Human-First Mechanical Leadership Standard v1.5

Core Operating Standard

Document type: Standard and Operating Guidance (Baseline)
Version: 1.5 (Stable)
Supersedes: v1.4
Maintained by: Dennis Knight — LeadershipPlaybooks.com
Last updated: 03-22-2026

---

0. Document Intent

This document is the **core operating standard** for HFML.

It is written to do three jobs:

1. Define HFML terms and boundaries so teams do not reinterpret them into generic process theatre.
2. Set the minimum operational baseline (what "running HFML" actually means).
3. Define the required artefacts (proof) so HFML remains scalable, auditable, and repeatable.

This is not the playbook. This is the baseline guidance that the playbook sits on top of.

---

1. Purpose

HFML exists to solve recurring failure patterns in teams and organisations:

- Delivery relies on a few people catching everything (hero dependency)
- Work is busy but momentum is low (drag)
- Processes accumulate and remain long after they stop being useful (decay)
- Bad news is delayed or hidden because truth feels unsafe (signal failure)

HFML treats these as **system design problems**, not "people problems".

---

2. Scope

HFML applies to:

- Leadership systems (how work is prioritised, executed, governed)
- Operational systems (how work is routed, tracked, handed over, reviewed)
- Team systems (how signals, friction, truth and learning are handled)

HFML does **not** prescribe a tool stack. It prescribes behaviours, constraints, and evidence.

---

3. Definitions

3.1 Mechanical

Mechanical means reliable by design. A mechanical system produces consistent outcomes with minimal dependence on memory, heroics, or individual temperament.

3.2 Human-First

Human-first means mechanisms exist to:

- reduce cognitive load
- reduce interruptions
- reduce rework
- increase safety to surface truth

If a mechanism increases surveillance vibes, shame, or admin without value, it fails HFML.

3.3 Rail

A **Rail** is a mechanism that makes the right outcome happen by default.

A rail is not:

- a reminder
- a “please do this” policy
- a document that is ignored

A rail is:

- enforced by the system, or
- so low-friction it is the natural default

3.4 Drag

Drag is energy burned with no corresponding value. Drag can be mechanical (slow systems, duplicated work, rework) and/or human (unclear priorities, fear, politics, decision bottlenecks).

3.5 Bus Factor

Bus Factor is the resilience of a critical outcome to a key person being absent.

- 1 = single point of failure
- 3 = minimum healthy resilience on critical paths
- 5+ = strong resilience through rails + documentation + cross-coverage

3.6 Zombie Process

A **Zombie process** is a process that persists after it stops delivering value.

HFML rule: **zombie processes are prohibited**. If ignored, delete or rebuild. Do not keep "because we've always done it".

---

4. HFML Baseline Principles

These principles define the non-optional constraints that preserve HFML integrity.

Principle 1: Protect human energy

If a task is repetitive and low-variance, either:

- automate it
- simplify it
- delete it

Principle 2: Rails beat reminders

If the system relies on memory and motivation, it is fragile by design.

Principle 3: Failure is data

Breakage indicates missing rails or decayed rails. Punishing failure destroys signal quality.

Principle 4: Reduce hero dependency

No critical outcome should depend on one person's presence or memory.

Principle 5: Serve the human

Mechanisms must reduce stress, confusion, or errors. If a mechanism creates friction without value, it must be rebuilt or removed.

5. Adoption Modes (Implementation Baseline)

HFML is implemented through adoption modes to prevent “all-or-nothing” failure.

5.1 Core Engine (minimum HFML)

A team is considered “running HFML” when it:

- uses HFML definitions consistently
- runs the HFML lifecycle continuously (Section 6)
- maintains the required proof artefacts (Section 8)
- runs weekly triage (Section 9)
- runs a monthly Zombie Hunt review (Section 9)

5.2 Boosted Mode (recommended)

Boosted Mode adds:

- explicit Bus Factor targets for critical paths
- Drag Index pulse checks
- Vital signs monitoring (small, non-toxic indicators)
- standardised templates for proof artefacts

5.3 Custom Mode (context extensions)

Custom Mode includes:

- industry constraints (regulated environments)
- extended controls and audit mappings
- automation integrations

Custom Mode must not violate the baseline principles.

6. HFML Lifecycle (Operating Loop)

HFML is maintained through a continuous lifecycle. This is the minimum operational loop for system health and decay prevention.

Phase 1: Drag Radar

Purpose: Identify the highest-cost drag and the most fragile outcomes.

Triggers include:

- repeated incidents
- repeated questions
- consistent missed deadlines
- handover failures
- visible burnout
- “busy but not shipping”

Technique: **Mars Diagnostic**

Ask: “If a key person got dropped on Mars today, what breaks immediately?”

Required proof: Drag Audit (see Section 8)

Phase 2: Build the Rail

Purpose: Convert a fragile outcome into a mechanism.

Rail requirements:

- **Default alive:** works without hero presence
- **Low friction:** easy to follow, hard to bypass
- **Observable:** you can tell if it is working
- **Outcome-aligned:** it protects a specific outcome, not “general good practice”

Required proof: Rail Specification (see Section 8, Appendix A)

Phase 3: Human Test

Purpose: Ensure the mechanism supports humans rather than policing them.

Human Test checks:

- does it reduce cognitive load?
- does it reduce interruptions?
- does it remove ambiguity?
- does it feel safe and supportive?

Required proof: Mini Playbook (see Section 8)

Phase 4: Zombie Hunt

Purpose: Prevent decay and delete dead process.

Rules:

- ignored rails are either unnecessary or unusable
- delete first, rebuild second

- keep the system lean

Required proof: Decay Check (see Section 8)

---

7. Roles and Responsibilities (Baseline Ownership)

Roles are defined to prevent “everyone owns it” failure.

7.1 Architect (System Owner)

Accountable for system reliability.

Responsibilities:

- ensures lifecycle cadence runs
- prioritises which rails get built
- approves Rail Specifications
- removes organisational blockers

7.2 Operator (Delivery Owner)

Accountable for output and signal quality.

Responsibilities:

- executes high-variance work
- uses rails as designed
- surfaces drag honestly
- reports bypass patterns and friction early

7.3 Mechanic (Maintenance Owner)

Accountable for system maintenance and proof integrity.

Responsibilities:

- maintains documentation and templates
- runs Zombie Hunt rituals
- monitors decay signals
- ensures proof artefacts stay current

---

8. Required Proof Artefacts (Minimum Set)

HFML requires receipts. This prevents HFML becoming opinion-led and unscalable.

8.1 Drag Audit (Phase 1 proof)

Minimum contents:

- Top 5 drag points (ranked)
- Top 3 fragile outcomes
- One priority: “next rail to build”
- Date, owner (Architect), contributors (Operators)

8.2 Rail Specification (Phase 2 proof)

Minimum contents:

- outcome protected
- trigger input
- mechanism steps / constraints
- owner (Architect), maintainer (Mechanic)
- success definition
- failure modes and fallback

A complete example is provided in Appendix A.

8.3 Mini Playbook (Phase 3 proof)

Minimum contents:

- why this rail exists
- how to use it in 60 seconds
- what “good” looks like
- where to report friction

8.4 Decay Check (Phase 4 proof)

Minimum contents:

- rails used vs ignored
- friction notes
- deletion list
- rebuild list
- next Drag Radar trigger

9. Cadence (Minimum Operating Rhythm)

Weekly: Triage loop (15 minutes)

Minimum agenda:

- what broke?
- what repeated?
- what created the most drag?
- what rail do we build next?

Output:

- update Drag Audit queue (Phase 1)
- assign owner for the next Rail Specification (Phase 2)

Deploy happens asynchronously (Phase 3): once a rail is built, the Mini Playbook is published immediately and the Human Test is completed during the first week of usage.

Monthly: Zombie Hunt (60 minutes)

Minimum agenda:

- which rails are ignored?
- which rails create friction?
- what gets deleted this month?
- what must be rebuilt?

Output:

- Decay Check (Phase 4)
- deletion actions assigned

Quarterly: Vital signs + resilience review (60 minutes)

Minimum agenda:

- Bus Factor review on critical paths
- Drag Index pulse results (if used)
- repeat failure patterns and major refactors

Output:

- re-prioritised rail backlog
- updated resilience focus list

10. Vital Signs (Small, Non-toxic Indicators)

HFML uses minimal indicators to guide attention, not punish humans.

Recommended:

- Bus Factor on critical paths: target 3+
- Rework rate: target under 5%
- Meeting load: target under 15% capacity
- Time to unblock: decreasing trend
- Interrupt frequency: aim to reduce

---

11. Drag Index (Optional, Boosted Mode)

A quarterly pulse survey to detect friction early.

Example prompts (1–5 scale):

- I know what “good” looks like this week.
- I can raise bad news without fear.
- I spend energy on outcomes, not bureaucracy.
- Handoffs are predictable.
- I can find what I need without chasing people.

If Drag Index < 3.0, trigger Drag Radar.

---

12. Implementation Rules (Baseline Controls)

Rule 1: Delete beats add

If you only add rails and never delete, you create process debt.

Rule 2: Low friction wins

If it is annoying, people will bypass it.

Rule 3: Truth must be safe

If truth is punished, Drag Radar becomes blind.

Rule 4: No hero worship

Reward reliable systems, not late-night rescues.

Rule 5: Close the loop

Every Drag Radar must lead to:

- a rail built, or
- a process deleted

---

13. Quickstart (Minimum Viable HFML, 5 Days)

Day 1: Run Drag Radar with the Mars Diagnostic

Day 2: Choose one fragile outcome, write the Rail Specification

Day 3: Publish the Mini Playbook and deploy

Day 4: Run the Human Test and collect friction

Day 5: Zombie Hunt, delete one useless process

This is the smallest credible HFML implementation.

---

14. Common Misinterpretations (Guardrails)

“A checklist is a rail”

Only if enforced or naturally used by default. Otherwise it is a zombie.

“HFML is just process improvement”

HFML is process improvement constrained by:

- human energy protection
- resilience targets
- deletion discipline
- proof requirements

“HFML slows teams down”

Initial structure reduces later chaos by lowering rework, interruptions, and hero dependency.

---

15. Version Control and Change Log

v1.0 (Foundational Draft)

- Defined HFML scope and core terms
- Introduced rails, drag, bus factor
- Established lifecycle loop

v1.1 (Operational Clarity)

- Added proof artefacts to prevent theatre
- Introduced cadence baseline
- Introduced vital signs guidance

v1.2 (Human-First Safeguards)

- Added explicit human-first constraints to prevent surveillance drift
- Added truth-safety as a signal quality requirement
- Strengthened Zombie Process deletion discipline

v1.3 (Baseline Standardisation)

- Formalised the standard structure and governance baseline
- Locked terms and lifecycle structure
- Introduced roles and responsibility mapping
- Included compliance-style language that improved precision but reduced approachability

v1.4 (Readability Modernisation)

- Introduced Adoption Modes (Core / Boosted / Custom)
- Rewrote key sections into plain English
- Renamed lifecycle phases for memorability
- Renamed roles to remove industrial bias
- Added Appendix A Rail Specification example

v1.5 (Canon Alignment)

Reason:

- v1.4 retained old role names (Designer, Driver, Tuner) and artefact names (Drag Snapshot) that diverged from the book's canon terminology.

Changes:

- Roles renamed: Designer → Architect, Driver → Operator, Tuner → Mechanic
- Artefact renamed: Drag Snapshot → Drag Audit
- Rail Spec → Rail Specification (full name used consistently)
- Lifecycle phase parentheticals removed (Triage, Design, Deploy, Review)
- All references updated to match book canon

Core integrity statement:

- No change to HFML principles, lifecycle engine, or proof requirements.
- This is a terminology alignment pass to ensure book and standard speak the same language.

Appendix A: Example Rail Specification (Baseline Reference)

Rail Name: Central Intake Rail
Version: 1.0
Owner (Architect): [Name / Role]
Maintainer (Mechanic): [Name / Role]
Primary Users (Operators): [Team / Function]
Date: [YYYY-MM-DD]
Status: Active

A1. Outcome Protected

All work requests enter the system through a single visible lane so nothing is lost, duplicated, or prioritised via private DMs.

A2. Problem Statement (Drag Removed)

Current symptoms:

- requests arrive via Slack/Teams DMs, email, hallway chats, “quick favours”
- prioritisation becomes political (loudest wins)
- tracking is fragmented
- Operators are interrupted constantly
- important work slips because it never entered the system

A3. Trigger Input

A request is anything that consumes team time, including:

- incidents/support
- bugs/defects
- feature requests
- improvement ideas
- “can you just…” asks

A4. Rail (Mechanism Design)

Rule: If it’s not in Intake, it doesn’t exist.

System of record: choose one board/queue as Intake (Jira/ServiceNow/Planner/Trello).

Mechanism steps:

4. Publish the single intake link and pin it in the primary team channel.

5. DM deflection: if an Operator receives a DM request, they reply with the intake link.
6. Require a priority tag in intake:
    - P1: service down / safety issue
    - P2: time-bound
    - P3: normal
    - P4: idea/backlog
7. Weekly triage assigns:
    - accept/reject
    - owner
    - next action
8. Intake visibility is default (transparent queue).

A5. Constraints (Default Alive)

- work does not start without an intake item (except true P1 emergencies)
- every intake item must have:
    - owner
    - priority
    - brief definition of done

A6. Success Definition

This rail is working when:

- DM interruptions drop
- fewer surprise requests appear mid-week
- stakeholders can see queue position and status
- prioritisation becomes transparent
- Operators report reduced drag and improved focus

A7. Failure Modes and Fallbacks

- **Bypass via DMs increases**

Fix: Architect reinforces rule publicly. Mechanic improves quick-reply templates.

- **Intake becomes a dumping ground**

Fix: weekly triage rejects low-value items and clarifies what “in scope” means.

- **Everyone marks P1**

Fix: tighten P1 criteria and require confirmation during triage.

A8. Human Test (Safety Check)

This rail must feel protective:

- it reduces interruptions

- it makes prioritisation fair
- it prevents lost work

It must not feel like policing.

A9. Proof Outputs

- Drag Audit linkage: “DM interruptions” trend
- Rail Specification: this document
- Mini Playbook (60 seconds): how to submit and tag requests
- Decay Check signals:
    - bypass count (DMs)
    - intake items without owner
    - average time-to-triage

