PRODUCT DOCUMENT · ORCHESTRATION RULES

Orchestration rules: a deterministic engine, model work inside it

The engine decides who may act, what runs where and what counts as done. The models plan, build, review and test. The rules are where the two meet, and an organisation tunes them without opening the core.

PUBLIC EDITION

6	4	3	1
agent roles, plus the person who approves	orchestration packs to start from	rungs of reviewer independence	human act behind every merge

APPLIES TO

Solo, Team and Enterprise

every number a setting with a default

PUBLISHED

5 November 2026

Saphan Studio · saphan.ai/product

CONTENTS

01 Summary

02 What the engine decides, what the model does

03 The roles

04 Rules every orchestration keeps

05 Routing by class of work: our recommendations

06 Four orchestration packs

07 What an organisation tunes, and what stays closed

08 Why the founding customer programme tunes this

09 What this document does not claim

10 The Saphan ecosystem and this document set

HOW TO READ THIS DOCUMENT

Section 1 is the summary. Sections 2 to 4 describe the split between engine and models, the roles and the rules every role keeps. Sections 5 and 6 cover routing and the four orchestration packs. Section 7 says what an organisation can tune and what stays closed, and section 8 how the founding customer programme does that tuning.

SECTION 01

Summary

Saphan Studio joins two kinds of work. **The engine is deterministic**: it decides who may act, what runs where, under which limits, and what counts as done. **The models do the work**: they plan, write code and tests, review and measure. The orchestration rules are where the two meet. They have to fit: too loose, and a model can do what nobody approved; too tight, and the agents stop working the moment nobody is watching.

This document describes those rules: the roles, the rules every role keeps, how work is routed, the four operating modes (“orchestration packs”) and what an organisation can tune.

An organisation tunes the settings around the core. It never opens the core: human gates, signatures, the record, confinement and the rule that no author reviews its own work stay closed.

That fit is different in every organisation. This is the main work of the founding customer programme (section 8): tune what an organisation needs, without opening the core so far that someone can break it by accident.

SECTION 02

What the engine decides, what the model does

THE ENGINE (DETERMINISTIC)	THE MODELS (AGENTS)	PEOPLE
Accepts only signed orders, admitted machines and verified seats	Plan the work inside an order	Sign orders
Places each run on a seat allowed for its class of work	Write code and tests in their own working copy	Approve gates; decide
Confines each run; egress closed by default	Review another agent’s change	Merge
Refuses by name, with the reason and the next action	Test a whole train from outside	Choose settings and packs
Enforces budget caps and records cost per run	Measure and prepare repair orders	Answer questions, three at a time
Writes every act into an append-only, signed record	Draft documents and decisions	Own the record and its backups

A model never holds a gate. When an agent reads a rule as permission to merge, approve or tear down, the engine refuses it at the permission grant, and that refusal is correct.

SECTION 03

The roles

ROLE	DOES	NEVER
Planner (master)	Writes orders and test plans, coordinates a track, verifies returns, prepares each human act	implements, reviews, or merges; runs work against scope a person has not accepted

ROLE	DOES	NEVER
Implementer (executor)	Implements one order in its own working copy and returns proof	creates or removes working copies; pushes; merges
Reviewer	Judges a change in five acts and recommends	writes code; reviews its own work
Tester	Checks a whole train from outside, once, at its end	fixes the product; counts as a review
Record keeper (scribe)	Keeps the record, the ledger and the canon current	dispatches, gates or merges
Advisor	Measures, advises, prepares decisions with a recommendation	decides; writes another role's files
The person (approver)	Signs, approves, merges; owns irreversible acts	—

One seat can serve several roles in turn, one session per role. A Solo installation with one seat still keeps the roles apart.

SECTION 04

Rules every orchestration keeps

- **One order, one planner.** Every order names its planner in its first line. A message in chat re-assigns nothing.
- **A recommendation and a decision are two recorded events.** Silence never ratifies a recommendation.
- **A known next act is performed, not announced.** An agent that knows the next step takes it.
- **A full fleet is a wait, never a stop.** Work queues for a free seat; it is not abandoned.
- **A question to a person carries its recommendation**: the problem measured, the options, the recommendation and the cost if it is wrong, numbered so the person answers with a letter. At most three at a time.
- **Nothing reaches a person unrehearsed.** A command handed to a person has been rehearsed along the whole path, merge preconditions included. Acts come in one batch, not as a scroll of lines.
- **Verify before you assert.** A status such as merged or done is stated only from live evidence; otherwise it is "unverified".
- **Fix or shout.** Another track's defect is fixed through the right track or reported in red. A silent workaround is not allowed.
- **A host is drained, never waited out.** A failing machine is cordoned and its work re-placed.
- **Every number is a setting with a default.** No hard-coded hosts, paths or people.

SECTION 05

Routing by class of work: our recommendations

Work is routed by its class, not by habit. The engine places each run on the cheapest seat that is *allowed* for its class.

Routing is fully customisable by rules. The organisation decides which classes of work exist, where each one runs and who reviews it. The table below is our recommendation, and the default a new installation starts from.

CLASS OF WORK	RECOMMENDED: WHERE IT RUNS	RECOMMENDED: REVIEW
Small and medium changes over an existing seam (one package and its tests)	any admitted seat, including low-cost or free model lanes	another vendor
New stores, tables or projections	a full-strength model	named review, full-strength
Network work	a full-strength vendor that can bind a listener	full-strength
Security: grants, egress, signatures, authentication, confinement	a full-strength vendor	two reviews, one full-strength
Anything a person reads or clicks	a full-strength vendor	full-strength, plus the tester
Read-only measurement (reproduce, measure, write one document)	any free seat	the planner
Deterministic work: tests, builds, browser walks	build machines, never the machine that serves the record	—

- **Cost is quoted before it is spent.** Every run carries an estimate; its cap is twice the estimate. A run that exceeds its estimate but stays under its cap is still reported as a pricing defect.
- **Capacity is measured, never guessed.** Before dispatch the engine checks the lane's current headroom; absence of evidence is never a pass.
- **Every vendor is treated the same.** A mechanism is built for every vendor's agent; where that is impossible the vendor and the reason are named.

SECTION 06

Four orchestration packs

An orchestration pack describes one way of running the agents: when it applies, who does what, where a person is needed, how often things are watched, how work and reviews are routed, which settings to start from, and how the mode ends. A new installation starts from these packs instead of from nothing, and changes them as it learns.

PACK	WHEN	WHAT CHANGES
Attended day	the approver is at the desk and answers within minutes	acts and decisions flow; one queue of rehearsed acts; signatures in batches; hourly one-line reports
Unattended night	the approver is away or asleep	the agents do everything that does not need a person; low-cost lanes carry small work and first reviews; one morning file with what landed, what waits for a hand and what broke
Release crunch	the countdown to a release decision	only release blockers move
Incident pressure	the record store, a machine or a model lane is overloaded	shed load; short quiet windows; protect the person’s acts

Rules every pack keeps

- **Generic**: no host names, no people, no project names. An installation’s specifics live in its own workspace.
- **Every number is a setting with a default**, and a check verifies that every setting a pack names exists.
- **Any vendor**: a pack never assumes one AI vendor.
- **A person’s confirmations stay theirs**: codes typed for gates and merges are never carried, relayed or typed by an agent.

The packs follow the engine: when a release changes how work is scheduled, routed or reviewed, the affected pack is updated in the same release.

SECTION 07

What an organisation tunes, and what stays closed

Tunable: settings with defaults

- **Routing, fully by rules:** the classes of work, where each one runs and who reviews it
- Reviewer independence (`review.cross_vendor`) and review seats (`dispatch.review_seats`)
- The largest change a low-cost lane may take (S, M or L)
- Concurrency per vendor account and per machine
- Budgets per train and caps per run
- Watch cadences: machine probes, fleet glance, silent-planner check
- Continuous scheduling per train, and its window
- Which skills are switched on per project
- Which pack applies, and the day and night hours that switch between them
- Which notices wake a person’s phone, by severity

Closed: the core

- Gates are human acts. No agent approves or merges.
- Orders, gates and evidence are signed and checked when opened.
- The record is append-only. A dashboard is a view of it, never a second truth.
- Runs are confined; egress is closed unless a person opens it with a recorded reason.
- Refusals are named, with the reason and the next action.
- The author’s own session never reviews its own work.
- An agent never relays a person’s confirmation code.

By edition

	SOLO	TEAM	ENTERPRISE
Built-in rules and packs	used as shipped	used as shipped	used as shipped
Settings with defaults	yes	yes	yes
Add the organisation’s own rules	—	yes	yes
Change built-in rules (within the closed core)	—	—	yes
Calibration packs	—	load, update, switch layers off	everything Team has; off until the organisation enables it; anonymous sharing off by default; export of the organisation’s own pack

A **calibration pack** is starting knowledge for a new installation: cost figures per vendor, about a hundred generic lessons, base skills and order templates. It is knowledge, never rules: the core is never in a pack. Each edition column includes everything in the column to its left.

> **Why a separate place for house rules matters.** A customer has house rules too: which mode is the default, how a command is handed to a person, which products may be named in public. A product that gives them no place gets them written into the product’s own rules, where the next upgrade overwrites them. The local rules file is read after the product rules and may never contradict them; where it does, the product rules win.

SECTION 08

Why the founding customer programme tunes this

The engine’s determinism and the models’ work have to fit each other. The right fit depends on the organisation: how many approvers it has, which vendors it admits, how much it lets low-cost lanes do, how strict its separation of duties must be, how its day is split between attended and unattended hours.

01 **Start from a pack** and the product defaults.

02 **Measure**: idle seats, rounds per change, first-review acceptance, time from signature to merge, cost per train.

03 **Change one setting at a time**, recorded as a decision with its reason.

04 **Keep what improves the numbers**; roll back what does not. The record shows both.

05 **Write the organisation’s house rules** into its own layer, never into the product core.

The aim is an orchestration the organisation can run without us, with the core closed and the settings its own.

SECTION 09

What this document does not claim

- The packs describe how Saphan Studio’s own agents are run. They are a starting point, not a guarantee of results in another organisation.

SECTION 10

The Saphan ecosystem and this document set

Saphan Studio is one product in a small ecosystem. The tools around it are open source under Apache 2.0 and each works on its own; together they cover a change from the first read of the code to the evidence that it is right.

`Discover reads the code` → `Skills Studio checks the skills` →
`Saphan Studio runs the agents under a person's approval` → `Mocks Studio stands in for real systems` →
`Tests Studio and Proven judge the result`

TOOL	ITS PART
Saphan Studio saphan.ai	The control plane: signed orders, confined agents, independent review, human approval, one signed record.
Discover github.com/saphan-ai/discover	Reads a codebase before anyone changes it, with a source for every claim; proposes the skills that fit it.
Skills Studio github.com/saphan-ai/skills-studio	Checks and signs the skills agents are given, so anyone can verify them again.
Mocks Studio github.com/saphan-ai/mocks-studio	Stand-in systems agents and tests work against instead of the real ones.
Tests Studio github.com/saphan-ai/tests-studio	Shows what a test run really proved, in the framework the team already uses.
Proven github.com/saphan-ai/proven	An AI code reviewer that reports a problem only with evidence.

The document set

DOCUMENT	WHAT IT ANSWERS
Saphan Studio for developers and DevOps	start here: what Saphan Studio is and how the pieces fit
Skills and craft	what every agent carries into the work
Testing and review	how a change earns its merge
Orchestration rules (this document)	a deterministic engine, model work inside it
Security	what an agent can reach, and what you can prove afterwards

For partners: a founding customer onboarding guide and an Enterprise continuity plan, at saphan.ai/for-partners.

All documents: saphan.ai/product · Code quality against published baselines: saphan.ai/research/proven-effort · Operator documentation: docs.saphan.ai · Contact: hello@saphan.ai