PRODUCT DOCUMENT · SKILLS AND CRAFT

Skills and craft: what an agent carries into the work

Every agent under Saphan Studio works by the craft of its role and the skills its project switches on. A skill adds instruments and numbers for one stack; it never relaxes a rule. This document lists both.

PUBLIC EDITION

55	7	71	0
skills in the library	roles with a written craft	numbered craft rules, four core roles	skills that relax a rule

FORMAT

Agent Skills (SKILL.md)

open format · verified by Skills Studio

PUBLISHED

5 November 2026

Saphan Studio · saphan.ai/product

CONTENTS

01 Summary

02 Craft and skills: the difference

03 The craft, role by role

04 The skill library: 55 skills

05 What a skill leaves to people

06 How a skill reaches an agent

07 One ecosystem: from the code to the right skills

08 What this document does not claim

09 The Saphan ecosystem and this document set

HOW TO READ THIS DOCUMENT

Section 1 is the summary. Section 2 explains the difference between craft and skills, section 3 shows the craft role by role, and section 4 lists all 55 skills. Sections 5 to 7 cover what stays with people, how a skill reaches an agent, and the open-source tools around it (Discover proposes skills from the code).

SECTION 01

Summary

Every agent that works under Saphan Studio carries two things into its work.

- **The craft of its role.** How a planner, an implementer, a reviewer or a tester works: short, numbered rules such as “a defect is reproduced by a failing test before any fix” or “judge the change against the contract, never the author’s claim”. The craft is the same in every project.
- **The skills the project switches on.** A skill adds the concrete instruments and numbers for one stack or one practice: the Go toolchain, PostgreSQL query plans, a WCAG target, a browser driven through Playwright. A project switches on only the skills it needs.

A skill can add to the craft or raise its bar. It can never relax it.

The library holds 55 skills, from language toolchains to architecture: how a screen reads its data, how a process holds its connections, how a cache is invalidated. The Saphan Studio project itself switches on 28 of them. Section 4 lists every skill with one line on what it does; section 3 shows the craft, role by role.

	TODAY	WHERE TO CHECK
Skills in the library	55	section 4
Roles with a written craft	7 (common, planner, implementer, reviewer, tester, record keeper, advisor)	section 3
Numbered craft rules for the implementer, reviewer, tester and planner	71	section 3
Skills switched on in the Saphan Studio project	28	section 6
Tools in the same open-source chain	Discover, Skills Studio, Mocks Studio, Tests Studio, Proven	section 7

SECTION 02

Craft and skills: the difference

Saphan Studio runs agents in **roles**. One agent plans and coordinates a piece of work, others implement it, an independent agent reviews it, and at the end of a larger piece of work a tester checks the whole from the outside. A human approves every merge (see the companion document *Testing and review*).

	CRAFT	SKILL
What it is	How a role does its work	Instruments and numbers for one stack or one practice
Who sets it	Saphan Studio, the same for every project	The project: a person switches each skill on
How it reaches the agent	Delivered with every order for that role	Named in the order; delivered with it into the agent’s own working copy

	CRAFT	SKILL
Can it relax a rule?	—	No. A skill only adds an instrument, a number or a stricter bar
Format	Numbered rules in plain English	The open Agent Skills format (a folder with a SKILL.md file)

An example

The implementer's craft says: *a defect is reproduced by a test that fails on the base for the reason the order states, before any fix; after the fix, revert it once and watch the test fail*. That rule is the same in Go, Python or Swift. The skill `lang-go` adds *how* to do it in a Go module: which commands build and test it, how to run one test, how to read the race detector. The skill `testing-bar` adds a number the project may only improve, never lower.

Each skill states what it equips

Each skill lists the roles it serves and the craft rules it equips. It also declares the skills it conflicts with and the upstream practice it was written from. Parameters are explicit: a WCAG target, a performance budget or a quality floor is a value the project sets, never a value the agent guesses.

SECTION 03

The craft, role by role

The craft is short on purpose. Each rule is one line with a reason behind it, and a reviewer can check it on a diff. The counts are the numbered rules on 5 November 2026; the role names in brackets are the names used inside the product.

ROLE	RULES	EXAMPLES
Common to every role	–	Read every skill the order names before the first change. One program per call. No AI attribution in commits. Re-enter from the record after every restart: inbox, watchers, open debts.
Planner (master)	16	M1 every order says what it does *not* ask for · M2 success is a number with its command · M4 one slice per order · M6 a review order carries the change and the contract, never the author's claim · M8 a quality bar ratchets · M10 business constraints are written, never inferred · M12 measure before asking a human to sign a repair
Implementer (executor)	24	E1 red-first is a proof · E2 stop the line on an unexpected failure · E4 a refactor and a behaviour change are separate commits · E5 interfaces and stored data grow, they do not break · E6 tests run and tests skipped are numbers · E10 baseline first · E11 in a fix round, findings are claims to verify, not orders · E16 every read is bounded at the source · E22 every error is handled or named
Reviewer	17	R1 judge the change against the contract · R2 conformance first · R3 mutate the conditions: invert each new guard once · R4 read the dependency delta · R5 re-run a performance claim · R7 every blocking finding carries its location and a named remedy · R12 the author's reasoning never lowers a finding's class · R14 a re-review judges only the findings and the fix
Tester	14	T1 test against the contract, not the implementation · T2 use your own disposable instance · T3 outside in · T6 leave and come back, every way the platform allows · T7 an edge matrix for every acceptance item · T9 chaos and repeat · T12 a defect is returned as a reproduction, never a fix · T13 what you cannot run is declared UNRUN

ROLE	RULES	EXAMPLES
Record keeper (scribe)	–	Rebuild the state from the record, never from memory. Keep the record and its index current. Never dispatch, gate or merge.
Advisor	–	Rebuild from the record at every start. Keep one file of standing rulings, current the same day. Measure before routing. Bring a human at most three decisions at a time, each with a recommendation.

> A role also has a **boundary**: what it may never do. An implementer never merges and never creates or removes working copies. A reviewer writes no code. A tester never fixes the product. Only a human merges. These boundaries are enforced by the engine’s permission grants, not only written down.

SECTION 04

The skill library: 55 skills

Each line below is the skill's own name, the roles it serves and what it equips them to do. Grouping is for this document only.

Languages and toolchains (8)

SKILL	ROLES	WHAT IT EQUIPS
`lang-go`	implementer, tester, reviewer	Build, test and judge changes in a Go module.
`lang-typescript-node`	implementer, tester, reviewer	TypeScript and JavaScript on Node or Bun.
`lang-python`	implementer, tester, reviewer	Python projects: pyproject, requirements, uv, Poetry or Pipenv.
`lang-rust`	implementer, tester, reviewer	Rust and Cargo workspaces: tests, benches, examples.
`lang-dotnet`	implementer, tester, reviewer	C# and F# solutions and their test projects.
`lang-jvm-spring`	implementer, tester, reviewer	Java or Kotlin on the JVM, Gradle or Maven, with or without Spring.
`lang-kotlin-android`	implementer, tester, reviewer	Android apps and libraries in Kotlin, including lifecycle tests.
`lang-swift-ios`	implementer, tester, reviewer	iOS apps, extensions and Swift packages.

Dependencies and supply chain (3)

SKILL	ROLES	WHAT IT EQUIPS
`supply-chain-go`	implementer, reviewer	Judge and make dependency changes in a Go module: go.sum, vendoring, proxies, generators.
`supply-chain-npm`	implementer, reviewer	Dependency changes from the npm registry: lockfiles, .npmrc, install scripts.
`supply-chain-python`	implementer, reviewer	Dependency changes from PyPI or a mirror: lock files, hashes.

Data and storage (5)

SKILL	ROLES	WHAT IT EQUIPS
`db-postgres`	implementer, tester, reviewer	Query plans, DDL locking, connection limits, tests against a real server.

SKILL	ROLES	WHAT IT EQUIPS
schema-migrations	planner, implementer, reviewer	Change a schema without breaking running code or losing data.
idempotent-writes	implementer, tester, reviewer	One intent, exactly one effect, even when a request or message is delivered twice.
read-sql	implementer, tester, reviewer	The one way to read a SQL store: filters in the query, keyset pages, a deadline.
read-couchdb	implementer, tester, reviewer	The one way to read CouchDB: indexed queries, paging, the changes feed.

Bounded reads (3)

SKILL	ROLES	WHAT IT EQUIPS
read-files	implementer, tester, reviewer	Read a file tree with a named glob, depth and count bounds, streaming for large files.
read-git	implementer, tester, reviewer	Read a git repository with ranges, pathspecs and counts, never a full history walk.
read-rest-api	implementer, tester, reviewer	Read an HTTP API with server-side filters, cursor pages, deadlines and conditional repeats.

Architecture: reads, data access and performance (8)

SKILL	ROLES	WHAT IT EQUIPS
read-path-serves-projections	implementer, tester, reviewer, advisor	Every screen and read API serves a stored projection written in the background, so a read computes nothing at request time and stays inside a declared response budget under load.
background-work-and-projections	implementer, tester, reviewer, advisor	The background writers that keep those projections current: idempotent, checkpointed, resumable, published atomically, with a staleness alarm.
data-access-layer	implementer, tester, reviewer, advisor	Every data source behind one owned, typed boundary, so callers use a contract and switching engines touches only the layer.
cache-layer	implementer, tester, reviewer, advisor	One ready-made cache library per process instead of hand-written caches: invalidation in the write's own transaction, request coalescing, a size bound, hit and miss metrics.
connection-lifecycle	implementer, tester, reviewer, advisor	One pool or client per process per data source, opened at start and closed at shutdown, with explicit timeouts and short read transactions.
request-lifecycle	implementer, tester, reviewer, advisor	The work behind a request lives exactly as long as its caller: cancellation propagates, every request has a deadline, no subprocess on the request path.

SKILL	ROLES	WHAT IT EQUIPS
client-data-fetching	implementer, tester, reviewer, advisor	A screen fetches per section in parallel, refetches once after a change signal, uses a push channel instead of fixed polling.
architecture-census	planner, reviewer, advisor	A review profile that turns one symptom seen on several surfaces into a census of request paths, names the missing standard layer and proposes ready-made libraries.

Interfaces and contracts (5)

SKILL	ROLES	WHAT IT EQUIPS
api-http-conventions	implementer, tester, reviewer	Conventions an HTTP API is built and reviewed against; the repository's own style wins.
api-mcp-server	implementer, tester, reviewer	Test and review a Model Context Protocol server from outside, as a client.
contract-first-boundaries	implementer, tester, reviewer	One machine-readable contract per boundary; clients and fakes derived from it; checked in CI.
semver-and-changelog	implementer, reviewer	Classify changes under semantic versioning; the version decision stays with a human.
deprecation-program	planner, implementer, reviewer	Retire an interface others depend on, in announced steps.

Front end and product surfaces (9)

SKILL	ROLES	WHAT IT EQUIPS
frontend-engineering	implementer, tester, reviewer	A framework-neutral practice for building and judging web front ends, screen by screen.
framework-react-web	implementer, tester, reviewer	React component and hook tests in the browser stack.
ui-web-playwright	tester, reviewer	Drive a web product in a real, contained browser through Playwright.
ui-terminal-pty	tester, reviewer	Drive a terminal program or full-screen TUI through a pseudo-terminal.
ui-mobile-native	tester, reviewer	Drive a native app on an emulator or simulator, lifecycle included.
accessibility-wcag	planner, tester, reviewer	Measure a screen against the WCAG target the project declares.
perf-web-vitals	implementer, tester, reviewer	Measure page performance in the lab against a declared budget.
ui-walk-before-handoff	tester, reviewer	Walk every screen before it reaches a person.

SKILL	ROLES	WHAT IT EQUIPS
ux-every-screen-answers-the-person	tester, reviewer	Each screen says what happened, who did it, why it matters and what to do next.

Mobile apps (3)

SKILL	ROLES	WHAT IT EQUIPS
framework-jetpack-compose	implementer, tester, reviewer	Android screens in Jetpack Compose: state hoisting, lifecycle-aware state, recomposition cost, navigation, accessibility, Compose tests and benchmarks.
framework-swiftui	implementer, tester, reviewer	iOS and macOS screens in SwiftUI: state ownership, async work bound to view lifetime, strict concurrency, navigation, accessibility, previews.
mobile-presentation-architecture	implementer, tester, reviewer, advisor	One declared presentation pattern (MVVM, MVI, Clean Architecture and others), applied the same way on every screen.

Infrastructure and delivery (4)

SKILL	ROLES	WHAT IT EQUIPS
iac-terraform	implementer, reviewer	Infrastructure as code with Terraform: plans read before anything is applied.
containers-dockerfile	implementer, reviewer	Container images: small, pinned, reproducible.
k8s-manifests	implementer, reviewer	Kubernetes manifests: limits, probes, least privilege.
ci-github-actions	implementer, reviewer	CI workflows on GitHub Actions: pinned actions, least-privilege tokens.

Security and privacy (2)

SKILL	ROLES	WHAT IT EQUIPS
web-security-hardening	implementer, tester, reviewer	Concrete controls and checks for anything a browser talks to: forms, sessions, cookies, headers.
privacy-and-pii	planner, implementer, reviewer	Handle personal data: minimise, protect, keep it out of logs.

Quality and operations (5)

SKILL	ROLES	WHAT IT EQUIPS
testing-bar	planner, implementer, reviewer	A numeric quality floor (coverage, mutation score, warnings) that may improve but never worsen.

SKILL	ROLES	WHAT IT EQUIPS
`review-hygiene`	planner, reviewer	A whole-range or whole-repository review: function length, complexity, duplication, dead code.
`observability-stack`	implementer, tester, reviewer	New I/O is observable without exposing secrets.
`feature-flags-and-rollout`	planner, implementer, tester, reviewer	Evidence with the flag on and off; staged rollout stays with humans.
`adr-in-repo`	planner, implementer, reviewer	Record decisions that are expensive to reverse in the repository; acceptance stays with a human.

SECTION 05

What a skill leaves to people

Several skills draw a line on purpose. The agent prepares the evidence; a person makes the call.

SKILL	THE AGENT DOES	A PERSON DECIDES
`feature-flags-and-rollout`	tests with the flag on and off; sequences migrations	staged rollout, canary thresholds, flipping a flag in production
`semver-and-changelog`	classifies each change; keeps the changelog	the version number and the release
`adr-in-repo`	writes the decision record with options and consequences	accepting the decision
`deprecation-program`	plans the steps; finds the callers	the dates announced to other teams
`accessibility-wcag`	measures each screen against the target	the conformance target itself

SECTION 06

How a skill reaches an agent

01 **A person switches the skill on for the project.** It is a recorded act, not a file someone copies in.

02 **The planner names the skills in each order**, in its own section, with the exact copy that will be delivered.

03 **The engine delivers the order, the role's craft and the named skills** into the agent's own working copy, together.

04 **The agent reads every named skill before its first change.** It reads no skill the order does not name.

05 **The reviewer receives the same craft and skills**, so the change is judged by the bar it was built to.

Skills Studio, our open-source tool (Apache 2.0, github.com/saphan-ai), checks a skill's fingerprint and signature, so a project can prove which version of a skill an agent was given.

Switched on in the Saphan Studio project

lang-go · lang-typescript-node · framework-react-web · frontend-engineering · ui-web-playwright · ui-terminal-pty · testing-bar · supply-chain-go · supply-chain-npm · web-security-hardening · contract-first-boundaries · read-couchdb · read-sql · read-git · read-files · read-rest-api · review-hygiene · read-path-serves-projections · background-work-and-projections · data-access-layer · cache-layer · connection-lifecycle · request-lifecycle · client-data-fetching · architecture-census · framework-jetpack-compose · framework-swiftui · mobile-presentation-architecture

---

SECTION 07

One ecosystem: from the code to the right skills

Skills do not stand alone. They sit in a chain of Saphan tools, each open source under Apache 2.0 at github.com/saphan-ai, and each usable without Saphan Studio. Together they answer one question: *which skills should these agents carry for this code, and can we trust them?*

`Discover reads the code` → `skills proposed from what it found` → `Skills Studio checks and signs` → `a person switches them on` → `Studio delivers them with each order` → `Proven and Tests Studio judge the result`

TOOL	ITS PART
Discover	Reads a repository without changing it and without running its code, offline, and proves at the end that it changed nothing. It describes the code with a source for every claim: languages, versions, how it is built, tested and run, its convention files. Saphan Studio reads that profile at setup and **proposes the skills whose "use when" matches what Discover found**: a go.mod proposes `lang-go` and `supply-chain-go`, a package-lock.json `supply-chain-npm`, a migrations folder `schema-migrations`.
Skills Studio	Checks a skill before it reaches the agents, records what was checked, and hands it over with a fingerprint and signature anyone can verify again.
Mocks Studio	Stand-in systems (mail, file store, user directory, object store, Kubernetes and more) that agents, tests and skills work against instead of the real one.
Tests Studio	Shows what a test run really proved, in the framework the team already uses.
Proven	An AI code reviewer that reports a problem only with evidence: a failing test or a query and its rows.
Saphan Studio	Switches skills on per project as a recorded act, delivers them with every order and keeps which version each agent was given.

A proposal is only a proposal: the person who owns the project switches each skill on. Nothing reaches an agent because a tool guessed it.

SECTION 08

What this document does not claim

- A skill does not make a model correct. It equips the agent and the reviewer with the same instruments and numbers; the independent review and the human approval remain the check.
- The skills were written for the stacks we and our first customers use. A stack without a skill still works under the common craft; it lacks the instrument for that stack.

SECTION 09

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 (this document)	what every agent carries into the work
Testing and review	how a change earns its merge
Orchestration rules	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