Skynet — Workflow Source Map (as-is → to-be, by workflow and platform source)

For: Agon, Sead, Sven (CTO), Vince — and anyone who needs to see how work flows through Skynet and where each piece comes from.

Companion to: the approved functional design (what the app is), the architecture and deployment design (how it is built and run), the delivery plan (in which order).

---

1. Purpose and how to read this

The functional design says what Skynet does. This document says **how each workflow runs today, how it runs in Skynet, and where every step comes from** — ported from one of our four apps, kept in a standalone app for now, read from ecomprofits, built new, or a manual step that disappears.

One section per workflow. Each section has the same shape:

- **Trigger · actors · inputs**
- **As-is today** — numbered steps: who, tool, what happens, whether it is manual, and where it breaks
- **To-be in Skynet** — numbered steps: who, module, what happens, automated or human-gated
- **Source map** — one row per to-be step: source platform, code path today, verdict, slice
- **What people stop doing**
- **Evidence status** — `code-backed` (read from the repos), `confirmed` (Agon confirmed the practice), `recommended` (no established practice today or inferred from code; the to-be is a recommendation built on the approved functional design, to be confirmed by the named D-card), `pending` (must be confirmed before it is treated as fact)

Slices: `S1` = the first slice (the two-week build), `S2` = Intelligence merge, `S3` = Studio merge + strategy layer, `S4` = Workspace cutover, `S5` = SaaS hardening. Slices are a sequence, not calendar dates.

2. Source-verdict legend

VERDICT	MEANING
`PORT`	Rebuilt as native Skynet behaviour from static-ads-automation (the product base) — code moves into `apps/api` / `apps/web` in S1
`PORT (phased)`	Functionality that lives in a standalone app today (meta-ads-scraper, RescaleOS) and is merged into Skynet natively in a later slice; the standalone keeps running on `server.rescale.media` until its replacement is accepted, and may receive temporary bridge-track-B changes while S1 runs; then it is retired
`REUSE`	Logic stays in the service that owns it (rescale-workos on Railway) and Skynet calls it over an API
`READ`	Skynet reads the data; it never owns or writes the logic (ecomprofits)
`NEW`	No platform has it today; Skynet introduces it
`REPLACES MANUAL`	A human step disappears or shrinks to a review/approval gate

Plain rule for readers: only static-ads-automation is `PORT` in the first slice. The scraper and RescaleOS are `PORT (phased)`; in S1 Skynet links to them, and bridge track B is the only allowed standalone change set. Workos is `REUSE`. Ecomprofits is `READ`, always.

3. Workflow inventory

#	WORKFLOW	OWNER OF THE AS-IS KNOWLEDGE	EVIDENCE STATUS
W1	Competitor research and tracking	no routine today (Agon confirmed); to-be agreed	confirmed
W2	Product research → tracked product	no routine today (Agon confirmed); to-be recommended	recommended
W3	Deep teardown + market intel / VOC	no routine today; to-be recommended	recommended
W4	Strategy: coverage map, gaps, next batch — **NEW workflow**	recommended; D32 confirms with Mathew + Cyrus	recommended
W5	Script creation (NPT / BPT / briefs)	engine code-backed; Bian's practice inferred (D31)	recommended
W6	Hook session + approval gates — **NEW workflow**	recommended; D32 confirms	recommended
W7	Static production (Explore / Expand / Long-form)	Vince	code-backed
W8	Video production (storyboard / clips / narration / assembly)	Vince	code-backed
W9	Canva polish + localisation	Vince	code-backed
W10	Naming + creative code	Sead + Sven	code-backed
W11	Launch preparation + media-buyer hand-off — ends when the launch package / launch entity exists	cascade code-backed; buyer practice inferred (S0b/D41)	recommended
W12	Performance readback + winner / loser tagging	Sead	code-backed
W13	Learnings, memory, lineage	no defined practice today; to-be recommended	recommended

#	WORKFLOW	OWNER OF THE AS-IS KNOWLEDGE	EVIDENCE STATUS
W14	Workspace: board lifecycle, approvals, assignment, reconciliation (ClickUp) — owns everything after the launch entity exists	cascade code-backed; exceptions inferred (D42)	recommended
W15	Admin: models, spend, settings, users	Agon	code-backed
W16	Team chat: notifications, chat ingestion, agents in chat (Telegram + Discord)	ecomprofits pattern (Agon built it); to-be recommended	recommended

W10 · Naming + creative code

Trigger. A creative is ready to be launched, or a Creative card is created in ClickUp.

Actors. Strategist / media buyer (today), the cascade service (workos), Skynet Production (to-be).

Inputs. Product number, market, channel, creative kind (video/image), team-made vs rip, the four people (funnel builder, strategist, video editor, media buyer), CU id.

As-is today (code-backed: rescale-workos `service/src/modules/naming/*`, `cascade/apply-names.ts`; ecomprofits `naming_conventions`)

1. A Creative card exists in ClickUp (spawned by the demand cascade, or added by hand). — automated / manual
2. The workos cascade resolves the code parts: kind from Format (`RC` video / `TM` image), `i` if Approach ≠ Imitation/Expand, `seq` from `rescale_service.next_naming_seq(product, kind)` in ecomprofits Postgres, variant letter `A/B/C…`, optional `hook N`. It writes the code to the Creative card. — automated (`ensureCreativeCode`)
3. When all fifteen inputs exist, `applyAdName` assembles the v3 ad name (`#910.0DE - EcoDrive™ | TMi2 | DR - DE - FB | … | v3`) and writes it to the Launch card's `Ad name NC`; if a human input is missing it withholds with a reason. — automated, fails closed
4. **Manual:** in the static-ads tool, the human types the creative code by hand into the winner upload form (`product_creatives.creative_code`, free text — e.g. `TMi60-A`, `TM 1 - 2`). Two systems, no link between them. — manual, error-prone
5. **Manual:** the media buyer copies the ad name from ClickUp and pastes it into Meta Ads Manager when uploading. — manual

6. ecomprofits parses the ad name with the account's naming rules (`Rescale Meta Ad (v2)` , delimiter `|` , position 1 = `creative_code`) into `meta_ads.parsed_naming` , exposed as `nc_creative_code` on `meta_ads_dashboard_view` . — automated

To-be in Skynet

1. Production issues the code at export: `POST /api/creative-codes` → kind from asset type, `team_made` from session mode (Explore = made, Expand = rip), `seq` from **the same** `rescale_service.next_naming_seq` (one counter, never a second generator). — automated
2. The export stamps the code into filenames (`<code>__<concept>.<ext>`) and `manifest.json` , and asks workos for the assembled ad name (`POST /naming/ad-name` , the same assembler). — automated
3. The winner-upload form is read-only for the code (prefilled). — `REPLACES MANUAL` step 4
4. The existing ClickUp Launch card carries the package — code, assets, ad name; the buyer copies one string. — `REPLACES MANUAL` step 5 (copy stays, retyping and dropdown errors go)
5. ecomprofits parses as today; Skynet reads `nc_creative_code` back (W12). — unchanged

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VER
1 issue code	rescale-workos	`naming/creative-code.ts` (`buildCreativeCode`), `naming/sequence.ts` → `rescale_service.next_naming_seq`	`REU` one gran one coun one (inva how calls the arch doc)
2 stamp + ad name	static-ads export + workos assembler	`routes/products.py:213` → `export_service.export_product`; `naming/assemble.ts` (`assembleAdName`)	`POR` (exp `REU` (ass via r `POS` `/nam` `name`
3 read-only code on winners	static-ads	`routes/products.py:450` upload form	`POR` `REP` `MANU`
4 launch card carries the package	workos Launches list	`domain/ids.ts` `LISTS.launches`, `apply-names.ts`; S8a	`REU` (bric W11
5 parse back	ecomprofits	`naming_conventions` rules, `meta_ads.parsed_naming`, `meta_ads_dashboard_view.nc_creative_code`	`REA`

What people stop doing. Typing creative codes by hand in the static-ads tool; reconciling two different code sources; renaming exported files.

Evidence. code-backed. Contract: day-0 lock §4.2. Cards: S3, V7, S8.

W12 · Performance readback + winner / loser tagging

Trigger. ecomprofits' Meta sync has new insight rows for an ad whose name carries a creative code.

Actors. Nobody today (numbers are looked up by hand); Skynet readback job (to-be); strategist / media buyer reading results.

As-is today (code-backed: ecomprofits `27-meta-ads.sql`, `40-dashboard-views.sql`, `meta-ads-sync/business-logic.ts`)

1. ecomprofits syncs Meta insights per ad per day into `meta_ad_insights` (final) and `meta_intraday_ad_insights`; native currency plus `*_eur` columns. — automated
2. At sync time it computes `winner_loser_status` (eligible when 7-day spend > €80; winner if 7-day ROAS ≥ breakeven × 1.15; loser if below breakeven) and `creative_hit`. — automated
3. Hook rate (`video_plays / impressions`) and hold rate (`video_thruplay_watched / video_plays`) are computed only in the ecomprofits TypeScript dashboards, not stored. — automated, dashboard-only
4. **Manual:** a strategist opens Looker / the ecomprofits dashboard, filters by product or ad name, reads spend / ROAS / hook rate, and decides what won. Nothing links the number to the script, session or evidence that produced the ad. — manual
5. **Manual:** "winner" is written into ClickUp by hand (or by the workos platform-sync from ecomprofits, on the Launch card) and, separately, uploaded as a winner into the static-ads tool with a typed code. — manual, duplicated

To-be in Skynet

1. A scheduled readback job reads `skynet_creative_daily` (view over `combined_meta_ad_insights` ⋈ `meta_ads.parsed_naming`, always filtered by the workspace's ecomprofits `account_id`) for every creative code Skynet has issued. — automated
2. It upserts `core.performance_snapshots` (spend, impressions, three-second, thruplay, purchases, EUR values, hook rate, hold rate, ROAS, cost per purchase, breakeven, 7-day fields, `winner_loser_status`, `creative_hit`) and writes a `performance` lineage node linked to the code. — automated
3. Production's winners gallery and product creatives show the code, the latest snapshot and the lineage chain (session → concept → asset → export → code → performance). — automated
4. Winner / loser in S1 = ecomprofits' status, shown as-is. In S3, Skynet re-evaluates with the team's own rules (workos "classify" thresholds) and proposes winner tags with a human confirmation gate. — automated + gated (S3)

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT
1 read layer	ecomprofits	`combined_meta_ad_insights`, `meta_ads.parsed_naming`; new view `skynet_creative_daily`, role `skynet_readonly`	`READ`
2 snapshots + lineage	—	—	`NEW` (`core.performanc` `core.lineage_*`)
3 winners show code + chain	static-ads	`pages/winners/WinnersGallery.tsx`, `ProductCreatives.tsx`	`PORT` + `NEW` (line
4 winner rules	ecomprofits (status) → workos (classify knobs)	`business-logic.ts` `calculateWinnerLoserStatus`; workos platform-sync `classify.ts`	`READ` (S1) → `REU` `NEW` gate (S3)

What people stop doing. Looking numbers up ad by ad; retyping winners into two tools; guessing which script produced a result.

Evidence. code-backed. Contract: day-0 lock §4.3. Cards: S5a, S7, V10; week-3 S5b/S9.

W13 · Learnings, memory, lineage

Trigger. A creative wins or loses; a strategist edits, picks or kills a draft; a pattern is judged worth reusing.

Actors. Strategist (ratifies creative/script learnings), team lead (ratifies workspace-wide learnings), media buyer (performance context).

As-is today (code-backed for SuperMemory and RescaleOS; the human loop has no defined practice)

1. static-ads winner uploads and re-ingests sync to SuperMemory through the product-creatives flow (`routes/products.py` upload / reingest → `supermemory_ingest.py`), tracked in `product_creatives_sm_snapshot`; the backfill scripts are repair tooling. — automated
2. RescaleOS stores script-level `learnings`, `sign_off` and `supermemory_writes`; `sign_off` is a human gate, `learnings` may be appended by the nightly loop-closer, `weekly_results` are typed in by hand. — automated + manual
3. **No shared definition of a "learning":** what the team learned from a batch lives in chat, Loom and heads; nothing links evidence → winner → statement. — undefined

To-be in Skynet (recommended; functional design §3.4, §3.7)

1. Lineage is written automatically at every hop (W10, W12): session → concept → asset → export → code → performance → winner. — `NEW` S1
2. A **validated learning** is a reusable statement tied to evidence: `statement + product scope + evidence refs + ratifier + date`. — `NEW` S3
3. It qualifies only from a human-confirmed performance winner (with lineage) or a human-rated output explicitly marked worth reusing. Raw edits, opinions, chat and personal taste never become shared learnings automatically — they stay in the instinct/personal layer unless ratified. — `REPLACES MANUAL` 3, S3
4. Ratifier = the strategist for creative/script learnings; performance-confirmed learnings carry the media-buyer/performance context in the evidence packet. — `NEW` S3
5. Learnings are stored product-scoped first; promotion to workspace-scoped is a separate action ratified by the team lead. — `NEW` S3
6. Review states: `candidate → validated → superseded | rejected`; contradicting evidence marks a learning superseded, never deleted. — `NEW` S3
7. Only validated learnings write to long-term memory (SuperMemory behind one writer); the instinct loop proposes personal checklist rules from edits/picks/kills, ratified by the author, team-wide only after approval. — `PORT` (one writer) + `NEW` S3

What comes from where (explicit)

- From static-ads: winner uploads → SuperMemory `product_<slug>_creatives`; product docs → `product_<slug>` + `_rag`.
- From RescaleOS: `scripts.learnings` + `sign_off`, `ad_category=WINNER` transitions → SuperMemory `winner:v1:…` (`winnerHook.ts`), audited in `supermemory_writes`.
- Intentionally excluded in S1: any automatic learning from performance numbers; personal taste captures; the instinct loop.

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT	SLICE
1 lineage	—	`core.lineage_nodes` / `core.lineage_edges` (card S4); export lineage write (V7); performance lineage write (S7); later cross-surface producers (S15)	`NEW`	S1 → S3
2–6 validated learnings	—	`core.validated_learnings` (S15), winner gate (S9)	`NEW`	S3
7 one memory writer + instinct loop	static-ads + RescaleOS	`supermemory_ingest.py`, `winnerHook.ts`; Instinct Transfer digest (V17/V19)	`PORT` + `NEW`	S3

Evidence. as-is code-backed (no human practice to gather); to-be agreed, awaiting Agon's confirmation. D32 confirms the human-rated-output rule with Mathew.

W15 · Admin: models, spend, settings, users

Trigger. Someone needs to change which model runs a job, add a key, cap spend, or add a user.

Actors. Agon / admin.

As-is today (code-backed)

1. static-ads: core model slugs and the image-model registry live in code (`app/pipeline/config.py` `CLAUDE_MODEL` , `IMAGE_MODEL_IDS`), while some per-user defaults such as `claude_model` are read from the `settings` table (Fernet-encrypted OpenRouter key there too); admin/provider keys live in `system_settings` . Some model changes are settings-driven, but the workflow/model surface is code-shaped, not centrally administered. — mixed (settings + code)
2. meta-ads-scraper: teardown/classifier models and cost caps in its `settings` table via the admin Settings page. — manual, per app
3. RescaleOS: Claude subscription accounts rotated via env pairs; usage dashboard at `/admin/claude-usage` . — manual, per app
4. Users: three separate user tables and logins (static-ads `users` , scraper `admin_users` , console `users`). — manual, three times
5. Spend: OpenRouter dashboard caps set by hand; no per-workflow view. — manual

To-be in Skynet

1. `core.providers / models / model_bindings`: an admin changes the model per workflow key in the Admin page; the next run uses it, no deploy. — `REPLACES MANUAL` 1–2 (S1 for Production; scraper/Studio workflows join at their merge)
2. `usage_events` + `spend_guards`: every model call posts usage; warn at a threshold, pause at a cap, resume from the UI; kill switch. — `NEW` (S1)
3. One user table, one login, roles per workspace for **Skynet itself**; legacy users imported and invited once. The scraper and Studio keep their own logins and settings pages until their merge slices. — `REPLACES MANUAL` 4 (S1, Skynet only)
4. Claude subscription runner stays an internal-only binding inside the Studio runtime (S3). — `PORT (phased)`

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT	SLI
1 bindings	static-ads	`app/pipeline/config.py` L111–216; `system_settings_service.py`	`PORT` → `NEW` tables	S1
2 spend guard	TrenchOS digest (design input)	—	`NEW`	S1
3 users/roles	static-ads, scraper, console	`auth.py`, `admin/lib/auth.ts`, `console/src/server/auth.ts`	`PORT` (legacy user import into Skynet core auth only; scraper/Studio auth stays in place until their merge slices)	S1
4 subscription runner	RescaleOS	`console/src/server/agent/accounts.ts`	`PORT (phased)`	S3

Evidence. code-backed. Cards: S1a, S1b, V4, V5, V8, M1.

W1 · Competitor research and tracking

Trigger. A competitor is worth watching; a tracked competitor's library changes; a product is created and needs its competitors attached.

Actors. Researcher role (Agon is the developer, not a user); strategist as the receiver of findings.

Inputs. Competitor name, Ads-Library page or keyword lens, markets, schedule, alert rule.

As-is today (tooling code-backed; practice confirmed by Agon: no established routine)

1. The scraper admin exists and works (`/brands/new` , CSV upload, quick scrape, deep crawl, per-brand schedule, classification, families, transcripts, `/ads` browse, `/intel` , gold labelling) — but **nobody runs it as a process yet**; it is used ad hoc, mostly by Agon while building it. — no owner, no cadence
2. Crons do the mechanical work unattended: scheduler `*/30` , reaper, brand metrics, transcription, classification. — automated
3. **Manual:** a finding leaves the tool by screenshot, Loom or a hand-typed ClickUp card; the tool has no hand-off action. The static-ads tool runs its own second scraper on the same libraries. — manual, duplicated

To-be in Skynet (agreed with Agon; the design in the functional design §3.0–§3.1)

1. S1: the Intelligence entry in the shell **links** to the standalone scraper; nothing changes in the tool. — unchanged
2. S2: **three triggers, all allowed** — a researcher adds a competitor by hand (name, lens, markets); a product is created or promoted (A15) and its research-run competitors are attached as suggestions; an alert rule fires (new family running ≥ N days, rank rise) and creates an inbox item — never an action by itself. — `PORT (phased)` + `NEW`
3. S2: **routine is per product, not per person** — each product ↔ competitor link carries a schedule (default: weekly quick scrape, monthly deep crawl) and an alert rule; changes reach the owner of the product through the inbox (S3), not by opening the tool on a fixed day. — `NEW`
4. S2: **keep or discard is a human decision with a suggestion** — "silent for 90 days → archive?", "same family as an existing competitor → merge?"; nothing is deleted automatically. — `NEW`
5. S2: **a finding leaves the tool by a button on any ad or family** — *Create script from this* (a brief carrying the ad, its teardown and the angle/metric the researcher selects as the reason), *Recreate* (Production Expand session), *Add to brief* (append to an existing brief), *Tag for teammate* (inbox item with evidence). — `REPLACES MANUAL` 3
6. S2: static-ads' own scraper is retired; collections become tags on the one corpus. — `REPLACES` duplicate

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT	SLICE
1 link-out	—	scraper admin at `meta-ads.rescale.media`	link (no change)	S1
2 triggers + entities	meta-ads-scraper + new	`admin/app/brands/*`, `AddBrandForm.tsx`; `brands` → competitors/lenses (A13a); alerts (A13b)	`PORT (phased)` + `NEW`	S2
3 per-product schedules	meta-ads-scraper	`BrandScheduleCard.tsx`, scheduler cron `server.js:1516` → `product_competitors` (A13a)	`PORT (phased)`	S2
4 keep/discard suggestions	—	—	`NEW` (A13b)	S2
5 action buttons	static-ads + scraper + new	`scraped_ads.py:738` Expand entry; teardown worker; brief entity (A12b); inbox (S14)	`PORT` (Expand) + `NEW`	S2 (S3 for inbox)
6 retire duplicate scraper	static-ads	`services/decodo_client.py`, `scrape_runner.py`, `ad_collections`	retire (A19)	S2

What people stop doing. Running two scrapers; pasting findings into chat; re-registering brands per tool; remembering to check.

Evidence. as-is confirmed by Agon (no routine exists); to-be agreed with Agon. D21 shrinks to: confirm the default schedule, the alert thresholds and the archive suggestion window with the first researcher who uses it.

W2 · Product research → tracked product

Trigger. A niche or keyword set is worth scanning; a ranked opportunity deserves to become a product.

Actors. Product researcher (owns seeds and runs), strategist (approves promotion; Mathew for pilot products), ops/workspace manager (workflow fields).

As-is today (tooling code-backed; practice confirmed by Agon: no established routine)

1. Keywords/niches are seeded by hand on `/research/keywords` or by the CLI seeder; runs start on `/research` (one deep crawl per keyword × country → cluster → score →

`research_products_ranked`). — automated after a click; no owner yet

2. The only exit is **Export CSV**. A chosen product is then created by hand in ClickUp (`products` list, Product ID, economics) and again in the static-ads tool (`/products`, brand identity, docs, looks) — three set-ups, no link between them. — manual, triplicated
3. Rejected or deferred opportunities leave no trace. — manual

To-be in Skynet (recommended from the functional design §3.0–§3.1, user stories 17–18)

1. The **product researcher owns the seeds**; strategists can request, the researcher curates. — `PORT (phased)` S2
2. The app **suggests** keywords from two sources only — repeated terms in tracked-competitor ads and rising terms in the market-intel/VOC corpus — into a review queue; nothing auto-seeds. — `NEW` S2
3. Runs produce the ranked list with reasons, gaps and freshness, as today. — `PORT (phased)` S2
4. **Promotion is a gate:** the researcher recommends, a strategist approves (Mathew for pilot products); researchers never create tracked products alone. — `NEW` S2
5. **Promote to product** creates on day one: the Skynet product, lineage back to the opportunity and run, the suggested competitors/lenses from the run, the ClickUp Product card through the bridge, and a reserved naming product number. — `PORT (phased)` + `REUSE` + `NEW` S2
6. It also creates placeholders for brand identity, docs, reference looks and offer, all flagged **incomplete**. — `NEW` S2
7. Completion is split: researcher → niche, competitors, evidence; strategist → positioning, claims, brand truth; ops → workflow/admin fields. — `REPLACES MANUAL` 2, S2
8. A product cannot enter script generation until its readiness checklist is complete; the app shows the checklist instead of letting thin context leak downstream. — `NEW` S2
9. Rejected/deferred opportunities stay in the list with `status`, `reason_code`, note, `decided_by`, `decided_at`. — `NEW` S2
10. Deferred opportunities get a cool-down (default 30 days) and are hidden from the active view until new evidence reopens them. — `NEW` S2

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT
1, 3 seeds + runs	meta-ads-scraper	`admin/app/research/`, *`lib/research/{runner,cluster,score}.cjs`*, *`research_`* tables (A14)	`PORT` (phased)
2 suggestions	—	tracked-competitor terms (A13a), VOC corpus (A18)	`NEW` (A14)
4–8 promotion + readiness	static-ads + workos + new	`routes/products.py:43,151,248,282`; workos `LISTS.products`, `Product ID`, `rescale_service.next_product_number`; bridge (S8a → A15)	`PORT` + `REUSE` + `NEW` (A15)
9–10 decisions + cool-down	—	—	`NEW` (A14)

What people stop doing. Exporting CSVs; creating the same product three times; losing rejected opportunities.

Evidence. as-is confirmed by Agon (no routine); to-be agreed, awaiting Agon's confirmation. D21 confirms the approver per product and the cool-down default with the first researcher.

W3 · Deep teardown + market intel / VOC

Trigger. A competitor family deserves a full breakdown; a brief needs customer language.

Actors. Researcher / strategist.

As-is today (tooling code-backed; practice: Agon ran the first fan-outs while building; no routine)

1. Teardown per family or brand fan-out from the admin; cron `*/5` drains the worker with Mathew's breakdown prompts; results on `/breakdowns`; model + cost cap in Settings (Qwen won the bake-off, ~$0.007 per teardown). — automated after a click
2. Market intel scopes (subreddit / search) run by cron; documents, threads, aliases, stats pages; **no export** and no in-app search across sources; no scientific-literature source at all. — automated, dead-end
3. **Manual:** a teardown is read on screen; insight and customer quotes are copy-pasted into briefs or chat. — manual

Bridge (temporary, standalone, track B): two as-is paths exist before S2 — (a) the scraper's LFS package: Ads-Library link → long-copy candidates → lander filter → Grok extraction → ranked HTML package dropped into Mathew's folders, driven from RescaleOS by `/lfs`; (b) PubMed evidence via

the RescaleOS `/evidence-layer` skill (six lenses, WebFetch first, optional MCP) — the WHY next to Reddit's HOW. Both are retired by A19/A20/B0 when the native flows below exist.

To-be in Skynet (recommended)

1. Manual teardown on any family/ad stays the baseline trigger. — `PORT (phased)` S2
2. Brand fan-out on the top families of a competitor/product link after a fresh scrape or rank change. — `PORT (phased)` S2
3. **No auto-teardown on every alert.** Auto-teardown is opt-in per product ↔ competitor link, only for alerted families above a strict threshold, not torn down recently, under the daily cost cap, queued as background work with an inbox artefact when done. — `NEW` S2
4. **VOC is consumed in-app first:** search/filter by source, keyword group, date range, product, claim/pain cluster. — `PORT (phased)` + `NEW` S2 (A18, V15)
5. Export stays as a secondary action (by group + date range) for offline analysis or sharing. — `NEW` S2 (A18)
6. **PubMed evidence corpus — the WHY next to the VOC HOW:** six lens scopes per keyword group (mechanism, root cause, symptoms, ingredients, failed solutions, desired outcomes), every study classified by evidence strength, findings in plain customer language with a citation, searchable in-app and exposed as an MCP to the agent runtime and RescaleOS. — `NEW` S2 (A18b)
7. When a teardown or research artefact becomes a brief, the matching VOC packet (top phrases/threads for the product and group, with source metadata and date window) is auto-attached as evidence, and the PubMed evidence packet (top findings per lens, with citation and evidence class) next to it; the user can add/remove items. — `REPLACES MANUAL` 3, S2/S3 (A12b, A18c)
8. Teardown outputs and VOC stay separate artefacts, co-selectable when creating a brief, so persuasion analysis and customer language travel together. — `NEW` S2

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT	SLICE
1–2 teardowns	meta-ads-scraper	`lib/teardown/{worker,prompts}.cjs`, `ad_teardowns`, `enqueue_brand_teardowns` (A12a) — prompt becomes a versioned, Mathew-editable asset	`PORT (phased)`	S2
3 opt-in auto-teardown	—	alert rules (A13b) + teardown queue	`NEW` (A13b/A12a)	S2
4–5 VOC in-app + export	meta-ads-scraper	`lib/market-intel/`, *`mi_`* tables (A18, V15)	`PORT (phased)` + `NEW`	S2
6–7 evidence packets	—	brief entity (A12b), Studio brief queue (V18)	`NEW`	S2 / S3

What people stop doing. Copy-pasting teardown insights and customer quotes; reading market intel in four separate pages.

Evidence. as-is confirmed by Agon; to-be agreed, awaiting Agon's confirmation. D23 confirms the first keyword groups with Mathew.

W4 · Strategy: coverage map, gap finder, next-batch allocation — NEW workflow

As-is today (from the workos code + what the team has said): "what do we make next" is decided by hand — Mathew's judgment, weekly calls, and the media buyer's pasted WoW demand post (W14 step 1). RescaleOS's `frankenstein` and `trend-watcher` loops drop `recommended` script rows, so a small automated proposal channel already exists inside the engine. There is no map of what is covered and what is thin.

To-be in Skynet (S3) — recommended, confirm with Mathew + Cyrus (D32)

1. The coverage map fills itself from scraped data and performance: labels on tracked ads (angle, format, awareness, structure), our own creative codes and their snapshots → cells segments × awareness × angles per product, each with an evidence count and confidence — never a bare heat colour. — `NEW`
2. The gap finder ranks thin cells by importance × reach and writes **proposals**, not orders: "3 net-new concepts for segment X at problem-aware", each with the evidence that produced it. — `NEW`

3. The team lead sets the split (net-new / variations / scaling) — a human gate; the app pre-fills the split from last batch's results and the gap ranking. — gated, `NEW`
4. Approved proposals become brief cards in the queue that feeds Script Studio (W5) and the demand line for Production (W14 step 2); the weekly proposal batch replaces the pasted WoW post as the *default* input, the post stays possible. — `NEW` + `REUSE` (spawn)
5. Every proposal records why (cells, evidence refs, prior results); rejected proposals keep the reason so the gap finder learns which suggestions the team does not want. — `NEW`

Source map: all `NEW` (V18, S15 coverage store); inputs `READ` (ecomprofits performance) + Intelligence labels (S2) + RescaleOS `recommended` rows (folded into proposals, A21). Design inputs: Mathew's system map, TrenchOS digest, Instinct Transfer digest.

Evidence. recommended — the to-be is the functional design §3.3; the only human input still needed is D32 (which decisions Mathew and Cyrus expect the map to make, and what must stay theirs).

W5 · Script creation (NPT / BPT / briefs)

Trigger. A demand line asks for new concepts (Explore) or a rip (NPT); a winning ad deserves a variation; a research sweep or teardown surfaces an opportunity; (S3) an approved proposal from the coverage map.

Actors. Bian (engine owner, runs the loops today), strategists (review, assign), editors (produce).

As-is today (engine code-backed: RescaleOS; the human practice is inferred from the code and marked for Bian's confirmation — D31)

1. Source ads: `/npt` or `/bpt` runs do a competitor sweep and POST a structured dump to `POST /api/research/ingest` → `research_ads` (tier, format, angle, avatar, awareness, hook type…). — automated (Claude subscription runner)
2. Scripts are born via `POST /api/engine/scripts` / `/upsert` into SQLite `scripts` (creative-DNA columns, `content` JSON, `script_revisions`); the hourly `npt-build-loop` on **Bian's Mac** builds the engine and may add `recommended` rows; `frankenstein` (weekly) grafts the best hook onto the best body; `trend-watcher` restamps trends. Workstreams `npt`, `bpt`, `npt-niels`. — automated, Mac-only authoring
3. Humans vote on `/opportunities` (`script_votes`); a strategist assigns on the scripts list (`POST /api/assign`) — this stamps `editor` and an `ad_name` in RescaleOS's own format (`{script_id}_{MARKET}_{TYPE}_{EDITOR}_{YYYYMMDD}`, **different** from the workos v3 ad name) and flips status. — manual
4. Editor uploads `produced_ad_url`; strategist sets `ad_category` + `sign_off`. — manual
5. KPIs are **typed by hand** into the WEEKLY RESULTS grid (`weekly_results`); Monday `kpiReminder` nags; no importer from ecomprofits. — manual

6. Nightly `loop-closer` writes `learnings` + `ad_category` (if blank) and recycles winning hooks; `feedback_inbox` polls Telegram/Discord; `backlog` digests feedback for Bian's accept. Cloud runs only the operational jobs; authoring stays on the Mac; skills edits Mac-only. — automated, split across two machines
7. In workos, Scripts are ClickUp cards spawned by the demand cascade; `Script → approved` spawns Creatives. RescaleOS scripts and ClickUp Script cards are **not linked**. — duplicated
8. **Inferred, confirm with Bian:** how much of a script the engine writes vs Bian edits; which gates he applies before "ready"; what he refuses to let the engine decide. — `pending` (D31)

To-be in Skynet (S3) — recommended

1. S1: link-out to the standalone console; nothing changes for Bian. — unchanged
2. Script Studio native: agent sessions with the team's skills and human gates, per-user overlays, scripts in unified Postgres (`studio` schema); the engine runs as a service in the stack, the subscription runner as an internal-only binding; every loop (build, dream, lint, backlog, loop-closer, frankenstein, trend-watcher) runs on the stack under one scheduler; skills and prompts are versioned assets edited in-app. — `PORT (phased)` (A10, A11, A21, V14, V23)
3. **Scripts start from evidence the system proposes** — a coverage proposal (W4), a teardown + VOC packet (W3), a winner to vary, or a demand line — and a human approves at the gates; the engine's `recommended` rows become proposals in the same queue. — `NEW` + `PORT (phased)`
4. The hook gate comes first (W6); KPIs come from readback (W12), never typed; the RescaleOS ad-name format is retired in favour of the creative code + v3 ad name. — `REPLACES MANUAL` 5, `REUSE` naming
5. One script entity feeds the ClickUp Script card through the bridge until S4 retires the board (S16). — `REUSE` + `REPLACES` duplicate 7
6. Bian's coexistence: both run during the migration window, the stack becomes authoritative per D31, the Mac jobs are switched off last (A20). — gated by D31

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT	SLICE
2 Studio native	RescaleOS	`console/src/db/client.ts`, `src/server/agent/`, *`engine/skills`* *(45) +* *`_doctrine`* *+* *`memory`*, *`engine/automation/`*, `night-shift-jobs.cloud.json`	`PORT (phased)`	S3
3 evidence-first proposals	RescaleOS + new	`frankenstein.sh`, `npt-trend-watcher.sh` → proposals; W4/W3 inputs	`PORT (phased)` + `NEW`	S3
4 KPIs from readback	RescaleOS `weekly_results` → ecomprofits	`POST /api/scripts/results`; W12	`READ` replaces manual	S3
5 one script entity ↔ ClickUp	workos	`forward.ts:186`, `LISTS.scripts` (S16)	`REUSE`	S3

What people stop doing. Typing KPIs; running authoring on one laptop; keeping two script records; starting from a blank page.

Evidence. engine as-is code-backed; Bian's practice inferred (D31 confirms); to-be recommended.

W6 · Hook session + approval gates — NEW workflow

As-is today (code-backed): hooks live inside scripts (RescaleOS `hook`, `hook_type` columns); `frankenstein` grafts the best-hook-rate opener onto the best-hold-rate body; approval = the strategist's `sign_off` in the console and the `approved` status on ClickUp Script/Creative cards (workos QA gates: assignee + parent + required fields, enforcement off by default). Who approves what in practice is undocumented — D32.

To-be in Skynet (S3) — recommended, confirm with Mathew + Bian (D32)

1. A hook session over-produces (N hooks per brief), culls against the brand-tone band, claim validation and the one question "would this target stop scrolling?", using live competitor hooks (scraped, labelled) and the VOC corpus as evidence; only survivors are shown. — `NEW` (V16)
2. The strategist picks keepers and kills the rest **with a reason**; each kill is a signal for the instinct loop (V17). — gated, `NEW`
3. Approval is per **batch** for hooks (fast) and per **asset** for scripts and creatives (accountable) — recommended default, D32 may change it. — gated

4. Hook rate is read back per launched hook (W12) and shown next to the session so hooks are judged on their own job. — `READ`
5. Gates carry an owner, an evidence packet and a "waiting on" state visible in the inbox (S14). — `NEW`

Source map: `NEW` (V16, V17, S14) + `READ` (hook rate) + inputs from Intelligence labels (S2) and VOC (A18); seed content from Mathew's Hook Engine digest under claim validation.

Evidence. recommended; D32 confirms who approves what, per-asset vs per-batch, and the evidence required.

W7 · Static production (Explore / Expand / Long-form)

Trigger. An approved script (Explore), a competitor or own ad to adapt (Expand / Expansion), a lander to write long-form ads from.

Actors. Strategist / designer (Vince's tool users).

As-is today (code-backed: static-ads-automation)

1. Product setup: `/products` → brand identity (`PUT /brand-identity` , audit log), memory prose, product looks (R2), documents (embedded to pgvector + SuperMemory). — manual once
2. Explore: `/products/:id/sessions/new` — paste script or winner image, aspect, N images, looks, image model, funnel stage or awareness, language, knowledge source → `POST /sessions` → pipeline steps 1 analysis → 2 concepts (12) → 2 copy-QA/translate → 3 image gen (OpenRouter image models) → 4 vision review; SSE progress; retry from step. — automated
3. Human approves concepts (`PUT /approvals`), winners appear on `/winners` . — manual gate
4. Expand: drop competitor images → OCR → localize → clone render. Expansion: own ads → faithful translation → text-only re-render. Also auto-Expand from a scraped ad. Long-form: lander URL + avatar level → N long-form ads with images. — automated
5. **Download:** per image or client ZIP (`downloadImagesAsZip`), `winners.zip` . — manual
6. **Manual, outside the app:** rename files to the Meta naming convention (BE tokens etc.), upload in Ads Manager by hand, paste the ad name from ClickUp. Nothing calls the Marketing API. — manual, error-prone
7. Winner upload back into the tool with a hand-typed creative code (W10 as-is step 4). — manual

To-be in Skynet

1. S1: all of the above ported natively into Production on unified Postgres, unchanged in behaviour (parity checklist per feature). — `PORT`
2. S1: export issues the creative code and stamps files + manifest + ad name (W10). — `REPLACES MANUAL` 6 (renaming), 7 (typing)

3. S1: every export writes lineage (session → concept → asset → export → code). — `NEW`
4. S1: model choice per workflow from bindings; spend guard. — `NEW`
5. S2: Expand from a tracked ad inside Intelligence; static-ads' own scraper retired. — see W1
6. S3: locked product spec, prompt layers, statics-as-testing-ground before video (functional design §3.5). — `NEW`

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDIC
1 Production native	static-ads	`routes/{products,sessions,winners}.py`, `pipeline/{orchestrator,expand,expansion,longform}.py`, `pipeline/steps/`, *frontend* `pages/products/`, `winners/*`	`PORT`
2 stamped export	static-ads + workos	`export_service.py`, `download.ts`; W10	`PORT` + `REUSE`
3 lineage	—	—	`NEW`
4 bindings + guard	static-ads	`pipeline/config.py`, `claude_client.py`, `openrouter_client.py`	`PORT` → `NEW`

What people stop doing. Renaming files; typing codes; deploying to change a model.

Evidence. code-backed. Interview (Vince): where humans intervene in practice, which metadata must survive to export, which regressions would block cutover.

W8 · Video production (storyboard / clips / narration / assembly)

As-is today (code-backed: static-ads-automation)

1. `/video-storyboard/new`: product, approved script, optional reference video / song / character → deterministic split into `storyboard_scenes`. — automated
2. Copy board: human edits/splits/merges/reorders scenes, generates shot prompts. — manual
3. Stills per scene (`/stills`, `/stills-all`); human **picks** one per scene, refines. — manual gate
4. Render (`/render`, `/render-all`) → leased `storyboard_jobs` → worker → ElevenLabs narration with timestamps → fal Kling / HeyGen per `RENDER_PROVIDER` → raw / mute / voiced variants; optional ChatCut assembly to one MP4. — automated

5. Delivery: `frames.zip`, `clips.zip`, `manifest.json`, `audit.json`; on-screen text is deliberately not rendered — the editor composites it from the manifest. — manual hand-off
6. **Manual:** editor assembles the final cut in their tool; final MP4 uploaded to Meta by hand; the launch card gets the URL by hand. — manual

To-be in Skynet

1. S1: ported natively, unchanged (`PORT`); delivery bundle carries the creative code + manifest (W10). — `PORT` + `REPLACES MANUAL` (naming)
2. S1: the editor's deliverable lands as the export package on the existing ClickUp Launch card via the bridge (functional design user story 15). Native inbox handling is S3; the native launch board is S4. — `NEW`
3. S3: video recipes registry, hooks from the hook session, auto-assembled cut accepted or replaced. — `NEW`

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY
1 storyboard native	static-ads	`routes/video_storyboard.py`, `services/storyboard_{queue,runner,narration,delivery}.py`, `worker.py`, `pipeline/video_storyboard.py`, `storyboard_llm.py`
2 delivery → launch task	—	—
3 recipes	—	—

Evidence. code-backed. Interview (Vince): which steps are brittle, what "delivered" means for media buying.

W9 · Canva polish + localisation

As-is today (code-backed)

1. Canva: admin connects once (OAuth, tokens encrypted); from a session the human clicks **Open in Canva** (`POST /api/canva/open` uploads the asset, creates a design), edits, returns (`/auth/canva/return` → `POST /api/canva/return` stores the polished bytes in `canva_polishes`, many per image); gallery `/gallery/canva-edits`. — manual edit, automated round-trip

2. Localize video: upload MP4, pick language → headless pipeline (transcribe → translate → ElevenLabs TTS → drift absorption → optional music preservation → optional on-screen caption replacement → mux) → download. — automated
3. **Manual:** naming of variants per market; deciding when polish/localisation is mandatory; upload by hand. — manual

To-be in Skynet

1. S1: both ported natively; the polished version stays linked to the original (lineage) and inherits the creative code with a variant/market marker per the naming rules. — `PORT` + `NEW` (lineage)

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT	SLICE
1 Canva + Localize native	static-ads	`routes/canva.py`, `services/canva_service.py`, `routes/localize_video.py`, `pipeline/localize_video.py`, `onscreen_*.py`	`PORT`	S1

Evidence. code-backed. Interview (Vince): when Canva/localisation is mandatory, how variants are named, what comes back vs stays external.

W11 · Launch preparation + media-buyer hand-off

Scope. From "creative approved" to "launch package / launch entity exists and is handed to the buyer". Board lifecycle after that is W14.

Actors. Strategist (approves the creative), the workos cascade, the media buyer.

As-is today (cascade code-backed: workos; the buyer's practice inferred from the Launch card fields — confirm in S0b/D41)

1. Creative card → `approved` in ClickUp → `handleCreativeApproved` (`forward.ts:550`) resolves Campaign, Funnel, Adset (skipped for Pinterest), creates the **Launch** task `[product] launch — <creative>` (status `briefed` , checklist brief), mirrors `Creative URL` , `Headline` , `Ad copy` , sets six relationships + `Channel` , bumps adset `Ad count` , tries the ad name. — automated
2. Machine fills over time: `Ad name NC` , campaign/adset names, later `Platform ad ID` , spend, ROAS, KPI met, Launched date, Zombie, Funnel missing?. — automated (W14 platform-sync)
3. **Manual (media buyer), inferred from the human-owned fields:** open the Launch card, use the `Creative URL` on the linked Creative/Launch card, paste the ad / campaign / adset names into

Meta, build the ad, set account and budget, set the assignee (the Media Buyer token in the ad name), later flip the Friday-call outcome status. — manual

4. **Unknown until S0b:** request source per brand, approval chain, asset packaging (Drive vs ClickUp), who publishes, where it breaks. — `pending`

To-be in Skynet — recommended

1. S1: Production's export attaches the export package (code, files, manifest, ad name) to the already-existing ClickUp Launch card through the workos bridge (`POST /launches/:id/package`); the buyer keeps working in ClickUp as today. The bridge writes only `Creative URL` if empty, one package comment and the `skynet-package` tag; never status, assignee or a machine-owned field. — `REUSE` + `REPLACES MANUAL` 3 (rename/retype only)
2. S4: the launch board is native — the buyer picks up a task that already carries assets, code and ad name, pre-launch statuses writable, outcome statuses observe-only; ClickUp mirror as rollback until the board flips. — `PORT (phased)` (S8b, V9)
3. S4: the launch board carries the launch package natively; any campaign/adset placement or budget suggestion stays discovery-bound until it is carded explicitly after D41. — deferred, not yet carded
4. S4: proof of launch comes back automatically (Platform ad ID via platform-sync → Launched); the buyer stops flipping statuses by hand. — `REPLACES MANUAL`

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT	SLICE
1 launch package via bridge	workos	`forward.ts:550` (`handleCreativeApproved`), `LISTS.launches` , `FIELDS.launches` , `clients/clickup.ts` ; `registerSkynetBridgeRoutes` + `POST /naming/ad-name` + `POST /launches/:id/package` (S8a)	`REUSE`	S1
2 native board	workos + new	S8b, V9	`PORT (phased)` + `NEW`	S4
3 placement/budget proposals	workos + ecomprofits	Product card economics fields (`FIELDS.products`), `platform-sync/classify.ts` knobs	deferred until carded after D41	S4+
4 proof of launch	workos	`sync-launches.ts` (LIVE flip on first spend)	`REUSE`	S4

What people stop doing. Downloading, renaming, retyping names; hunting for the right file; flipping statuses by hand.

Evidence. cascade code-backed; buyer practice inferred — S0b/D41 (Cyrus + one media buyer) confirm the exact package, what they still rewrite, what delays launches.

W14 · Workspace: boards, demand, approvals, reconciliation (ClickUp)

Scope. Everything that lives in the 13 creative-workflow ClickUp lists and the workos cascade after a launch entity exists, plus demand intake and board hygiene. Company-ops lists stay in ClickUp by decision.

Actors. Media buyers, strategists, editors, ops; the workos service.

As-is today (code-backed: rescale-workos; operating exceptions inferred — D42 confirms with Sven + the ops owner)

1. **Demand intake:** a media buyer pastes the weekly "WoW post" into `Raw post` on a Demand Week card; status → `parsed` runs `parseDemand` (OpenRouter extraction + validation) into Demand Line subtasks + bounce rows; a human fixes, flips → `locked` → `spawnLineOnce` creates Scripts (Explore) or Creatives (Expand / relaunch / NPT / winner iteration). The Monday `seedDemandWeek` exists but is not wired — weeks are created by hand. — manual + automated
2. **Platform-sync** every 15 min reads ecomprofits (`meta_ad_insights`, campaigns, adsets, Funnelish, the unified view, Shopify/BestFulfill economics, `get_creatives_ranking`), matches by `Platform ad ID` → CU id → name, writes `Platform ad ID`, spend, ROAS, the LIVE flip, `KPI met`, `Zombie`, `Classification note`; knobs from the Thresholds Google Sheet; never flips Winner/Loser. — automated
3. **Reconcile** every 15 min (structural, provenance, week batch, slots, launch outcomes, fulfillment, rip sources, funnel availability, orphans, gates, briefs, names); **bounty** weekly → Payroll Requests; **QA gates** dry-run unless `QA_GATES_ENFORCE`. — automated
4. **Human:** the Friday call flips outcome statuses (Winner / Super Winner / Loser) — propagates to Creatives and Scripts; hand-created cards; fixing bounce rows; exceptions and workarounds nobody wrote down. — manual, `pending` (D42)
5. Authoritative today: ClickUp for every list; `rescale_service` (ecomprofits Postgres) for sequences and rankings; `FIELDS.<list>` marks machine- vs human-owned fields; `ugcCreators` has no field definition at all. — code-backed

To-be in Skynet — recommended

1. S1: Workspace entry = link to ClickUp; workos unchanged on Railway. — unchanged
2. S3: demand is **proposed** from the coverage map / allocation (W4) instead of a pasted post — the post remains an input; Skynet creates Demand Lines through the bridge and the cascade spawns as today. — `NEW` + `REUSE` (S16)
3. S3: outcome statuses are **proposed** by Skynet's winner rules (W12, S5b) and confirmed by a human (S9) — the Friday call becomes a confirmation of proposals, not the source of the data. — `NEW` + `REUSE` rules
4. S4: board-by-board cutover to native boards in the order D42 names (launch board first), ClickUp mirror as rollback, workos into the stack, ClickUp read-only for the 13 lists; company-ops lists untouched. — `PORT (phased)`
5. S4: QA gates enforced natively (assignee + parent + required fields); bounty keeps running from the native stores. — `PORT (phased)`

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY	VERDICT	S
2 demand from allocation	workos	`demand-week.ts` (`seedDemandWeek`, `handleParsed`, `handleLocked`), `spawn.ts`, `demand-parser.ts`	`REUSE` (spawn) + `NEW` (source)	S
3 outcome proposals	workos + ecomprofits	`platform-sync/{cron,insights,classify,match}.ts`, `thresholds.ts`; W12	`REUSE` rules → Skynet gate	S
4 native boards	workos	`domain/ids.ts` (13 lists, `FIELDS`), cascade modules, `reconcile-cron.ts`, `qa-gates/`, *`bounty`*	`PORT (phased)`	S
5 gates + bounty	workos	`qa-gates/{gate,enforce}.ts`, `rules/bounty*.ts`	`PORT (phased)`	S

What people stop doing. Pasting the WoW post; creating weeks by hand; reading numbers in two places; deciding outcomes without a proposal.

Evidence. code-backed for the cascade; exceptions/authority inferred — D42 confirms with Sven + the ops owner.

W16 · Team chat: notifications, chat ingestion, agents in chat (Telegram + Discord)

Trigger. A job finishes (scrape, teardown, research run, batch), a rule fires (alert, spend guard, winner proposal), a teammate needs someone's attention, or a question is asked in chat.

Actors. Everyone in the team channels; the Skynet bot; the agents (Studio, Intelligence).

As-is today (code-backed)

1. **ecomprofits** has a real notification platform (`packages/features/notification-platform`): an event registry with typed payloads, `emitNotificationEvent` → outbox `notification_events` with a dedup key, a dispatcher worker + sweeper, per-account destinations (Discord channels by purpose — budget, performance, system — and Telegram chats — ad alerts, system alerts, ops digest), mutes and digests; naming-convention violations are one event family (`naming-violation-event-factory.ts`, severities critical/warning/info, per-kind filters, CSV attachment). Agon built it. — automated, the pattern to copy

2. **RescaleOS** polls Telegram (NPT + BPT) and Discord into `feedback_inbox` (10-min tick) and clusters it into the backlog for Bian; the console posts notifications to users. — automated, inbound exists
3. **workos** posts to Discord channels (machine room, system alerts, store monitors) and Telegram from crons; alerts have no quick actions. — automated, outbound only
4. **Manual:** the team asks each other questions in chat; nobody can ask the system; scraping/research results are read in the tool, not announced; nobody scrapes team chats for signals. — manual

To-be in Skynet — recommended (functional design user stories 9, 27)

1. S3: one **notification platform** in Skynet on the ecomprofits pattern: event registry (typed payloads per feature), `core.notification_events` outbox with dedup, dispatcher + sweeper, per-workspace destinations (Discord channels by purpose, Telegram chats), mutes, digests; every job and rule in Skynet emits through it — never inline sends. — `PORT` (from ecomprofits, one adapter) S3
2. S3: **the bot tags humans with proposals and quick actions**: "scrape of <competitor> finished — 3 new families, 1 alerted; suggest: teardown / add to brief / ignore"; "research run ranked 12 opportunities — top 3 …; promote?"; "winner proposal for <code> — confirm / reject"; each message carries buttons/replies that call the same Skynet actions with the same permission checks, and lands in the inbox (S14) with the evidence packet. — `NEW` S3
3. S3: **the team tags and asks the agent in chat**: `@skynet` (Discord) / the bot (Telegram) in a thread → an agent session with the thread as context (product, ad, brief detected from links/codes), answers in the thread, can be asked to act ("create a script from this", "show hook rate for RCi31-A"); every action goes through the gates; the transcript is attached to the inbox item. — `NEW` S3
4. S2/S3: **chat ingestion**: selected Telegram/Discord channels (team discussion, customer/community channels) become **sources** in the market-intel corpus and the feedback lanes — messages ingested with consent per channel, author salted-hashed like the market-intel sources, tagged by channel/keyword group, searchable in-app next to Reddit/Trustpilot/Amazon; team-feedback channels feed the backlog as RescaleOS does today. — `PORT (phased)` (RescaleOS pollers, market-intel adapter) + `NEW` S2 (sources) / S3 (feedback)
5. S3: every notification is also an **inbox item** with owner and state, so chat is a surface, not the record. — `NEW` (S14)

Source map

TO-BE STEP	SOURCE	CODE PATH TODAY
1 platform	ecomprofits	`notification-platform/src/server/{emit,dedup,destinations,mutes}` `events/registry`, `naming-violation-event-factory.ts`
2 bot proposals + quick actions	workos + new	`DISCORD_*_CHANNEL_ID` posts, `webhooks.ts` shape; Skyn actions + permissions
3 agent in chat	RescaleOS + new	`feedbackPollers/{telegram,discord}.ts`, `managedSession.ts` (session with context)
4 chat ingestion	RescaleOS + meta-ads-scraper	`feedbackPoller.ts`, `lib/market-intel/ingest-worker.` `mi_scopes` (`scope_type` gains `telegram_channel`, `discord_channel`); team-feedback channels ported into inbox (S14)
5 inbox record	—	S14

What people stop doing. Reading results in the tool to find out something finished; asking each other what the system knows; losing chat signals.

Evidence. as-is code-backed (three systems); to-be recommended from Agon's ask ("Telegram and Discord scrapers; agents the team can tag and ask; the bot tags humans, e.g. scraping finished, this is what we suggest"); D33/D32 confirm channel consent rules and which channels are sources.

Cross-reference and evidence status

WORKFLOW	APPENDIX A ROWS	CARDS (S1)	EVIDENCE
W1	A.1 brands/scrape/crawl/browse/families	— (S2)	mixed
W2	A.1 research	— (S2)	mixed
W3	A.1 teardown/market-intel	A18 (S2)	mixed
W4	—	— (S3)	pending
W5	A.3	— (S3)	mixed
W6	—	— (S3)	pending
W7	A.2 Explore/Expand/Long-form/products	V3, V7, V10, M1	code-backed
W8	A.2 Storyboard	V3, V10	code-backed
W9	A.2 Canva/Localize	V3, V10	code-backed
W10	A.4 naming, A.5	S3, V7, S8a	code-backed
W11	A.4 launches	S8a (S1); S6, S8b, V9, S11 (S4)	pending (S0b)
W12	A.5	S5a, S7	code-backed
W13	A.2 SuperMemory, A.3 learnings	S4, V7	mixed
W14	A.4	— (S3/S4)	mixed
W15	A.2 settings, A.3 accounts	S1a, S1b, V4, V5, V8	code-backed
W16	ecomprofits notification-platform; A.3 feedback pollers; A.4 Discord posts	S27, A23, A24 (S2/S3)	recommended

Interview plan (as-is sections marked pending or mixed): Agon + weekly operator (W1, W2, W3) · Mathew + Cyrus (W4) · Bian (W5, W6) · Mathew + Bian (W6) · Vince (W7, W8, W9, W13) · Cyrus + one media buyer (W11 — S0b) · Sven + ops owner (W14). Three to five questions each are listed in the delivery plan's discovery cards.