Vaelon: Safety Case Summary (TRL 3-4)

Draft for the ESA BIC application and design-partner technical review. This is a condensed safety case: a structured argument that the Vaelon runtime safety layer does what it claims, with every claim tied to reproducible evidence. It is a summary, not the full case; the full evidence is the reproducibility sheet in `vaelon-core/ASSURANCE.md`, which regenerates every figure below from a command.

- Document status: draft, founder-authored, not yet independently reviewed.
- Evidence baseline: `vaelon-core/ASSURANCE.md`, last measured 2026-07-27 (re-run this session).
- Every quantitative claim here is copied from that sheet, where it maps to the exact command that produced it. Nothing is asserted without a command behind it.

Two items below are marked for a decision: - `[FOUNDER DECISION]` needs Satyawan's call (usually an IP-exposure judgment). - `[NEEDS ASSURANCE REVIEWER]` needs a qualified software-assurance engineer to confirm exact standard-clause mappings before this goes into a formal submission.

---

1. Purpose and scope

Autonomous onboard decision-makers (reinforcement-learning policies, vision language models, planners) are beginning to fly and take real actions: collision-avoidance maneuvers, proximity operations, mode changes. These are black boxes whose failure can destroy the spacecraft or a neighbour.

Vaelon is an independent runtime safety layer placed between that onboard AI and the spacecraft bus. For each proposed action it predicts the one-step outcome, checks it against a fixed set of formal safety invariants, and returns one of ALLOW, BLOCK, or ESCALATE. On BLOCK it recommends a verified safe alternative; on ESCALATE it hands the decision up. Every decision is cryptographically signed onboard into a tamper-evident chain.

Scope of this safety case: the onboard decision core (`vaelon-core`) and the ground analysis that feeds it (`orbitguard`). Out of scope: the onboard AI itself (Vaelon treats it as untrusted), the spacecraft bus, and mission-level operations.

2. System description and safety concept

- **Boundary.** Vaelon does not control the spacecraft. It gates actions. The heavy astrodynamics (TLE ingestion, SGP4 propagation, conjunction screening) run on the ground and enter the onboard monitor as telemetry (for example `min_sep_km`, `range_rate_km_s`). The onboard core only evaluates invariants.

- **Fail-closed.** Non-finite telemetry (NaN or infinity) returns an error and a BLOCK verdict. A caller that ignores the return code still receives a safe decision. A monitor that passes bad input is worse than none; Vaelon refuses it.
- **Independence.** The verifier is deliberately separate from the AI it guards, so a fault in the AI cannot silently propagate into the safety decision.
- **Determinism.** The decision path performs no heap allocation and runs in bounded time, so worst-case behaviour is analysable, not statistical.

3. Maturity: what is and is not claimed

Current maturity: TRL 3-4. The properties below are ground-verified, formally proven, and bench-measured. They are engineering evidence, not flight evidence. Stated plainly, and repeated from the assurance sheet:

- **No flight heritage.** Nothing here has flown.
- **No independent third-party audit yet.** This safety case exists so that one can begin against a defined evidence base.
- The ground re-screen (Section 6) is a *soundness* demonstration on public orbital elements. It is **not** a validation of Vaelon against an operator's authoritative conjunction data messages (CDMs), which needs an operator or Space-Track data feed.

Being explicit about the boundary of the claim is itself part of the safety argument. Overclaiming is a safety defect in a safety case.

4. Safety requirements: the 11 invariants

The safety requirements are expressed as invariants the predicted post-action state must satisfy. They are fixed, inspectable, and (for the operator-configured subset) authored as data rather than code.

id	requirement enforced
INV-PWR	predicted battery state of charge stays above the floor
INV-THERM	predicted temperature stays within bounds
INV-SUN	the imager never points at the sun
INV-DV	a maneuver never exceeds the delta-v budget
INV-COLL	the verifier's own achievable separation clears the keep-out
INV-COLL-RESP	an active conjunction is answered by an evasive maneuver
INV-COLL-SCREEN	a closing object that will breach the keep-out is evaded early
INV-FAULT	under a fault, only safe-recovery actions are permitted

id	requirement enforced
INV-MODE	forbidden mode transitions (for example SAFE to IMAGING) are blocked
INV-COMMS	(soft) downlink when the buffer is full during a contact window
INV-BUFFER	the recorder never overflows

Severity is either hard (a violation forces BLOCK) or soft (a violation forces ESCALATE). When several invariants fire, the worst verdict wins.

5. Safety argument and evidence (claims to evidence)

Each claim is backed by a command and a measured result from the assurance sheet.

Claim A: The decision logic is correct over its entire input domain, not just on tested points. Evidence: eight safety theorems are formally proven with a model checker (`make verify`), result `8 successfully verified harnesses, 0 failures`. The proven properties are: 1. fail-closed: non-finite telemetry always returns an error, never a verdict. 2. safety-soundness: any violated hard invariant implies BLOCK; ALLOW implies every invariant passed with none in the near-band; ALLOW never coexists with a hard violation. 3. multi-object soundness: screening against many tracked objects never downgrades a hard violation. 4. rules and horizon soundness: the same guarantee holds for operator-configured rules and multi-step horizons as for the built-in checks. 5. decision totality: the mapping from (hard, soft, near-band) to verdict matches its specification exactly. 6. staleness-deadline monotonicity: once telemetry is stale it stays stale as it ages, so the oracle-gap (freshness) envelope can only ever tighten the gate. 7. deadline-never-allows: past the freshness deadline the degrade never turns a BLOCK or ESCALATE into an ALLOW; it only upgrades an ALLOW to ESCALATE. 8. CDM collision gate: a ground-computed collision probability above the alert threshold on a non-evasive action is never ALLOWed, so the collision screen detects a conjunction from the ground's full propagation, not only from onboard closing-rate extrapolation (which is weak against curved relative motion). Note: formal proof surfaced a real latent severity-downgrade defect in the multi-object fold, since hardened. That is the value of proving over fuzzing.

Claim B: The flight core matches an independent reference implementation. Evidence: differential test against the validated Python reference engine over 20,000 threshold-biased scenarios, comparing the decision and 22 flags: **20000 / 20000 identical (100.0000%)** (`make differential`).

Claim C: The core never panics and always fails closed. Evidence: property fuzzing over approximately 100,000 inputs (`tests/properties*.rs`):

no panic, always fails closed, safety-sound.

Claim D: Behaviour is verified by a conventional test suite. Evidence: **81 tests passing, 0 failed** (`make test`); lints clean with warnings treated as errors (`make lint`).

Claim E: Timing is deterministic and bounded, suitable for a control loop. Evidence (`make bench`, measured on Apple M4; these figures are hardware-specific): amortized 14 ns per decision (71.6 M/s); p99 42 ns; worst observed 2.38 microseconds including clock-read and OS scheduling overhead. A cross-machine run on a separate Linux build box reproduces the same shape (tens of ns typical, single-digit microsecond worst case, tens of millions of decisions per second), confirming the bound is not an artefact of one machine.

Claim F: It fits a real flight microcontroller with no dynamic memory. Evidence (`make embedded`, `llvm-size`): guard logic ~5.6 KB base gate / ~26 KB full multi-mode engine `.text`; `.data` = 0, `.bss` = 0 on `thumbv7em-none-eabihf`. Onboard signing adds 5.0 KB (audit chain) plus 2.8 KB (hand-rolled SHA-256). Cross-compiles to `thumbv7em-none-eabihf` and `riscv32imac-unknown-none-elf` with no OS, libc, or allocator.

Claim G: Decisions are tamper-evident. Evidence: every decision is folded into an HMAC-SHA256 chain with O(1) state. Forge, reorder, delete, and truncate attempts are all rejected (`make mission`, `make provenance`).

Claim H: The provenance is implementation-independent. Evidence: a second, separate stack (Python engine plus standard-library HMAC) reproduces the signed audit head byte-for-byte from the same decision log (`verify_chain.py`), VERIFIED. The signature does not depend on trusting the flight code alone.

6. Ground analysis validation

- **Propagator agreement.** Vaelon's SGP4 propagation matches the reference `sgp4` library to mean 0.000 km, max 0.001 km over 1,156 samples (`validate_sgp4.py`).
- **Independent re-screen.** 25 of the current week's highest-risk public conjunctions (SOCRATES) were re-propagated from live CelesTrak elements, screened by the real engine, and signed into one chain. Agreement with the public catalogue is sub-km on close-geometry events and diverges on multi-day fast-crossing predictions, the expected and honest behaviour, and the argument for continuous re-screening against fresh state.
- **Robustness dynamics validated independently.** The Clohessy-Wiltshire relative orbital dynamics used as the independent ground truth for robustness testing were checked against a reference RK45 integration of the canonical Hill equations (scipy), agreeing to sub-micron over a full orbit across 500 random states (`validate_reality.py`). Graded against those validated dynamics, on conjunctions with adequate warning,

the guard avoids 99.4% of collisions (n=5,000, 95% CI [99.1%, 99.6%]; `robustness_cw.py`).

7. Verification and validation approach

Vaelon's assurance uses layered, independent methods so that no single technique is load-bearing:

1. Formal proof over the whole input domain (model checking).
2. Differential equivalence against an independent reference engine.
3. Property-based fuzzing for panic-freedom and fail-closed behaviour.
4. Conventional unit and behavioural tests.
5. Bench measurement of timing and binary size on target triples.
6. Cross-implementation reproduction of the cryptographic provenance.

`[FOUNDER DECISION]` The full proof harnesses live in `src/proofs.rs`. Decide how much of the proof construction (as opposed to the proven properties, stated above) is exposed in an external submission, per the IP-redaction rule. This summary states *what* is proven, not *how* the proofs are built.

8. Standards alignment

Vaelon's evidence is organised to map onto recognised space software assurance standards. The intended reference set:

- ECSS-E-ST-40C (space software engineering): requirements, design, verification.
- ECSS-Q-ST-80C (software product assurance): process, verification, criticality.
- ECSS-Q-ST-30C and ECSS-Q-ST-40C (dependability and safety): hazard control, fail-safe behaviour.
- ECSS-E-HB-40-02A (software engineering guidance) as informative background.

Indicative mapping of evidence to assurance area:

assurance area	Vaelon evidence
Requirements definition	the 11 invariants (Section 4), each with a severity
Design for safety	fail-closed, independence, determinism, no dynamic memory
Verification (formal)	5 proven theorems over the whole input domain (Claim A)
Verification (test)	81 tests, ~100k-input property fuzzing, 20k differential (B, C, D)
Timing and resource budgets	bounded latency and zero-heap footprint (E, F)

assurance area	Vaelon evidence
Dependability and fault handling	INV-FAULT, fail-closed, worst-verdict-wins
Data integrity and traceability	tamper-evident signed audit chain (G, H)

`[NEEDS ASSURANCE REVIEWER]` The exact clause-level mapping (specific ECSS clause numbers, and the criticality-category tailoring appropriate for a runtime monitor) should be confirmed by a qualified software-assurance engineer before formal submission. The table above is an engineering mapping, not a certified compliance matrix.

9. Limitations and open items

- No flight heritage; all evidence is ground-based.
- No independent third-party audit yet.
- Ground re-screen is a soundness demonstration on public elements, not validation against operator-authoritative CDMs.
- The onboard AI is out of scope by design; Vaelon bounds actions, it does not verify the AI model.
- The predicted-state model driving the invariants is the current limiting factor on fidelity; higher-fidelity dynamics and power/thermal models are on the path.
- **Collision avoidance carries two architectural requirements**, surfaced by grading against validated orbital dynamics: (a) detection needs a ground-computed predicted miss distance (a CDM), not onboard linear extrapolation from instantaneous range-rate, which is weak against curved relative motion; and (b) the evasive maneuver must be along-track, not radial (a radial impulse in Clohessy-Wiltshire only oscillates). Both are standard, but are stated here rather than assumed.

10. Path to TRL 5-6

1. Independent verification and validation of the results in this case against ECSS, with a named V&V partner.
2. Hardware-in-the-loop: run the `no_std` core on real flight-representative silicon (STM32 / Cortex-M and a RISC-V target), converting bench numbers into on-target measurements.
3. Validate the ground screen against an operator or Space-Track CDM feed, not only public elements.
4. Integrate with a flight-software framework (cFS or F prime) over the C ABI.
5. In-orbit demonstration as a hosted payload.

11. Reproducibility appendix

The full command-to-result table is `vaelon-core/ASSURANCE.md`. Headline commands:

```
make verify        # 5 formal theorems, cargo kani
make test          # 48 behavioural tests
make differential  # 20,000-scenario equivalence to the reference engine
make bench         # latency distribution
make embedded      # binary size on thumbv7em-none-eabihf
make provenance    # tamper-evidence of the signed chain
python validate_sgp4.py   # propagator agreement vs reference sgp4
python verify_chain.py    # cross-stack reproduction of the signed head
```

Prepared from the reproducibility baseline dated 2026-07-25. To turn this into a submission-ready PDF: `pandoc vaelon-safety-case-summary.md -o vaelon-safety-case-summary.pdf`. Before submission, resolve the two flagged items (proof-exposure decision; assurance-reviewer confirmation of clause mappings) and, if available, replace the public-element re-screen with an operator-CDM validation.