Do Actions — Canonical Definition & Enforcement Doctrine

Purpose

This document defines **what a "Do" action is**, what it is not, and how Clonehaus treats execution-capable behavior across OS, Studio, and Runtime. It is the canonical reference for Phase 3 and beyond.

If Do actions are implemented incorrectly, authority collapses into theater. This doctrine exists to prevent that failure.

Core Definition

A Do action is any operation that causes a real-world or system-level change outside the agent's internal reasoning loop.

If an action changes state beyond the agent's own thoughts, it is a Do action.

Canonical phrasing

> A Do action is an execution-capable operation that changes system state, triggers external effects, or commits organizational authority beyond advisory output.

The Non-Negotiable Boundary

Thinking is not doing.

The following are explicitly *not* Do actions: - Reasoning - Explaining - Recommending - Simulating outcomes - Classifying or labeling - Drafting content

These are internal cognitive operations and never require execution authority.

When an Action Becomes a Do Action

An action becomes a Do action if it crosses **any** of the following boundaries.

1. External Systems

- Writing to a database
- Sending emails or messages
- Triggering webhooks
- Calling external APIs
- Creating, modifying, or deleting files

- Executing scripts or jobs

2. Organizational Consequences

- Issuing refunds
- Approving or rejecting transactions
- Changing account or user status
- Assigning tasks to humans
- Automatically escalating incidents

3. Irreversibility

- The action cannot be safely undone by ignoring it
- The action creates audit, legal, or financial consequences

If the answer to *“Can we undo this by just ignoring it?”* is **no**, it is a Do action.

What a Do Action Is Not in Clonehaus

The following never grant execution rights: - “The agent decided to do X” - “The model output looked confident” - “The prompt allowed it” - “It worked in testing”

Execution is never implied, inferred, or prompt-based.

Clonehaus Execution Principle

> **Clonehaus agents are allowed to think freely, but they are only allowed to act deliberately.**

This principle governs: - UX behavior - Authority design - Runtime enforcement - Audit and compliance guarantees

---

Where Do Actions Live in the System

Clonehaus OS

- Defines *whether* a class of actions may ever be executed
- Sets global and domain-level boundaries
- Encodes EAPP constraints that can block execution

Clonehaus Studio

- Defines *who* the agent is
- Shapes identity, behavior, cognition, and escalation posture (LPS)
- Never grants execution rights on its own

Clonehaus Runtime

- Decides *whether a specific Do action may execute*
- Evaluates authority derivation
- Applies EAPP constraints
- Enforces fail-closed behavior
- Records non-bypassable audit trails

No Do action can execute without Runtime approval.

Product Behavior by Phase

Before Phase 3C

- Do actions are visible but disabled
- Clear, plain-language reasons are shown
- No execution affordances exist

Phase 3C and Beyond

- Do actions follow a strict lifecycle:
- Action requested
- Authority derived
- EAPP evaluated
- Approval or refusal issued
- Execution (if approved)
- Audit logged

There is no silent success path.

Why This Matters

Most agentic systems fail because they blur the line between reasoning and execution.

Clonehaus enforces that boundary explicitly, deterministically, and visibly.

This doctrine is foundational and must remain stable as features evolve.