assigned explicitly to the integrator or operator. The principal ones are these.

- **Transport authenticity on the robot.** The authenticity of the permit signal between gate and bridge relies on the transport security of the robot middleware. In a ROS 2 deployment this means SROS2 in enforcing mode; without it, permit messages can be forged.
- **Closing bypass paths.** The gate guards the command path it is placed on. Any other route to the actuators must be closed by the integrator.
- **Physical safety.** Protective stops, safety-rated monitoring and the final physical stop belong to the robot's safety system.
- **Key custody.** Physical and procedural protection of signing keys belongs to the deploying organization. PAIL supports keeping the signing key in an external key store.

- **Time.** Validity windows depend on reasonably synchronized clocks; the gate refuses to operate when it detects clock anomalies beyond a configured tolerance.

The remaining delegated items and the threat model are part of the published documentation.

---

10 Integration

10.1 The adapter contract

Integrating PAIL with a robot means writing or configuring an *adapter* around the decision library. An adapter has four responsibilities: to route every command through the gate, to act on the gate's decision without modifying permitted commands, to supply the gate with the robot's safety state and evidence inputs, and to stop forwarding when the permit lapses. The contract is specified, and a conformance suite exercises an adapter against a set of defined cases, including refusal without a token, refusal of out-of-bound commands, behaviour on loss of the gateway and behaviour when the adapter itself fails.

10.2 Relationship to SROS2

PAIL and SROS2 operate at different layers and are used together. SROS2 determines which participants may communicate on which topics and protects messages in transit. PAIL determines whether a given command is within what was authorized for a given step, and records that determination. PAIL depends on SROS2 for the authenticity of messages on the robot's internal network and does not attempt to provide it.

10.3 Requesters

Systems that request motion integrate through a gRPC interface defined by published protocol definitions: submitting and cancelling plans, and receiving structured denials that state the reason, the disposition the caller should adopt, and the earliest time a retry is meaningful. A requester needs no knowledge of the robot's internal interfaces.

---

11 Operations

Authorization in a running facility requires operator functions, and these are subject to the same discipline as issuance: every action is authenticated and journaled, an unlock and a key revocation require an operator's signature, and releasing a hold requires a signed approval.

- **Safe stop and unlock.** Reports that indicate forged or malformed authority, or a failure of the Issuer's own journal, place the Issuer in a safe-stop state in which no new plans are accepted. Leaving that state requires an unlock signed by an operator key, valid for a short window and usable once.

- **Key revocation.** Revoking an issuing key places the whole Issuer in a permanent lockdown that persists across restarts. A gate refuses the key once it has loaded a new key set that carries the revocation, so after a revocation the new key set is applied to every gate.
- **Site context events.** Operators and site systems inform the Issuer of conditions such as a person entering a zone or a robot entering maintenance. These events update the site model on which authorization depends.
- **Maintenance lockout.** A robot can be locked out for maintenance by one or more holders; issuance for that robot resumes only when every lock has been released.
- **Controlled change.** Software and configuration changes to robots can be performed under a hold during which issuance for the affected robots is suspended, the stop evidence submitted by the executor is bound into the record (it is not proof of a physical stop), and release requires an approval. Changes to policy and to the site model can be made subject to approval by two distinct parties, verified by the Issuer at start-up.

The contracts for these functions are part of the specification. Some are optional and are enabled by configuration.

---

12 Observation

The observation path gives operators a view of authorization state without becoming part of it. A collector verifies a copy of the journal while the Issuer continues to write it, ingests observation records from robots while distinguishing signature-verified records, uploads over an authenticated connection and self-reports, and exposes the result through a read-only interface and dashboard.

The design gives particular attention to uncertainty. Each displayed value carries the basis on which it is considered current, such as the time it was received or a bound established by a challenge the robot answered, and values whose basis has expired are shown as stale or unknown instead of as last known. The observation records explicitly state that they do not prove a physical stop. If the collector fails, robots continue to operate under the authority they hold; nothing in the collector can grant or withdraw it.

---

13 Measured Results

The following results are taken from the measurement reports published with the release. Each is valid for the configuration and the build stated and for nothing beyond it.

Platform	Scope	Result
Software bundles (Docker, Linux arm64 and amd64)	Simulated TurtleBot3 base, non-ROS adapter preview, multi-robot fleet, gate unit tests, lifecycle and plan replacement	On the 0.2.0 bundles: simulated TurtleBot3 test, 8 of 8 checks · adapter preview 15 of 15 · fleet of two robots, 15 of 15 each · gate tests 37 of 37 · lifecycle 21 of 21 · plan replacement 8 of 8
TurtleBot3 hardware, wheels raised	Encoder-based observation and acceptance judging	13 of 13 scenes in three runs on a pre-release build; scoped to that build
Jetson Orin Nano Super, CPU only	Synthetic reference with the real gate and bridge	7 of 7 scenarios; an 8-hour segmented soak with 752 exercise and cancel cycles and 22 injected faults, every cleanup passing
Isaac Sim on one L40S virtual machine	Bypass and lifecycle test matrix against the simulator's SDK endpoint	14 of 14 cases with the expected result in two fresh-state repeats of the release run: 13 passing and one positive control, which must detect a bypass, detected; one run of an earlier pre-release recorded one inconclusive case, which the documentation discloses

Added latency of the gate and bridge on the velocity command path, relative to a direct subscriber, was measured on 16 October 2026 with SROS2 enforcement enabled:

Platform	Command rate	Median	99th percentile
Jetson Orin Nano Super	10, 20 and 50 Hz	3.8–4.3 ms	5.4–7.0 ms
TurtleBot3 (Raspberry Pi 3B+), wheels raised	10 and 20 Hz	14.3–14.7 ms	21.9–23.4 ms

These figures describe the time for a command to pass from the input topic to the drive topic in steady state. They were obtained with an earlier build of the same gate and bridge and have not been repeated on the current release. The TurtleBot3 figures are an upper bound because the measurement probe shared the processor. They do not include the driver, firmware or motor control loop, which PAIL does not touch, and they are not a latency guarantee.

Not measured: robots other than those listed, floor driving, other simulators or accelerators, and any form of functional-safety certification.

14 Availability

Open core. PAIL Core is published as source under the Apache License 2.0. It comprises the gate decision library, the ROS 2 gate and motion bridge, the robot gateway, the adapters, the protocol definitions and client example, the journal libraries, the observation collector and dashboard, the verifiers, the specification and its test vectors, and the conformance suite.

Issuer. The Issuer is provided as an evaluation executable for research, development, interoperability testing and internal continuous integration. Its source is not published. Production operation is licensed separately.

Commercial offerings. Glowfly offers, separately from the open core: the Issuer for production use; a replicated Issuer for deployments that cannot accept the durability characteristics of a single node; the hosted anchor service; and a fleet operations layer that drives the change-control contracts described in Section 11.

Patents. Glowfly holds granted and pending patents related to the technology described here. Patent rights in the Apache-licensed materials are those granted by Section 3 of that license.

Regulatory position. PAIL does not claim conformity with any regulation or functional-safety standard. It produces records that a conformity assessment, an incident investigation or an audit can use.

---

15 Scope and Limitations

PAIL decides whether a requested action is authorized; it does not decide whether the action is wise. A correctly authorized action can still be the wrong action, and preventing that is the task of planning, perception and safety functions.

PAIL guards the command path on which its gate is installed. It provides no protection on paths that bypass the gate, and it relies on the robot middleware's transport security for message authenticity inside the robot.

The issuer-first guarantee is an ordering guarantee within the issuing process. Its behaviour under power loss depends on the durability configuration of the deployment, which the security documentation describes.

The security policy states the known limits of each release. For example, a journal cut back to an earlier signed checkpoint still reads as a valid record on its own and is revealed only by comparison with the head held by an external anchor. The same document states, together with the operating procedure, that a revocation of an issuing key reaches a running gate only when a new key set is applied.

The measurements in Section 13 cover a small number of platforms in defined conditions. They do not establish behaviour on other hardware, in other environments or at other scales.

---

16 Conclusion

As the origin of robot motion moves from engineered programs to generated plans, the question of who authorized an action, and how that can be shown, becomes a system requirement in its own right. PAIL treats it as one. It makes authority an explicit, signed and bounded object; it enforces that

authority where commands reach the actuators and removes it when conditions change or components fail; and it records each decision before it takes effect, in a form that parties outside the organization can verify with published tools.

The approach is deliberately narrow. PAIL does not replace transport security, functional safety or fleet management, and it states which properties it upholds and which it leaves to the integrator. Within that boundary it supplies what those layers do not: proof of authority for each physical action, and evidence of that proof that does not depend on trusting whoever holds the logs.

References

1. PAIL Core specification, edition PAIL-CORE-2026.1, test-vector set `bbddff16…`, including the canonical formats, the adapter contract, the communication contract and the test vectors. Published with the PAIL Core source.
2. PAIL trust boundary, threat model and security policy. Published with the PAIL Core source.
3. Object Management Group, *DDS Security Specification*.
4. ROS 2 documentation, *SROS2: security enclaves and access control*.
5. S. Josefsson and I. Liusvaara, *Edwards-Curve Digital Signature Algorithm (EdDSA)*, RFC 8032.
6. B. Laurie, A. Langley and E. Kasper, *Certificate Transparency*, RFC 6962.

Contact

Glowfly Identity Systems Co., Ltd. · www.glowfly.ai

Questions about the published source are handled through the project's issue tracker. Suspected vulnerabilities are reported through the address given in the project's security policy.

© 2026 Glowfly Identity Systems Co., Ltd.

PAIL Core is licensed under the Apache License 2.0. This document may be redistributed unmodified.