Route4 — A Non-Dependent Release Engine

Route4 is a system for releasing work with restraint, ritual, and trust — before, during, and after public release.

What it is not:

- Not a social network
- Not a marketing tool
- Not a recruitment platform

What it is: A release engine that builds audience through documented care, protects authorship, and treats capital as an outcome—not a driver.

Mary is the first client of Route4.

Core Philosophy

Route4 exists to help creators:

- **Build audience** before release without leaking story
- **Reward presence** instead of popularity
- **Expose process**, not outcomes
- **Preserve authorship** and canon
- **Let capital emerge** as a byproduct, not a dependency

The Two Laws

Every decision in Route4 defers to these laws:

1. **Expose decisions, not outcomes.**
2. **Reveal authorship last.**

How Route4 Works

Three Components

Route4 operates as three distinct layers:

- **Platform** — Release cycles, ritual stages, visibility levels, archive, invitations, Discord orchestration
- **Client Configuration** — Story, timing, ritual names, language, tone, credits
- **Execution Layer** — Discord hosts rituals; Angular app surfaces content

The platform never owns story. The client configures everything else.

Entry States

Work enters Route4 in one of two states:

Entry State A — Release-Ready Work

When: The work is finished or near-finished.

Examples:

- Locked or near-locked film / episode
- Mastered album or single
- Completed documentary

Route4 orchestrates:

- Public release
- Canon protection
- Presence structure
- Timing, silence, and revelation

This is Route4's **core use case**.

Entry State B — Release-Oriented Preparation

When: The work exists but isn't yet ready for public release.

Creators seeking:

- Careful exposure without dependency
- Preparation for eventual release
- Story protection during development

Route4 ensures:

- Story leakage is prevented
- Development constraints are imposed
- Transition to Entry State A is clear

Route4 does not: become a collaboration platform.

The Tools Metaphor

Outreach, Route4, and Discord each serve one purpose:

```
Outreach Tools    →  Deliver invitations (outside Route4)
Route4            →  Receive & structure presence
Discord           →  Host rituals (inside Route4)
```

Each tool handles its job. None does another tool's work.

Visibility Levels

These levels prevent story leakage while maximizing value.

Level	Access	Purpose	Examples
L0	Private (Voltron only)	Core team secrets	Scripts, edits, canon debates
L1	Witnessed Process	Craft observation	Writing sessions, shot councils, decisions
L2	Fragments (Public)	Residue, no context	BTS clips, anonymous signals, tools
L3	Revelation (Public)	Full story access	Episodes, credits, meta reflection

---

🎬 The Rituals (Master Canon)

Route4 executes a repeatable sequence of **rituals** ordered by when they occur.

Mary's Release Cycle

1. **Signal I — Casting Call** (L2 public)
2. **Table Read** (L1 witnessed)
3. **Writing Table** (L1 witnessed)
4. **Shot Council** (L1 witnessed)
5. **The Hold** (silence)
6. **Signal II — Anonymous Song Drop** (L2 public)
7. **The Drop** (primary release, L3)
8. **The Echo** (reflection opens, L3)
9. **Fragments** (residue, L2)
10. **The Interval** (meta podcast, L3)
11. **Private Viewing** (witness-only, L3)
12. **Archive** (permanent, L3)

🎪 Audience Formation — Early Stage

This section defines how Route4 forms an audience starting from zero.

Step 1 — Internal Readiness ✓

Status:

- Script exists
- Core team assembled
- Table reads occurred (undocumented)

Action:

- Acknowledge that history begins *now*
- Do not attempt to recreate undocumented past

Step 2 — Signal I (Casting Call)

First public-facing action.

Purpose

- Establish legitimacy
- Invite participation without revealing story
- Create the first public artifact

What Route4 Hosts

Route4 is the **canonical location** for the casting call:

- Project status (e.g., "Script complete")
- Tone and intent
- Roles or participation sought
- Constraints (what this is / is not)
- How to respond

No comments. No discussion.

Outreach Delivery

Outreach happens **outside** Route4.

Approved channels:

- Local acting groups
- Theater mailing lists
- Filmmaker communities
- Direct emails / DMs

Outreach message rules:

- Short
- Identical across channels
- Points to Route4
- No external discussion

Example:

> "Episodic indie project. Script complete. Intentional casting process. Details here: [Route4 link]."

Conversion to Route4

Once someone clicks the link:

- They enter Route4
- They encounter rules, tone, and constraints
- Outreach ends. Route4 begins.

Step 3 — Intake Without Noise

Route4 records:

- Who arrived
- When
- What they were invited to

Route4 does **not:**

- Rank arrivals
- Promote participation
- Expose metrics

Step 4 — Table Read (Documented)

First ritual conducted *with documentation*.

Route4 records:

- Audio (minimum)
- Stills if appropriate
- Presence (who was there)

Discord role:

- Temporary room
- Silent Witness mode
- Time-bound access

Archives as: non-canonical process artifact.

Step 5 — Silence (The Hold)

After the Table Read:

- Discord locks
- Route4 updates state
- No outward communication

This silence is **intentional**.

Ritual Details (Expanded)

Signal I — Casting Call

Purpose: Invitation without information

- Public-facing casting call
- No story context

- No character outcomes
- Establishes tone and intention only
- Signals that a world is forming

Visibility: L2 (Public)

Table Read (Witness Shadow)

Purpose: Validate voice and emotional truth

- No cameras
- No blocking
- Witnesses observe silently
- Dialogue and rhythm only
- Audio recorded for archive

Visibility: L1 (Witnessed Process)

Writing Table

Purpose: Lock story intent

- Decisions over exposition
- Craft revealed through discussion
- No plot summaries
- Witnesses observe craft, not outcomes

Visibility: L1 (Witnessed Process)

Shot Council

Purpose: Translate story into visual intention

- Why shots exist, not what they show
- Safe stills and masked previews only
- Craft of vision-making exposed
- Outcomes protected

Visibility: L1 (Witnessed Process)

The Hold

Purpose: Compression before release

- Intentional silence
- No explanation
- No engagement
- System-enforced, not optional

Visibility: L1 → L0 transition

Signal II — Anonymous Song Drop

Purpose: Emotional alignment without authorship

- Anonymous artist/songwriter/producer
- Discovery-first listening
- No credits revealed
- Shapes emotional tone

Visibility: L2 (Public)

The Drop (Primary Release)

Purpose: Public release

- Episode / work release
- Read-only window (24h)
- Credits locked
- Canon sealed
- No editing after publication

Visibility: L3 (Revelation)

The Echo (Reflection)

Purpose: Reflection without spoilers

- Opens after delay
- Reflection-only discussion
- No theories, no spoilers
- Witnesses and public mixed

Visibility: L3 (Controlled Reflection)

Fragments (Residue)

Purpose: Residue, not recap

- BTS artifacts
- Tools, hands, rooms
- No explanations
- No context
- Extended atmosphere

Visibility: L2 (Public)

The Interval (Meta Reflection)

Purpose: Meta conversation

- Podcast or long-form discussion
- Post-event only
- Behind-the-scenes discussion
- Creative process explored

Visibility: L3 (Reflection)

Private Viewing (Witness-Only)

Purpose: Scarcity and presence

- Witness-only attendance
- Private location (QR-based access)
- No recording
- One-time, no replay
- Creates scarcity through restraint

Visibility: L3 (Witness-gated)

Archive (Permanent)

Purpose: Preserve meaning

- Canon artifacts only
- Versioned history
- Immutable record
- Ongoing access for witnesses

Visibility: L3 (Permanent)

Presence Model

Route4 structures **presence**, not engagement.

Presence Answers Four Questions

Question	Meaning
Who may be present?	Role-based (public, witness, core team)
How may they be present?	Interaction mode (read-only, silent, discussion)
When is presence allowed?	Ritual window (scheduled, time-bound)
What does presence change?	Status effects (witness status, archive access)

Audience Presence Mapping

Ritual	Who	How	When	Presence Effect
Signal I — Casting	Public	Read-only	Open window	Signals world formation
Table Read	Witness	Silent observation	Scheduled	Witness status recorded
Writing Table	Witness	Read-only listen	Scheduled	Decisions observed
Shot Council	Witness	View artifacts	Scheduled	Visual literacy
The Hold	None	Silence	Fixed duration	Compression
Signal II — Song	Public	Listen-only	Short window	Emotional alignment
The Drop	Public	Read-only 24h	Fixed moment	Canon sealed
The Echo	Public/Witness	Reflection	Delayed open	Meaning emerges
Fragments	Public	Passive	Post-drop	Atmosphere extended
The Interval	Public/Witness	Listen/watch	Post-event	Meta reflection
Private Viewing	Witness	One-time only	Fixed	Scarcity
Archive	Public/Witness	Read-only	Ongoing	Canon preserved

💬 Discord as Execution Layer

Discord is **never the front door**. Route4 treats Discord as a **ritual execution surface**, not a community hub.

Operational Rule

> Route4 decides state.
> Discord reflects state.

Discord never advances a release, unlocks meaning, or defines eligibility.

Discord Invite & Presence Flow

Step 1 — Route4 Intake (No Discord)

- Visitor arrives via Splash or Casting Call
- Route4 records: identity, timestamp, invitation context
- **Discord is not shown**

Status: Known to Route4, not present in Discord

Step 2 — Eligibility Determination (Route4)

Based on release state and ritual configuration, Route4 determines:

- Whether Discord access is required

- What role should be granted (e.g., Witness)
- What channels should be visible

No Discord action occurs yet.

Step 3 — Ritual Window Opens (Route4 → Discord)

When a ritual requires live presence:

Route4 generates:

- Time-limited invite link
- Bound to ritual + role + expiration
- Unlocks required channels

The app exposes **one action:** "Enter Ritual Room"

Step 4 — Join & Permissions (Discord.NET)

On join:

- Route4 assigns role(s)
- Permission presets applied
- Presence logged (who/when/ritual)

Discord hosts interaction **within the ritual window only**.

Step 5 — Ritual Close (Route4 → Discord)

When the ritual window ends:

- Channels locked
- Invite expires
- No new joins permitted

Discord returns to dormant state.

Permission Presets

Route4 applies these Discord permission modes automatically:

Mode	Permissions
Read-Only	View messages, no posting
Silent Witness	View + listen, no chat, no reactions
Reflection	Limited posting, slow mode enabled

Mode	Permissions
Locked	No access
Admin Only	Core team only

Discord Channel Templates

Route4 provides neutral, reusable templates (client may rename):

Orientation:

- `#signal` — Pre-release artifacts
- `#how-to-witness` — Participation rules
- `#start-here` — Entry point

Releases:

- `#releases` — Primary release drops
- `#soundtrack` — Companion audio

Reflection:

- `#after-the-drop` — Post-release discussion
- `#interval` — Podcast and meta reflection

Residue:

- `#fragments` — BTS artifacts

Process (Time-Locked):

- `#writing-table` — Opened only during sessions
- `#shot-council` — Opened only during sessions
- `#cut-room` — Opened only during sessions

Private:

- `#core-team` — Voltron only
- `#canon-drafts` — Voltron only

🌐 Web App (Angular)

The Route4 app is the **spine** of the platform.

What It Provides

- Official releases
- Witness identity & verification
- Archive access
- Invitations & ritual notifications
- Deep links to Discord

What It Does NOT

- Host conversation
- Replace Discord
- Optimize for engagement

Navigation Structure

- **Signal** — Pre-release artifacts
- **Episodes** — Releases
- **Archive** — Complete history
- **Voltron** — Core team (admin)
- **Profile** — Witness status & invitations

🎞 Route4-FFmpeg API (Media Pipeline)

Route4 includes an **FFmpeg-powered media pipeline** that generates artifacts required by rituals and visibility levels.

This is **platform law**:

- FFmpeg capabilities exist to **enforce restraint**, **protect confidentiality**, and **automate ritual artifacts**
- Route4 does **not** expose codec/FFmpeg knobs to creators
- Creators choose **intent** (Fragment / Process Preview / Release Renditions), Route4 chooses the safe implementation

Three Primary Jobs (MVP)

1. CreateFragmentProxy (L2)

Auto-trim to 15–90 seconds with:

- Optional mute / room-tone
- Safe crop/blur/mask to avoid spoilers
- Multi-aspect outputs (9:16, 1:1, 16:9)
- Optional subtle watermark

2. CreateProcessPreview (L1)

Muted previews for craft sessions with:

- Low-res proxies
- Optional burn-in: "WITNESSED PROCESS — NO OUTCOMES"
- Crop/mask regions that leak dialogue/plot
- Still extraction for color/scopes discussion

3. CreateReleaseRenditions (L3 + operational)

- Public release encode(s)

- Private master / archive encode
- Private Viewing encode (offline-safe)
- Thumbnail set
- Credits roll variant for authorship reveal

Architectural Guardrails

- FFmpeg runs as a **queue-based worker**, never synchronous
- No user-supplied FFmpeg arguments (ever)
- Outputs stored in object storage; metadata in Route4 DB
- Job states: `pending → processing → complete → failed`
- Hard caps on duration, resolution, and concurrency

🎥 Media Providers (Frame.io Integration)

Route4 does not replace media hosting. Route4 **domesticates** it.

Rule: If a client has to learn hosting tools, Route4 has failed.

Responsibilities

Route4 owns (control plane):

- Release timing and state machine
- Visibility levels (L0–L3)
- Audience gating and invitations
- Canon and archive links
- What appears, when, to whom, and why

Media Provider owns (data plane):

- Storage and streaming
- Playback reliability
- Asset versioning
- Delivery performance

Provider Abstraction

Route4 integrates providers through an internal adapter. Provider names stay invisible to users.

Folder Structure (Frame.io)

```
route4-{clientSlug}/
├── {releaseKey}/
│   ├── 00-raw/          (L0)
│   ├── 10-process/      (L1)
│   ├── 20-fragments/    (L2)
│   ├── 30-release/      (L3)
│   └── 40-archive/      (L3)
```

Asset Naming

`{releaseKey}__{stage}__{kind}__v{n}.{ext}`

Examples:

- `S1E1__process-writing__process_preview__v1.mp4`
- `S1E1__residue-fragments__fragment__v3.mp4`
- `S1E1__drop__release_public__v1.mp4`

👥 Witness as Status

Witness is a role, a verb, and a status.

- **Witnessing** = presence, not performance
- **No leveling, no gamification**
- **Presence is acknowledged quietly**

Witness Unlocks

- Reflection channels
- Invitations
- Private Viewings
- Archive access

🔮 Release State Machine

```
DRAFT
  ↓
SIGNAL_I (Casting Call)
  ↓
PROCESS (Table Read, Writing, Shot Council)
  ↓
HOLD (Compression)
  ↓
SIGNAL_II (Anonymous Song)
  ↓
DROP (Primary Release)
  ↓
ECHO (Reflection)
  ↓
ARCHIVE (Permanent)
```

Enforcement:

- Discord never decides state
- App state drives Discord permissions

- Presence is time-bound
- Missed rituals are not replayed

📋 New Client Bootstrap (Route4)

This checklist defines how Route4 provisions, configures, and governs Discord for a new client.

> The client never needs to "learn Discord."
> Route4 treats Discord as an execution surface, not a product feature.

Phase 1 — Client Intake (Route4 Admin)

- ☐ Client name + short description
- ☐ Primary creator / core team members
- ☐ Preferred language pack (terms like Witness, Signal, etc.)
- ☐ Release cadence (episodic, seasonal, one-off)
- ☐ Desired rituals (enable / disable per client)
- ☐ Confidentiality sensitivity (low / medium / high)

Phase 2 — Discord Provisioning

Route4 handles:

- ☐ Create Discord server (or connect to existing)
- ☐ Apply Route4 default channel templates
- ☐ Create role set: Core Team, Witness, Member
- ☐ Apply read-only defaults to most channels
- ☐ Lock process rooms (closed by default)

Client only: Confirms names and access — no setup required.

Phase 3 — Permission & Safety Defaults

- ☐ @everyone posting disabled in ritual channels
- ☐ Slow mode enabled in reflection channels
- ☐ File uploads disabled in discussion channels
- ☐ Voice/stage rooms locked by role
- ☐ Invite links disabled or time-limited

Phase 4 — Ritual Mapping

For each enabled ritual:

- ☐ Map ritual → channel
- ☐ Define open/close behavior
- ☐ Define visibility level (L0–L3)
- ☐ Define Discord automation (lock/unlock)
- ☐ Define app deep links

Phase 5 — Client Review (15 minutes)

Client reviews and confirms:

- ☐ Channel names (optional rename)
- ☐ Role labels (optional rename)
- ☐ Ritual enablement

No Discord training required.

Phase 6 — Go Live

- ☐ Publish invite link via Route4 app
- ☐ Lock unused channels
- ☐ Set first release schedule
- ☐ Run first ritual dry test

Phase 7 — Ongoing Governance (Route4-Owned)

- ☐ Channel state managed by release cycle
- ☐ Roles assigned via Witness events
- ☐ Ritual rooms open only when needed
- ☐ Archive maintained in Route4 app

Design Principle:

> If a client asks how Discord works, Route4 has already failed.

💰 Care-Based Threshold Pricing

Route4 uses a **Care-based threshold model**.

Creators are **never charged** for intent, planning, or uncertainty.

Charges occur **only when a creator crosses an irreversible threshold** — when Route4 must protect meaning, allocate resources, or create permanence.

Non-Chargeable Rituals

Ritual	Reason
Splash Page	Passive presence only
Signal I — Casting Call	Declaration of intent
Intake Without Noise	Presence registration
The Hold	Silence by design

Chargeable Thresholds (Care Gates)

Gate	Ritual(s)	What Changes	Status

Gate	Ritual(s)	What Changes	Status
Gate 1 — Witness Activation	Table Read, Writing, Shot Council	Protected access + tracking	Yes
Gate 2 — Artifact Generation	Process recording, Fragments, Previews	Permanent artifacts created	Yes
Gate 3 — The Drop	Primary Release	Canon sealed + archive created	Yes
Gate 4 — Private Viewing	Private Viewing	Scarce access issued	Yes
Gate 5 — Distribution (Optional)	Post-Premiere Distribution	External escalation	Yes (Optional)

Reference Fees (Internal Model)

- **Signal Publication:** $25–$50
- **Witness Activation Window:** $50–$100
- **Fragment Batch:** $10–$20
- **Process Preview Set:** $30–$50
- **Release Packaging:** $75–$150
- **The Drop:** $100–$200
- **Private Viewing Event:** $50–$100
- **Distribution Gate Evaluation:** $100–$250

🎯 Why This Model Fits Route4

- Encourages restraint
- Discourages noise
- Aligns money with meaning
- Makes each ritual intentional
- Prevents platform abuse

Route4 never charges for presence — only for crossing.

📝 Platform vs Client Separation

Route4 (Platform) Provides

- Release cycles and state machines
- Ritual templates and automation
- Visibility level enforcement
- Presence tracking (Witness)
- Discord orchestration
- Archive permanence
- Invitation mechanics
- Media pipeline defaults

Route4 Does NOT Provide

- Story or lore
- Character names or outcomes
- Timing decisions
- Credits or authorship revelation
- Language or tone

Clients (e.g., Mary) Define

- Ritual names (Signal, Hold, Witness, etc.)
- Language and tone
- Story canon
- Timing and scheduling
- Credits and revelation
- All creative intent

Configuration, not customization. Route4 supports multiple clients through configuration alone, never forks.

🏗 Engineering Principles

1. **Configuration over customization** — Support multiple clients through configuration, not forks
2. **Discord is infrastructure, not product** — Discord APIs and permissions should remain invisible to clients
3. **State > events > UI** — Release state drives Discord behavior and UI rendering, not the other way around
4. **Minimal surface area** — Every feature must map directly to a ritual or visibility level
5. **No premature abstraction** — Abstractions emerge from Mary + one additional client, not speculation

---

🛠 Recommended Build Order

Phase 1 — Stabilize the Spine

- ☐ ReleaseCycleTemplate model
- ☐ ReleaseInstance state machine
- ☐ Visibility level enforcement
- ☐ WitnessEvent tracking

Phase 2 — Discord Integration

- ☐ Channel template provisioning
- ☐ Role creation and assignment
- ☐ Lock/unlock automation per stage
- ☐ Invite link lifecycle management

Phase 3 — App Surface (Angular)

- ☐ Signal view (anonymous artifact)
- ☐ Release view (episode / primary release)
- ☐ Archive (read-only)
- ☐ Invitations (private viewing)

Phase 4 — Admin Surface (Internal)

- ☐ Create / schedule releases
- ☐ Advance release stages
- ☐ Declare canon
- ☐ Issue invitations

📡 Post-Premiere Distribution Gate (Optional Extension)

This optional gate governs distribution decisions *after* the work has premiered and canon is sealed.

The Principle

> Distribution is not success.
> Distribution is a *response* to demonstrated resonance.

Route4 never optimizes for reach. It evaluates **readiness**.

The Threshold

The Distribution Gate may be considered only after:

- A premiere has occurred (public or private)
- Canon is sealed
- Credits are locked
- Archive is complete

AND when both are met:

1. **Revenue Signal** — Post-premiere revenue threshold reached (tickets, licensing, sponsorship)
2. **Participation Signal** — Demonstrated audience care via witness participation, retention, engagement

Views alone are explicitly insufficient.

Route4 AI-Assisted Distribution Agent

Route4 may provide an **AI-assisted decision agent** that evaluates readiness — not placement.

Possible outputs:

1. Hold (No distribution)
2. Limited Broadcast (YouTube)
3. Ad-Supported Platform (Tubi)
4. Transactional Platform (Prime)

5. Aggregator Submission (Filmhub)
6. Strategic Hold (Await interest)

Netflix or similar platforms are **never auto-selected** and require external inbound interest.

Why This Fits Route4

- Prevents reactive distribution
- Preserves leverage
- Avoids premature platform dilution
- Treats platforms as *outcomes*, not goals

Route4 answers one question only: Is it time to go wider — or not?

Final Engineering Constraint

> If a feature cannot be explained as supporting a ritual, visibility level, or release stage, it does not belong in Route4.

This constraint is **non-negotiable**.

Using AI Assistants (Guardrails)

When using Copilot, Claude, or any AI assistant:

- Treat this README as **source-of-truth**
- Reject suggestions that:
  - Add engagement mechanics
  - Expose story prematurely
  - Gamify participation
  - Collapse platform/client separation

Suggested priming prompt:

> "This document defines the authoritative architecture of Route4. Do not suggest features or designs that violate its constraints."

Final Note on Sustainability

If Route4 cannot support itself without compromising:

- Ritual
- Restraint
- Authorship
- Silence

then Route4 should remain small.

That is acceptable.

Mary is the first client of Route4. All principles above derive from and defer to Mary's creative vision.