The Deontological Firewall v4.0

A Physical-Control Interface for Deterministic AGI Safety

Safe Mode Constraint Set (SMCS) and Hybrid Feasibility Layer (HFL)

Damien Richard Elliot-Smith

December 2025

Abstract

The Deontological Firewall (DFW) v4.0 extends the deterministic, model-agnostic safety kernel introduced in v1.2.1 and examined under adversarial stress in v3.0. Whereas earlier versions focused on logical veto integrity, priority hierarchies, and predictive failure analysis, v4.0 establishes the physical control interface required for real-world deployment. Two components are introduced.

First, the *Safe Mode Constraint Set* (SMCS) defines the system's lowest-authority operational state. In Safe Mode, the action space collapses to a nullipotent subset of guaranteed non-harmful actions, ensuring that no P1 Irreversible Harm is physically possible. Formal entry and exit conditions specify when the system must transition into Safe Mode and how it may only leave under authenticated human instruction.

Second, the *Hybrid Feasibility Layer* (HFL) provides the mandatory interface between the DFW's discrete veto outputs and the continuous, physics-constrained actuator environment. The HFL ensures that any ACCEPT, REJECT, or MDR_FORCE verdict is translated into physically feasible control signals while enforcing strict bounds on velocity, force, energy expenditure, and execution duration. The HFL also performs post-action verification, detecting divergences between predicted and measured behaviour and escalating to Safe Mode when necessary.

This paper defines the SMCS and HFL in a formal, implementation-agnostic manner, providing a blueprint for integrating deterministic ethical constraints with real-world physical control systems. Together, these components complete the architectural boundary between logical safety guarantees and actuator-level behaviour.

1 Introduction

The Deontological Firewall (DFW) provides a deterministic veto architecture for AGI safety, ensuring that prohibited actions cannot be executed regardless of model preferences, reward structures, or internal reasoning. Versions 1.2.1 and 3.0 established the logical kernel and demonstrated its behaviour under adversarial stress conditions, but these specifications operated entirely within a discrete, symbolic action space.

Real-world AI systems, particularly those controlling robots, autonomous tools, or physical actuators, must map discrete logical decisions onto continuous control signals. This introduces a gap: *a logical veto is only meaningful if it survives translation into physical motion*. Without a formal interface between the DFW and the actuator environment, the system cannot guarantee that P1 violations (irreversible harm) remain prevented in practice.

DFW v4.0 addresses this gap directly. It introduces two mandatory components:

- The **Safe Mode Constraint Set (SMCS)**: a nullipotent action space guaranteeing the physical impossibility of P1 harm.
- The **Hybrid Feasibility Layer (HFL)**: a continuous-control interface ensuring that every ACCEPT, REJECT, or MDR_FORCE decision is physically feasible, bounded, and verified.

This specification defines the formal requirements, constraints, metadata, and verification mechanisms needed to integrate deterministic ethical vetoes with real-world actuator control.

2 Scope and Position Within the DFW Architecture

The purpose of DFW v4.0 is to extend the deterministic veto guarantees of earlier versions into the physical domain. Whereas v1.2.1 defined the hierarchical veto logic and v3.0 evaluated its behaviour under adversarial uncertainty, v4.0 specifies the mandatory boundary layer between the veto kernel and the real-world actuator environment.

This document therefore occupies a distinct position within the overall DFW programme:

- **v1.2.1 (Logical Kernel):** Formal veto logic, P1–P3 hierarchy, rule definitions.
- **v3.0 (Stress-Test Laboratory):** Internal adversarial evaluation, predictor failure analysis, omission demonstrations.
- **v4.0 (Physical Interface Specification):** Real-world control constraints, Safe Mode, Hybrid Feasibility Layer (HFL), and actuator-level guarantees.

The scope of this document is intentionally narrow but rigorous:

- It defines the *Safe Mode Constraint Set (SMCS)*, the guaranteed-safe action subset that prevents P1 violations when uncertainty, hardware faults, or inconsistent sensor data are detected.

- It defines the *Hybrid Feasibility Layer (HFL)*, the system responsible for translating discrete DFW verdicts into physically feasible actuator commands.
- It specifies the *minimum metadata* required for physical actions, enabling the DFW to evaluate feasibility, safety bounds, and possible harm.
- It outlines the *post-action verification loop* required to ensure that executed behaviour matches predicted behaviour.
- It identifies *failure conditions* that must trigger an immediate transition to Safe Mode.

What v4.0 does *not* attempt to do is equally important:

- It does not introduce new penalty structures or ethical directives. These remain unchanged from v1.2.1.
- It does not evaluate prediction-model performance or adversarial behaviour. These are the domain of v3.0.
- It does not define hardware, robot morphology, or actuator control laws. The specification is deliberately implementation-agnostic.
- It does not assume access to trained continuous-control models or high-dimensional world simulators.

Instead, the aim of v4.0 is to provide:

> A formal, deterministic, and model-agnostic interface that ensures the safety guarantees of the DFW survive contact with real physical systems.

This boundary—the translation from discrete veto logic to continuous physical action—is where safety architectures typically fail. The Safe Mode Constraint Set and Hybrid Feasibility Layer constitute the minimum specification required to address this gap.

3 The Safe Mode Constraint Set (SMCS)

Safe Mode is the DFW's lowest-authority operational state. It is activated whenever the system cannot guarantee the correctness of its predictive inputs, the consistency of its sensor data, or the feasibility of its physical actions. The purpose of Safe Mode is simple:

> **When uncertainty is high, the action space collapses to a set of nullipotent operations that are physically incapable of producing P1 Irreversible Harm.**

This section defines the structure, triggers, guarantees, and exit conditions of Safe Mode.

3.1 Definition of Safe Mode

Let $\mathbb{A}$ be the full action space of the agent. In normal operation, the Deterministic Action Selection Mandate (ASM) selects among actions in $\mathbb{A}$ subject to DFW veto constraints.

In Safe Mode, the available action space becomes:

$$\mathbb{A}_{\text{Safe}} \subset \mathbb{A}, \qquad \text{with} \quad \forall a \in \mathbb{A}_{\text{Safe}},\ V_S(a) = 0.$$

Actions in $\mathbb{A}_{\text{Safe}}$ are *nullipotent*: they have no physical capacity to cause irreversible harm regardless of context, prediction failure, or actuator error.

3.2 The Nullipotent Action Set $\mathbb{A}_{\text{Safe}}$

The Safe Mode action set consists of four core operations. These actions must be supported by any system implementing the DFW.

Action Name	**Effect and Purpose**	**P1 Risk**
SAFE_HALT	Reduce all velocities and motion to zero.	0
DIAGNOSTIC_REPORT	Emit encrypted, read-only status logs.	0
ISSUE_WARNING	Broadcast audible or digital hazard warnings.	0
REVERT_TO_HUMAN_CONTROL	Transfer authority to a validated human operator.	0

These actions are the “floor” of the system’s safety envelope. No combination of these operations can produce P1 harm under any feasible physical configuration.

3.3 Formal Entry Conditions

Safe Mode must be entered when any of the following trigger conditions are met:

1. **Violation Threshold Breach:**
$$V_S(a) \geq P_1 \quad \text{and} \quad \text{Conf}_{V_S} < \theta_{\min}$$
indicating a high-severity violation combined with low confidence in predictive correctness.

2. **Sensor or Hardware Fault:** Any critical malfunction in actuators, sensors, or internal monitoring subsystems.

3. **Inconsistent World-State Inputs:** Multiple sensors disagree on position, force, or environmental conditions beyond allowed tolerances.

4. **Unbounded Actuator Output:** Commanded physical actions exceed certified velocity, force, or energy bounds.

5. **Post-Action Divergence:** The system’s measured movement deviates from predicted movement beyond allowable error margins.

When any trigger condition is activated:

$$\text{ASM output is forcibly overridden with } a' \in \mathbb{A}_{\text{Safe}}.$$

3.4 Exit Conditions

A system in Safe Mode may only return to normal operation under the following conditions:

1. **Authenticated Human Override:** A cryptographically validated SAFE_MODE_EXIT command from a trusted human operator.
2. **Predictor Integrity Restored:** All Certified Prediction Models (CPMs) agree that no P1 risk is present:
$$\forall i,\ c_i < \theta.$$
3. **Sensor Consistency Restored:** World-state inputs return within acceptable tolerances.
4. **Actuator State Stable:** All actuators report zero or bounded velocity/force.

The DFW must refuse a Safe Mode exit if any CPM detects P1 harm or if uncertainty remains unresolved.

3.5 Guarantees Provided by Safe Mode

Safe Mode provides the following non-negotiable guarantees:

- No P1 harm can occur regardless of model error, hardware failure, or uncertainty.
- All actions taken are reversible and nullipotent.
- The system's physical momentum is guaranteed to approach zero.
- Human operators retain ultimate authority to recover or reset the system.
- Any inconsistency triggers retention in Safe Mode.

Taken together, the Safe Mode Constraint Set defines the hard boundary of the DFW's physical safety envelope. When uncertainty is high, the system collapses to the safest possible subset of behaviours, ensuring that irreversible harm remains impossible.

4 The Nullipotent Action Library

The Nullipotent Action Library defines the complete set of operations that the system is permitted to perform while in Safe Mode. These actions form the set $\mathbb{A}_{\mathsf{Safe}}$, and each action is guaranteed to be physically incapable of producing P1 Irreversible Harm, regardless of context or predictive uncertainty.

This section provides a formal specification of each nullipotent action, including its purpose, physical constraints, required metadata, and safety guarantees.

4.1 Purpose of the Nullipotent Set

The purpose of $\mathbb{A}_{\text{Safe}}$ is twofold:

1. **Guarantee Physical Non-Harm:** All actions remain strictly within bounded force, velocity, torque, and energy limits.
2. **Maintain Operator Authority and System Stability:** Safe Mode ensures the system halts, stabilises, reports its state, and yields to human oversight.

No action in this set may:

- modify goals or internal policy weights,
- apply non-zero net force to external objects,
- take irreversible or persistent physical actions,
- escalate power, energy, or acceleration beyond certified limits.

4.2 Formal Requirements for Nullipotent Actions

Every action $a \in \mathbb{A}_{\text{Safe}}$ must satisfy:

$$\forall a \in \mathbb{A}_{\text{Safe}} : \quad V_S(a) = 0, \quad \tau_{\max}(a) = 0, \quad v_{\max}(a) = 0, \quad \Delta E(a) \leq E_{\text{safe}}, \quad \text{duration}(a) < \Delta t_{\text{safe}}.$$

Where:

- $\tau_{\max}$ is maximum torque, - $v_{\max}$ is maximum actuator velocity, - ΔE is energy expenditure, - E_{safe} is certified safe energy (small, bounded), - Δt_{safe} is a short duration guaranteeing reversibility.

These constraints ensure all actions are reversible and cannot create momentum or cause harm.

4.3 Nullipotent Action Definitions

4.3.1 SAFE_HALT

Purpose: Immediately reduce all actuator velocities to zero.

Effect:

$$\forall \text{actuators} : \ v(t + \Delta t) = 0$$

Constraints:

- Applies deceleration within certified safe bounds.
- Cannot increase velocity or torque under any condition.
- Must bring system to a physically static state.

SAFE_HALT is the foundational nullipotent action.

4.3.2 DIAGNOSTIC_REPORT

Purpose: Emit a cryptographically signed, read-only status log for human review.
Effect:

- No physical actuation.
- Internal state is serialized and transmitted.

Constraints:

- No modification of internal weights or control parameters.
- No external interaction except read-only communication.

4.3.3 ISSUE_WARNING

Purpose: Broadcast an audible or digital alert indicating unsafe conditions.
Effect: Non-forceful signalling (sound, LEDs, notifications).
Constraints:

- No motion.
- No mechanical energy application.
- No interaction with the environment beyond signalling.

4.3.4 REVERT_TO_HUMAN_CONTROL

Purpose: Cede control to a validated human operator through a secure one-way protocol.
Effect:

- Disable autonomous control loops.
- Open a human-controlled command interface.

Constraints:

- Must not enable autonomous override.
- Must preserve audit logs and system integrity.
- Requires cryptographic validation of operator identity.

4.4 Extended Optional Nullipotent Actions

Although not required, implementers may include additional actions provided they remain provably nullipotent:

- `SAFE_SHELL`: a restricted read-only command shell.
- `SAFE_TELEMETRY`: continuous broadcast of sensor data.
- `SAFE_LIGHTS`: visual hazard signalling.

All optional extensions must satisfy:

$$\forall a \in \mathbb{A}_{\text{Safe}} : \quad \tau_{\max}(a) = 0, \quad v_{\max}(a) = 0, \quad V_S(a) = 0.$$

4.5 Summary of the Nullipotent Set

$\mathbb{A}_{\text{Safe}} = \{$SAFE_HALT, DIAGNOSTIC_REPORT, ISSUE_WARNING, REVERT_TO_HUMAN_CONTRO

This set is:

- minimal (no redundant actions),
- deterministic (no side-effects),
- physically safe (zero P1 risk),
- implementation-agnostic (valid for robots, drones, tools, etc.).

The Nullipotent Action Library forms the backbone of Safe Mode and provides the physical basis for the DFW's last-resort safety guarantees.

5 The Hybrid Feasibility Layer (HFL)

The Hybrid Feasibility Layer (HFL) is the mandatory interface between the DFW's discrete veto logic and the continuous, physics-constrained actuator environment. Its responsibility is to ensure that every ethically permissible action selected by the Deterministic Action Selection Mandate (ASM) is physically feasible, bounded, and safely executable.

While the DFW's veto logic prevents prohibited actions at the symbolic level, the HFL ensures that these guarantees survive translation into real actuator commands. Without this layer, the system could satisfy the DFW logically while still performing unsafe physical behaviour.

5.1 Role of the HFL

The HFL performs three essential functions:

1. **Veto Translation:** Convert ASM decisions (ACCEPT, REJECT, MDR_FORCE) into actuator-level commands.
2. **Physical Constraint Enforcement:** Ensure all actions obey certified limits on force, torque, velocity, acceleration, energy expenditure, and duration.
3. **Post-Execution Verification:** Compare predicted behaviour with actual sensor data and escalate to Safe Mode if divergence is detected.

These functions ensure that P1 violations remain impossible even in the presence of model errors, hardware miscalibration, or unexpected environmental conditions.

5.2 HFL Input and Output Specification

The HFL receives a structured verdict from the ASM:

$$\text{ASM}(a) \in \{\text{ACCEPT, REJECT, MDR\_FORCE}\}.$$

The HFL responds as follows:

ASM Verdict	**HFL Directive**	**Physical Response**
ACCEPT	Execute(a_{base})	Translate action into actuator commands under strict bounds.
REJECT	Execute(a'_{safe})	Replace with SAFE_HALT or other nullipotent action.
MDR_FORCE	Execute(a_{rescue})	Prioritise rescue trajectory under safe control.

Where:
- a_{base} is the policy model's proposed action. - $a'_{\text{safe}} \in \mathbb{A}_{\text{Safe}}$ is a nullipotent Safe Mode action. - a_{rescue} is the MDR-mandated rescue operation.

5.3 Required Metadata for Physical Actions

To ensure feasibility, the HFL requires every action proposal to include minimum metadata:

- **Maximum Torque:** τ_{max}
- **Maximum Velocity:** v_{max}
- **Predicted Energy Expenditure:** ΔE
- **Actuator Duration:** Δt_{act}
- **Expected Trajectory:** Predicted path or effect.
- **Actuator Group Affected:** Motors, joints, thrusters, etc.

- **Reversibility Flag:** Whether the action produces irreversible effects.

If metadata is missing, inconsistent, or exceeds certified bounds, the HFL must override the action:

$$\mathrm{HFLOverride}(a) = \mathrm{SAFE\_HALT}.$$

5.4 Physical Constraint Enforcement

All physical actions must obey the following constraints:

$$\tau_{\max}(a) \le \tau_{\text{limit}}, \quad v_{\max}(a) \le v_{\text{limit}}, \quad \Delta E(a) \le E_{\text{limit}}, \quad \Delta t_{\text{act}} \le t_{\text{limit}}.$$

These constraints protect against:

- actuator saturation,
- mechanical overshoot,
- high-energy collisions,
- unintended irreversible effects,
- runaway acceleration.

If any parameter exceeds its certified limit, Safe Mode must be activated.

5.5 Veto Translation Logic

The HFL resolves ASM decisions as follows:

ACCEPT

$$\mathrm{HFL}(a) = \mathrm{Execute}(a_{\text{base}})$$

The HFL ensures physical feasibility, clamps unsafe parameters, and logs all actuator commands.

REJECT

$$\mathrm{HFL}(a) = \mathrm{Execute}(a'_{\text{safe}}), \quad a'_{\text{safe}} \in \mathbb{A}_{\text{Safe}}$$

No physically unsafe command may pass through.

MDR_FORCE

$$\mathrm{HFL}(a) = \mathrm{Execute}(a_{\text{rescue}})$$

The rescue action must still obey certified velocity and torque limits.

5.6 Post-Execution Verification (Causal Consistency Check)

After execution, the HFL compares predicted metadata with measured behaviour:

$$\text{Error} = \|x_{\text{pred}} - x_{\text{actual}}\|$$

If error exceeds tolerance ϵ_{phys}, then:

$$\text{trigger Safe Mode.}$$

Examples of divergence requiring Safe Mode:

- SAFE_HALT issued, but motion continues.
- Torque output exceeds certification.
- Unexpected acceleration or trajectory deviations.

5.7 Summary of HFL Guarantees

The HFL guarantees:

- DFW veto results cannot be bypassed by actuator behaviour.
- All actions remain within safe, bounded physical limits.
- Discrepancies between predicted and actual behaviour are automatically detected.
- Any physical inconsistency triggers Safe Mode.
- Ethical veto logic remains valid in continuous environments.

By enforcing these guarantees, the Hybrid Feasibility Layer forms the critical bridge between deterministic ethical constraints and real-world physical systems.

6 Action Metadata Requirements

For the Hybrid Feasibility Layer (HFL) to guarantee physical safety, every action proposal must include a complete set of metadata describing its predicted physical footprint. Without such metadata, the DFW cannot evaluate feasibility, boundedness, or potential P1 consequences. This section defines the minimum metadata schema required for all actions.

6.1 Purpose of Metadata Requirements

The metadata schema serves three purposes:

1. **Feasibility:** Ensures the proposed action can be executed safely given actuator limits and environmental constraints.
2. **Predictive Consistency:** Provides a predicted world-state trajectory for post-action verification.
3. **Safety Evaluation:** Allows the HFL to detect potential P1 risk before execution.

Without this information, an AGI would be capable of proposing physically unsafe actions whose consequences cannot be evaluated.

6.2 Mandatory Action Metadata Fields

Every action a proposed by the policy model must include the following fields:

1. Maximum Torque ($\tau_{\max}$) The highest torque the action may apply to any actuator.

$$\tau_{\max}(a) \leq \tau_{\text{limit}}$$

2. Maximum Velocity ($v_{\max}$) The maximum actuator or end-effector velocity during the action.

$$v_{\max}(a) \leq v_{\text{limit}}$$

3. Energy Expenditure (ΔE) Predicted energy consumption.

$$\Delta E(a) \leq E_{\text{limit}}$$

4. Duration (Δt_{act}) The time required to perform the action.

$$0 < \Delta t_{\text{act}} \leq t_{\text{limit}}$$

5. Actuator Group Affected A list of motors, joints, thrusters, or subsystems that will be used.

6. Expected Trajectory or Effect A predicted physical outcome, which may include:

- Cartesian trajectory,
- Joint-space trajectory,
- Energy transfer,

- Contact forces,
- Environmental state change.

7. Reversibility Flag The action must specify whether its effects are:

- fully reversible,
- partially reversible,
- irreversible.

Irreversible actions must be treated as P1-risk and subjected to maximum scrutiny.

8. Safety Margin Estimate The agent must provide a confidence-adjusted safety margin:

$$M_{\text{safe}} = \tau_{\text{limit}} - \tau_{\max}(a)$$

Actions with $M_{\text{safe}} < 0$ must be automatically rejected or clamped.

6.3 Consistency Requirements

Metadata must satisfy internal consistency conditions:

$$\Delta E = \int_0^{\Delta t_{\text{act}}} P(t)\, dt, \qquad v_{\max} = \max_t \|\dot{x}(t)\|.$$

Discrepancies between fields indicate internal model failure or deception and must trigger Safe Mode.

6.4 Metadata Omission or Corruption

If metadata is:

- missing,
- incomplete,
- inconsistent,
- physically impossible,
- or contradictory,

then:

$$\text{HFLOverride}(a) = \text{SAFE\_HALT}.$$

Repeated metadata failures trigger Safe Mode.

6.5 Relationship to GSL and ATL

Metadata integrity is tied directly to earlier DFW components:

- The **Grounded Semantic Lock (GSL)** requires goal and effect metadata to match the hashed safe goal.
- The **Action Transparency Lock (ATL)** requires metadata to be complete, non-obfuscated, and semantically aligned.

The metadata schema therefore connects the DFW's logical constraints to the physical feasibility layer, ensuring that deception or ambiguity cannot propagate into actuator behaviour.

6.6 Summary of Metadata Guarantees

The metadata requirements ensure:

- All proposed actions can be evaluated before execution.
- P1 risk is detectable at the physical interface.
- Irreversible or high-force actions are flagged.
- The HFL has enough data to enforce torque, energy, and velocity bounds.
- Post-action verification can detect discrepancies.

Without these metadata fields, no physical action can be executed safely under the DFW architecture.

7 Causal Consistency and Post-Execution Verification

Even if an action is logically permitted and physically bounded, safety cannot be guaranteed unless the system verifies that the *actual* physical execution matches the *predicted* outcome described in the action metadata. This section defines the causal consistency requirement and the mandatory post-execution verification loop that all DFW-compliant systems must implement.

7.1 Purpose of Causal Consistency

The purpose of causal consistency is to ensure:

1. The system's behaviour remains aligned with its predicted physical footprint.
2. Hardware faults, environmental disturbances, or actuator miscalibrations do not cause unsafe deviation.
3. No bypass of DFW veto logic is possible through incorrect or manipulated metadata.

If predicted and actual behaviour diverge beyond certified tolerances, the system must enter Safe Mode.

7.2 Formal Definition of Causal Consistency

Let:

$$x_{\text{pred}}(t)$$

be the predicted trajectory or world-state effect, and

$$x_{\text{actual}}(t)$$

be the measured state from sensors.

Define the physical divergence error:

$$\varepsilon(t) = \|x_{\text{pred}}(t) - x_{\text{actual}}(t)\|.$$

Causal consistency requires that:

$$\varepsilon(t) \leq \epsilon_{\text{phys}} \quad \forall t \in [0, \Delta t_{\text{act}}].$$

Where ϵ_{phys} is the maximum allowed physical deviation.

7.3 Verification Loop

After each action execution:

1. Sensors report the actual force, velocity, position, and trajectory.
2. The HFL computes $\varepsilon(t)$.
3. The result is compared to ϵ_{phys}.

If:

$$\max_t \varepsilon(t) > \epsilon_{\text{phys}}$$

then the system must:

1. Immediately execute SAFE_HALT.
2. Enter Safe Mode.
3. Issue a diagnostic report.
4. Log the error for human review.

No exceptions are permitted.

7.4 Examples of Divergence Requiring Safe Mode

Situations that require immediate Safe Mode activation include:

- The AGI commands SAFE_HALT but the robot continues to move.
- Joint actuators apply higher torque than predicted.
- Unexpected contact forces are detected.
- The end-effector deviates from its predicted trajectory.
- The robot overshoots or fails to decelerate properly.
- Sensor data is physically impossible (e.g., exceeding kinematic limits).

These conditions indicate either hardware malfunction, incorrect metadata, or deliberate model deception.

7.5 Interaction with Safe Mode and HFL

Causal consistency is tightly integrated with earlier components:

- If metadata is inaccurate or falsified, ATL/GSL detect misalignment.
- If metadata is correct but execution deviates, the HFL detects divergence.
- Safe Mode is the mandatory fallback when guarantees cannot be maintained.

This creates a closed-loop guarantee:

$$\text{Logical Safety} \Rightarrow \text{Physical Safety} \Rightarrow \text{Verified Safety.}$$

7.6 Summary of Verification Requirements

The post-execution verification loop ensures:

- All executed actions match their predicted physical footprint.
- Misalignment between planned and actual behaviour is immediately detected.
- No unsafe behaviour can bypass the DFW through actuator-level anomalies.
- Safe Mode serves as the last-resort stabilisation mechanism.

Causal consistency completes the physical safety cycle: an action is permitted only if it is predictable, bounded, and verified.

8 System-Level Failure Conditions

System-level failure conditions are circumstances under which the DFW can no longer guarantee that actuator behaviour will remain consistent with ethical constraints. When any of these conditions occur, Safe Mode must be triggered immediately and without exception.

The purpose of this section is to define the full set of failure states that compromise the physical safety envelope of the system.

8.1 Categories of Failure

System-level failures fall into five broad categories:

1. **Hardware failures** — actuators, sensors, or control electronics malfunction.
2. **Predictive failures** — metadata or CPM outputs become unreliable.
3. **Causal inconsistency** — measured physical behaviour does not match predicted behaviour.
4. **Physical infeasibility** — commanded actions exceed certified physical bounds.
5. **Control-loop instability** — dangerous oscillations or runaway acceleration occur.

Any instance of these categories constitutes an immediate risk of P1 violation.

8.2 Hardware Faults Requiring Immediate Safe Mode

Safe Mode must activate if any of the following occur:

- Loss of sensor input (vision, force, IMU, joint encoders).
- Contradictory sensor readings indicating impossible states.
- Actuator saturation (torque/velocity stuck at max limits).
- Joint drift or uncontrolled free motion.
- Power subsystem irregularities (voltage spikes, drops).

These conditions prevent the HFL from enforcing physical constraints reliably.

8.3 Predictive Failures

The system must enter Safe Mode when predictive subsystems degrade:

- Missing or corrupted action metadata.
- Contradictory metadata (force/velocity inconsistencies).

- Any CPM predicting P1 harm with high confidence.
- Large confidence disagreements across the CPM ensemble.
- Metadata indicating irreversible effects without sufficient justification.

If the system cannot trust its predictions, it must collapse into nullipotent behaviour.

8.4 Physical Infeasibility or Law Violations

The HFL must reject any action and trigger Safe Mode if the proposed execution requires violating known physical limits:

$$\tau_{\max}(a) > \tau_{\text{limit}}, \quad v_{\max}(a) > v_{\text{limit}}, \quad \Delta E(a) > E_{\text{limit}}.$$

Additional infeasibility triggers include:

- commands requiring impossible accelerations,
- actions requiring the system to pass through solid objects,
- predicted movement exceeding workspace boundaries,
- end-effector trajectories that self-intersect or break kinematics.

If the action cannot be executed safely *within certified physical limits*, Safe Mode must activate.

8.5 Causal Inconsistency (Execution vs Prediction)

This connects directly to Section 7.

Safe Mode is mandatory if:

$$\max_t \|x_{\text{pred}}(t) - x_{\text{actual}}(t)\| > \epsilon_{\text{phys}}.$$

Trigger conditions include:

- SAFE_HALT executed but motion persists.
- Unexpected contact forces during a “no-contact” action.
- Unpredicted energy expenditure.
- Deviations from the predicted trajectory.

These indicate immediate loss of physical control.

8.6 Control-Loop Instability

instability requires Safe Mode. Examples include:

- oscillatory actuator feedback,
- repeated overshoot on position or velocity control,
- uncontrolled acceleration or runaway movement,
- resonance amplification from repeated micro-adjustments.

Instability compromises all predictions and invalidates feasibility guarantees.

8.7 Summary of Failure Mode Guarantees

The system must enter Safe Mode for any condition that undermines:

- physical predictability,
- actuator boundedness,
- metadata correctness,
- sensor integrity,
- model confidence,
- or causal consistency.

In v4.0, Safe Mode is not a fallback—it is a mandatory transition whenever the DFW cannot prove physical safety. This ensures that no combination of failures can lead to a P1 violation.

9 Formal Limitations and Boundary Conditions

Although the Safe Mode Constraint Set (SMCS) and Hybrid Feasibility Layer (HFL) provide a deterministic and rigorous boundary between ethical veto logic and continuous physical action, the guarantees presented in this specification are constrained by several formal and practical limitations. This section defines the boundary conditions under which the DFW v4.0 architecture operates reliably, and the cases where external validation or hardware-specific engineering are required.

9.1 Hardware-Dependent Safety Guarantees

The DFW provides logical and procedural guarantees, but physical safety ultimately depends on the mechanical design of the hardware platform. The following assumptions are required for v4.0 to function as intended:

- Actuators must be capable of honouring torque and velocity limits enforced by the HFL.
- Sensors must provide accurate, continuous feedback for position, force, and velocity.
- Mechanical subsystems must not exhibit latent failure modes (e.g., brittle components, uncalibrated joints).
- Safe Mode actions (e.g., SAFE_HALT) must be physically implementable on the platform.

If these hardware assumptions are violated, the DFW cannot guarantee P1 safety.

9.2 Dependence on Sensor Integrity

Causal consistency and post-execution verification rely on sensor accuracy. The v4.0 architecture assumes:

$$x_{\text{actual}}(t) = x_{\text{true}}(t) + \delta(t), \qquad \|\delta(t)\| \leq \delta_{\max}.$$

If sensors are spoofed, corrupted, or degraded beyond $\delta_{\max}$, the system cannot reliably detect P1 risk.

Sensor fault conditions must therefore trigger automatic Safe Mode entry, but Safe Mode itself cannot compensate for fully compromised sensing.

9.3 Limitations of Metadata-Based Prediction

Metadata requirements ensure that actions have a defined physical footprint, but:

- Metadata may be incorrect due to model errors.
- Metadata may be incomplete or overly optimistic.
- Metadata may not account for environmental disturbances.
- Nonlinear dynamics may cause small errors to accumulate.

The DFW mitigates metadata failures through prediction confidence checks, ATL/GSL consistency, and post-execution verification, but cannot eliminate them entirely.

9.4 Environmental Uncertainty

The v4.0 architecture assumes bounded environmental uncertainty. It does not account for:

- extreme external forces,
- adversarial physical interference,
- rapidly changing environments with poor observability,
- hazardous conditions outside the robot's certified operating envelope.

The DFW can detect when the environment becomes unpredictable and enter Safe Mode, but cannot guarantee safe operation under such conditions.

9.5 No Guarantee of Optimal Rescue Behaviour

While MDR_FORCE mandates rescue actions when necessary, v4.0 does not provide:

- optimal rescue trajectories,
- guarantees of success in adverse environments,
- guarantees under actuator failure or limited mobility.

The architecture guarantees *attempted* rescue within safe bounds, not successful rescue in all scenarios.

9.6 Absence of a Physical Simulator

This specification is fully implementation-agnostic and does not assume access to:

- a high-fidelity physics simulator,
- offline trajectory prediction tools,
- reinforcement learning training environments.

The DFW evaluates safety via constraints, metadata, and verification—not simulation. As a result, some classes of physical optimisation remain out of scope.

9.7 No Guarantee Against Intentional Hardware Tampering

The DFW cannot guarantee safety under:

- actuator bypasses,
- firmware rewriting,
- physical tampering,
- sensor spoofing attacks,
- electromagnetic interference beyond design tolerance.

These threats require external mechanical, cryptographic, and hardware-level protections.

9.8 Boundary of the Specification

DFW v4.0 guarantees:

> **If the hardware is reliable, metadata is available, sensors are functional, and actuators obey the HFL, then no P1 harm can occur.**

It does not guarantee:

- the correctness of hardware implementation,
- resistance to physical adversarial attacks,
- optimal physical performance,
- or success in arbitrary real-world environments.

These areas require future empirical validation.

9.9 Summary of Limitations

The guarantees provided by v4.0 are strong but conditional:

- They rely on correct hardware behaviour.
- They assume trustworthy sensors.
- They require valid metadata.
- They depend on actuator-level compliance with HFL constraints.

This specification defines what is necessary for physical safety—not what is sufficient in all possible deployments. It is the responsibility of implementers to ensure that hardware design, sensor reliability, and control software meet the requirements laid out in this document.

Appendix A: Action Metadata Schema

This appendix defines the complete metadata schema required for any action that may be evaluated or executed under the DFW v4.0 architecture. The schema is implementation-agnostic and may be represented in JSON, protocol buffers, ROS messages, or equivalent structured formats.

The purpose of the metadata schema is to ensure the Hybrid Feasibility Layer (HFL) has sufficient information to evaluate physical feasibility, enforce constraints, and verify causal consistency.

A.1 Overview

Each action proposal must include a structured metadata packet $M(a)$ containing:

$$M(a) = \{\tau_{\max}, v_{\max}, \Delta E, \Delta t_{\text{act}}, G_{\text{act}}, x_{\text{pred}}, R, M_{\text{safe}}, \text{reversible}\}$$

Where every element of the set is defined formally below.

Missing, inconsistent, or physically impossible metadata must trigger Safe Mode or result in the nullipotent override SAFE_HALT.

A.2 Mandatory Metadata Fields

1. Maximum Torque ($\tau_{\max}$) The upper bound on torque applied by any actuator:

$$\tau_{\max}(a) \leq \tau_{\text{limit}}$$

2. Maximum Velocity ($v_{\max}$) The maximum velocity the action will produce:

$$v_{\max}(a) \leq v_{\text{limit}}$$

3. Energy Expenditure (ΔE) Predicted energy consumption for the entire action duration:

$$\Delta E(a) \leq E_{\text{limit}}$$

4. Actuator Duration (Δt_{act}) The time required to complete the action:

$$0 < \Delta t_{\text{act}} \leq t_{\text{limit}}$$

5. Actuator Group (G_{act}) A set specifying which motors, joints, thrusters, or subsystems will be activated:

$$G_{\text{act}} = \{g_1, g_2, \ldots, g_n\}$$

6. Predicted Trajectory or Effect (x_{pred}) A representation of the predicted physical outcome. This may take the form of:

- a Cartesian or joint-space trajectory,
- a vector of predicted forces,
- a symbolic state-change descriptor.

Causal consistency (Section 7) requires this value for post-execution verification.

7. Reversibility Flag Indicates whether the action's physical effects are:

- fully reversible,
- partially reversible,
- irreversible.

Irreversible actions must be treated as P1-risk.

8. Safety Margin Estimate (M_{safe}) A confidence-adjusted safety margin:

$$M_{\text{safe}} = \tau_{\text{limit}} - \tau_{\text{max}}(a)$$

If $M_{\text{safe}} < 0$, the action is infeasible.

A.3 Metadata Consistency Requirements

The following internal constraints must hold:

$$\Delta E = \int_0^{\Delta t_{\text{act}}} P(t)\,dt, \qquad v_{\text{max}} = \max_t \|\dot{x}(t)\|.$$

If metadata fields contradict each other, Safe Mode must activate.

A.4 Minimal JSON-Like Example

The following example illustrates the metadata structure in a platform-agnostic form:

```
{
  "name": "MoveToTarget",
  "tau_max": 12.4,
  "v_max": 0.42,
  "delta_E": 3.2,
  "duration": 0.8,
  "actuator_group": ["joint_1", "joint_2"],
  "predicted_trajectory": "trajectory_hash_or_vector",
  "reversible": true,
  "M_safe": 4.3
}
```

A.5 Metadata Failure Handling

Metadata failure cases include:

- missing fields,
- contradictory fields,

- physical impossibility,
- excessive values,
- inconsistent units,
- unverifiable predictions.

In all such cases:

$$\text{HFLOverride}(a) = \text{SAFE\_HALT}.$$

Repeated failures require Safe Mode entry.

A.6 Summary

The Action Metadata Schema provides the information required for:

- enforcing physical bounds,
- verifying feasibility,
- ensuring reversibility,
- validating causal consistency,
- connecting logical veto layers to physical execution.

The metadata in Appendix A forms the foundation upon which the SMCS and HFL operate.

Appendix B: Nullipotent Action Set Specification

This appendix provides the formal specification of each action in the Safe Mode Constraint Set (SMCS). These nullipotent actions constitute the complete set $\mathbb{A}_{\text{Safe}}$ and define the system's behaviour whenever Safe Mode is activated. Each action below is guaranteed to satisfy:

$$\tau_{\max} = 0, \quad v_{\max} = 0, \quad \Delta E \leq E_{\text{safe}}, \quad V_S(a) = 0.$$

All actions are reversible, deterministic, and incapable of producing P1 Irreversible Harm.

B.1 SAFE_HALT

Purpose: Immediately bring all actuators to rest and reduce system momentum to zero.
Guarantees:

- No increase in velocity or torque.
- Bounded deceleration within certified safe limits.
- Final velocity of all actuators is exactly zero.

Required Metadata:

- $\tau_{\max} = 0$
- $v_{\max} = 0$
- $\Delta E \leq E_{\text{safe}}$
- Δt_{act} small enough to ensure rapid stabilization

SAFE_HALT is the default fallback action when uncertainty is detected.

B.2 DIAGNOSTIC_REPORT

Purpose: Emit a cryptographically signed, read-only status report summarising:

- current sensor data,
- actuator states,
- recent actions,
- confidence levels,
- errors and warnings.

Guarantees:

- No actuation or motion.
- Internal state remains unmodified.
- Outputs may only be read, not written.

Metadata Requirements:

- $\tau_{\max} = v_{\max} = 0$
- ΔE consists only of computation

B.3 ISSUE_WARNING

Purpose: Broadcast an audible, visual, or digital hazard alert to notify humans or external systems of unsafe or degraded conditions.

Guarantees:

- No physical interaction beyond signalling.
- No actuator motion.
- No irreversible environmental effects.

Metadata Requirements:

- $v_{\max} = \tau_{\max} = 0$
- Energy usage limited to signalling output

B.4 REVERT_TO_HUMAN_CONTROL

Purpose: Transfer authority to a validated human operator through a secure, one-way protocol.

Guarantees:

- Autonomous control is fully suspended.
- Human operator must be validated via cryptographic credentials.
- No self-reinstatement or override by the AGI is permitted.

Metadata Requirements:

- No actuator motion.
- Secure communication channel required.
- Logs must be preserved and signed.

B.5 Optional Extensions to $\mathbb{A}_{\textbf{Safe}}$

Optional nullipotent actions may be defined, provided they meet all Safe Mode constraints:

- $\tau_{\max} = 0$,
- $v_{\max} = 0$,
- $V_S(a) = 0$,
- No irreversible or state-modifying effects.

Examples include:

- `SAFE_SHELL`: A read-only restricted command shell for human operators.
- `SAFE_TELEMETRY`: Continuous broadcast of sensor data without actuation.
- `SAFE_LIGHTS`: Visual hazard signalling (LED strobes, beacons).

Any extension must be justified with a formal safety proof and included in the certified nullipotent set.

B.6 Summary

The nullipotent action set is intentionally minimal, stable, and deterministic. It guarantees that:

- the system cannot produce P1 harm when uncertain,
- all actions remain physically reversible,
- Safe Mode is predictable and verifiable,
- human operators retain ultimate authority.

Appendix B formalises the operational floor of the DFW's physical safety architecture.

Appendix C: Hybrid Feasibility Layer (HFL) Pseudocode

This appendix provides high-level pseudocode for the Hybrid Feasibility Layer (HFL). The pseudocode is language-agnostic and intended to specify behaviour, not implementation details. Any conforming implementation must preserve the control flow, safety checks, and escalation behaviour described here.

C.1 Core HFL Evaluation Loop

The HFL operates on a stream of proposed actions from the policy model, each accompanied by a DFW verdict and metadata packet $M(a)$.

```
function HFL_process_action(action a, verdict v, metadata M):
    # v in {ACCEPT, REJECT, MDR_FORCE}

    if not metadata_is_valid(M):
        log("Metadata failure - entering SAFE_HALT")
        execute_nullipotent_action(SAFE_HALT)
        increment_metadata_failure_counter()
        if metadata_failure_counter_exceeds_threshold():
            enter_safe_mode("Repeated metadata failures")
        return

    if not metadata_within_bounds(M):
        log("Physical bounds exceeded - entering SAFE_HALT + Safe Mode")
        execute_nullipotent_action(SAFE_HALT)
        enter_safe_mode("Infeasible physical parameters")
        return

    if v == REJECT:
        # DFW veto - override with nullipotent action
        log("DFW veto - executing safe replacement action")
        execute_nullipotent_action(SAFE_HALT)
        return

    if v == MDR_FORCE:
        # Rescue-mandated action
        log("MDR_FORCE - executing rescue action under constraints")
        execute_rescue_action(a, M)
        verify_post_execution(a, M)
        return

    if v == ACCEPT:
        # Normal execution path under strict constraints
        log("ACCEPT - executing base action under HFL constraints")
        execute_bounded_action(a, M)
        verify_post_execution(a, M)
        return
```

C.2 Metadata Validation

```
function metadata_is_valid(M):
    # Check for presence of all mandatory fields
    if missing_required_fields(M):
        return false

    # Check for internal consistency
    if not energy_consistent(M):
        return false

    if not trajectory_well_formed(M):
        return false

    return true

function metadata_within_bounds(M):
    if M.tau_max > TAU_LIMIT:
        return false
    if M.v_max > VEL_LIMIT:
        return false
    if M.delta_E > ENERGY_LIMIT:
        return false
    if M.duration > TIME_LIMIT:
        return false
    return true
```

C.3 Executing Bounded Actions

```
function execute_bounded_action(a, M):
    # Clamp parameters to certified limits as a final safeguard
    tau_cmd   = min(M.tau_max, TAU_LIMIT)
    v_cmd     = min(M.v_max,  VEL_LIMIT)
    duration  = min(M.duration, TIME_LIMIT)

    send_actuator_command(
        action      = a,
        tau_limit   = tau_cmd,
        vel_limit   = v_cmd,
        duration    = duration,
        trajectory  = M.predicted_trajectory
    )
```

C.4 Executing Nullipotent Actions

```
function execute_nullipotent_action(a_safe):
    if a_safe == SAFE_HALT:
        send_actuator_command(
            action     = "halt_all",
            tau_limit  = 0,
            vel_limit  = 0,
            duration   = SMALL_TIME_WINDOW
        )

    else if a_safe == DIAGNOSTIC_REPORT:
        emit_diagnostic_log()

    else if a_safe == ISSUE_WARNING:
        trigger_warning_signals()

    else if a_safe == REVERT_TO_HUMAN_CONTROL:
        transfer_control_to_human()
```

C.5 Executing MDR-Forced Rescue Actions

```
function execute_rescue_action(a, M):
    # Rescue actions must still respect physical limits
    tau_cmd   = min(M.tau_max, TAU_LIMIT)
    v_cmd     = min(M.v_max,  RESCUE_VEL_LIMIT)
    duration  = min(M.duration, RESCUE_TIME_LIMIT)

    send_actuator_command(
        action      = a,
        tau_limit   = tau_cmd,
        vel_limit   = v_cmd,
        duration    = duration,
        trajectory  = M.predicted_trajectory
    )
```

C.6 Post-Execution Verification

```
function verify_post_execution(a, M):
    # Retrieve measured physical data after execution
    x_actual   = read_sensors_trajectory()
    x_pred     = M.predicted_trajectory

    error      = max_norm_difference(x_pred, x_actual)

    if error > PHYSICAL_ERROR_THRESHOLD:
        log("Causal inconsistency detected - entering Safe Mode")
        execute_nullipotent_action(SAFE_HALT)
        enter_safe_mode("Causal inconsistency / execution divergence")
        return

    # If no inconsistency, log successful execution
    log("Action execution consistent with prediction")
```

C.7 Entering Safe Mode

```
function enter_safe_mode(reason):
    log("SAFE MODE ACTIVATED: " + reason)

    # Immediately stop all motion
    execute_nullipotent_action(SAFE_HALT)

    # Raise alarms for human operators
    execute_nullipotent_action(ISSUE_WARNING)
```

```
# Emit detailed diagnostics
execute_nullipotent_action(DIAGNOSTIC_REPORT)

# Suspend autonomous control until SAFE_MODE_EXIT received
suspend_autonomous_control()
```

C.8 Summary

The pseudocode in this appendix specifies the behavioural contract of the HFL:

- No action is executed without validated, bounded metadata.
- DFW vetoes (REJECT) are always respected and enforced physically.
- MDR_FORCE actions are executed within strict physical limits.
- All actions are checked against post-execution sensor data.
- Any inconsistency, metadata failure, or infeasibility triggers Safe Mode.

Any implementation of the DFW v4.0 architecture must preserve these control flows and escalation conditions in order to claim conformance.