Tredy Documentation

Practical guides for putting Tredy's services to work. Source: https://docs.tredy.ai

Tredy Documentation

Explore our guides and put Tredy's services to work — from first run to approved outcome.

Put Tredy's services to work — from first run to approved outcome.

Quickstart →

Browse services

How Tredy works

Tredy runs **services** — units of expert work that Tredy performs end to end and delivers as an approved **outcome**, keeping the accountable human in the loop through exceptions, decisions, and approvals. You're not buying a tool to operate; you're getting the finished work, priced on the outcome.

- **Quickstart** — Run your first service and get a real outcome in minutes.
- **Core concepts** — Services, runs, outcomes, and the knowledge layer — and how they fit.
- **Browse services** — See the services Tredy ships and what each one delivers.
- **API reference** — Trigger runs and read outcomes programmatically.

Choose your path

- **I want an outcome** — Pick a service, give it context, and let Tredy deliver the work.
- **I'm setting up my org** — Connect tools, invite members, and build your knowledge layer.

Quickstart

Run your first Tredy service and get a real outcome.

1. **Open your organization.** Sign in and select (or create) your organization.

Everything in Tredy — services, knowledge, outcomes, billing — is scoped to an org.

2. **Pick a service.** Browse the catalog and choose a service that matches the

work you need done. See Services overview.

3. **Give it context.** Connect the tools and documents the service needs. Tredy

shows the required setup steps before a run can start.

4. **Run it.** Trigger the service manually, on a schedule, or on an event. Tredy

does the work and delivers an **outcome** you can review, approve, and hand off.

> **Note:** A run can pause for you: when a real decision is needed, it surfaces in the outcome's **Needs You** section and the run resumes once you answer.

Core concepts

The handful of ideas that everything in Tredy is built on.

Service

A **service** is a unit of work Tredy performs end to end. Each service is an org-owned, **versioned definition** (with a `slug`, a capability manifest, and a deliverable contract) that produces runs and delivers outcomes.

Run

A **run** is a single execution of a service. Runs move through stages, can be paused, resumed, or cancelled, and can pause for a **human-in-the-loop** decision before continuing.

Outcome

An **outcome** is the delivered work product of a run — a surface to inspect, trust, change, approve, export, and route. Every outcome carries a **Proof Rail** (evidence, source versions, method, cost, and the approval trail) and a **Needs You** section that surfaces the real human decisions. Recurring services **reconcile** their runs so you track one living result over time.

Knowledge layer

Your org's documents, connected tools, and **business profile** form a knowledge layer that services draw on to ground their work.

Organization

Everything is scoped to an **organization** — members and roles, connected tools, knowledge, services, and billing all live at the org level.

Running a service

How to trigger a run, and what happens during one.

Trigger modes

Every service resolves to one of three trigger modes:

- **Manual** — Run on demand from the service page or API.
- **Schedule** — Run on a cadence (e.g. hourly, every N days).
- **Event** — Run when a connected tool fires an event (webhook or connector event).

If nothing is armed, a service defaults to **manual**.

Setup steps

Before a run starts, Tredy shows any **setup steps** it needs — connectors to

link, a reviewer to assign, documents to provide. Resolve them inline, then run.

During a run

A run moves through stages and can be **paused**, **resumed**, or **cancelled**.

When a real decision is required, the run enters a **human-in-the-loop** state:

it surfaces the question, waits for your answer or edit, and continues.

Outcomes

The delivered work product of a run.

An outcome is what you actually get from a run: a reviewable, approvable work product with full provenance.

The Proof Rail

Every outcome carries a **Proof Rail** — an evidence snapshot with source versions, the method and version used, the runtime / model / tool policy, cost, and the full approval trail. Approved publications are recorded as immutable, versioned **gate results**, and sign-offs produce **signed deliverables** with maker–checker signatures and a content hash.

Needs You

Outcomes separate the machine's work from the **true human decisions**. Anything that genuinely needs a person shows up in **Needs You**, so review is about decisions, not digging.

Reconciliation

Recurring services fold their runs into **one living outcome** — updated in place with a version trail — so you see what's new, open, and resolved over time instead of a stack of separate reports.

Connected work

An outcome or its artifacts can be **handed off** to a downstream service, linking work across runs so provenance follows the chain.

Approving & exporting

Review happens on the outcome itself. Depending on the service you can **request review, approve, apply a** signature**, and** export** — and these side-effecting actions run through governed steps, so sensitive links and exports are opened via a controlled action rather than exposed directly. Sign-offs produce signed deliverables with maker–checker signatures and a content hash.

Knowledge layer

The org context that grounds every outcome.

Services ground their work in your org's knowledge layer.

Documents

Uploaded and connected documents are parsed, categorized, and embedded into a per-workspace **vector namespace** that services query for grounding. A redaction / PII pipeline runs over ingested content.

Business profile

Each org has a **business profile** capturing its purpose, industry, best practices, policy backbone, philosophy, geography, activities, and client types. Tredy also maintains a refreshable **company brief** — a generated summary of your org's teams, services, and risks — that services can draw on.

Brand voice

Brand voice is available as a service capability (used by the marketing and creative services) so generated content sounds like your organization.

Connectors

Link the tools your services need.

Connectors let services read from and act on your existing tools. They're managed through Nango and scoped to your org (and optionally a user or workspace).

Available integrations

Airtable · Gmail · Google Drive · LinkedIn · Outlook · Shopify · Slack · Twilio

Link a connector once, and any service can request access to it as part of its setup steps. All connector access is recorded in an audit log.

Authoring a service

Describe the outcome you want in plain English, and Tredy assembles a governed, runnable service from a best-practice playbook.

You don't build a Tredy service by wiring stages by hand. You **describe the outcome**,

and Tredy's Service Designer assembles a governed, runnable definition from a proven

playbook — then you refine it in an editor.

How it works

1. **Describe it in plain English.** Write a one-paragraph description of the outcome you want.
2. **Tredy picks a playbook.** Your description is matched (lexically, then semantically) to a best-practice playbook — competitor intelligence, coverage / policy / security-control / regulatory / contract / vendor gap analysis, document analysis or comparison, regulatory monitoring, denials prevention. The playbook brings a proven tool chain, quality gates, and a deliverable format. Cross-cutting best-practice **skills** are always attached.
3. **The model drafts, code assembles.** An LLM proposes the inputs, agents, and stages; Tredy deterministically expands that into a strict service definition and overlays the playbook's governance. Only **read-only** capabilities are freely selectable — anything that exports or publishes is reachable **only through an approval-gated stage**, never freehand.
4. **Auto-lint, grade, and accept.** The draft is linted, graded by a reviewer model, and put through a deterministic **acceptance gate** (required capabilities, quality gates, review gates, deliverable sections). Tredy self-corrects up to twice, keeping a revision only if it genuinely improves.
5. **Refine in the Service Creator.** The result opens in the editor for you to adjust — nothing is persisted until you save.

Lifecycle & governance

Once you save, a service moves through a **maker–checker** lifecycle, fully audited at

every step:

```

draft → validated → submitted → approved → published → installed

```

Publishing hot-registers the service with zero runtime changes, and every definition

change is recorded with a hash for provenance.

> **Tip:** Because side-effecting steps are always forced behind approval gates, an authored service is safe to run before you fully trust it — it can't publish or export without a human approving.

All services

Every service Tredy ships, what it delivers, and where to start.

A Tredy service delivers a concrete **outcome**, not just an answer. Services come

in two shapes:

- **Runnable workflows** — Tredy executes a multi-stage run and produces media/assets.
- **Assurance surfaces** — a governed review-and-sign-off workpaper for regulated work,

produced by an authoring playbook.

Creative & marketing (runnable)

- **Marketing Arm** — Grounded social campaigns — calendar + real image-to-video packs from a listing's own photos.
- **New Marketing** — A full weekly marketing calendar + per-platform content pack.
- **Creative Production Arm** — Validated creative assets (scripts, scenes, video) built to hand off to another service.

Assurance & risk (review surfaces)

- **Compliance Guard** — A regulated compliance workpaper — questions, evidence, sign-off.
- **Policy Update** — Policy-change redlines with maker–checker sign-off.
- **Regulatory Horizon** — Continuous regulatory monitoring with a signed liaison brief.
- **Digital Footprints** — A triaged feed of your org's public-web exposure.

Marketing Arm

Grounded social campaigns — a content calendar plus real image-to-video packs, built from a source's own photos and held for your approval.

Marketing Arm turns a single source (a listing or brand page) into a grounded social campaign: a content calendar plus real AI **image-to-video "marketing packs"** (MP4 reel + cover + captions + voiceover) built from that source's *own* photos — held for your approval before anything is generated at cost or published.

When to use it

You want a recurring, on-brand social presence for real listings or products, with a human approving every campaign before money is spent or posts go out.

What you need

Setting	Notes
Source URL *(required)*	The listing / photo / brand page to ground everything in
Campaign goal	e.g. promote the source, get bookings, target luxury or families
Cadence & repeat	Plan length in days; run `once` or `repeat every period` (recurring spends real money, so it's opt-in)
Listings, platforms, reel length, voice	1–10 listings; Instagram & Facebook publish via Meta, others are plannable
Cost cap & approval mode	Per-clip USD cap (default $15); review every stage / before publish / draft only

Publishing to Instagram/Facebook needs a connected **Meta** account. Only real source data is used — the publisher and analytics require receipts, so nothing is faked.

How a run works

1. **Plan & scan** — set the cycle's goal/KPIs, read the source, and extract real listing facts + photos.
2. **Prioritize & plan** — pick this cycle's listings, then build the audience, theme, captions, compliance QA, and calendar.
3. **Review** *(you)* — approve the calendar, captions, and platforms before any paid generation.
4. **Produce** — build the storyboard from verified photos and generate the real MP4 reels.

5. **Publish** — post each approved item (or schedule it); items without a receipt are blocked, not faked.
6. **Close & learn** — ingest real analytics and write a cycle report that feeds the next run.

Where you approve

The **weekly review** gate is the main checkpoint — approve the calendar before any cost is incurred. In addition, **each external post requires publish approval**. You can raise this to "review every stage" in the approval mode.

What you get

A **Marketing Campaign Package**: the approved calendar, captions, real reels + covers, publish receipts, and a close-cycle report — with every asset traceable to the source.

New Marketing

A full weekly marketing calendar plus a per-platform content pack, produced end to end with a human review before scheduling.

New Marketing produces a **Weekly Marketing Calendar + Platform Content Pack**:

prioritized listings, per-platform variants, produced media, brand/compliance QA, and

a weekly report — reviewed by a human before anything is scheduled or published.

When to use it

You want a complete weekly marketing operation for multiple listings, delivered as one

reviewable calendar rather than piecemeal posts.

What you need

Setting	Notes
Source URL *(required)*	The listing / brand source to ground the week
Listing count	1–20 (default 3)
Platforms, voice, subtitles	Same platform set as Marketing Arm; subtitle options
Approval mode	Controls how much you review

Optional: **Telegram submission** (review on the go), a **social publisher** connector,

and **analytics** ingestion.

How a run works

1. **Intake & prioritize** — set the weekly goal, scan the listing inventory, and pick the listings to feature.
2. **Context & plan** — market context, audience segmentation, weekly theme, and the content calendar.
3. **Create** — per-platform strategy, creative briefs, variant generation, and produced media (with an FFmpeg fallback).
4. **QA** — brand and compliance checks over everything.
5. **Human review** *(you)* — review the full weekly calendar and individual posts.
6. **Publish & learn** — schedule, monitor, ingest analytics, and write the weekly report.

Where you approve

Four gates: **blocker review**, the **weekly calendar approval** (before scheduling),

per-post approval, and **publish approval** before any external output. Agents are guardrailed — no invented listing facts, no copied captions, and no "published/analytics" claims without connector receipts.

What you get

A weekly calendar + platform content pack. The outcome preview is a **planned-posts carousel** (title, caption, platform, scheduled time, media) with open / request-changes / approve actions on each post.

Creative Production Arm

A validated creative package — scripts, scene plans, generated assets, and cost logs — built to hand off to another service. It never publishes externally.

Creative Production Arm is a **connected "sister service"**: it produces a validated `creative_production_package` (scripts, scene plans, generated assets, validation proofs, storage + provider receipts, cost logs) and hands it to a parent marketing/sales/training service. It does **not** schedule or publish externally itself.

When to use it

When another service (or you) needs polished creative assets produced and validated, but the publishing/approval happens elsewhere.

What you need

Setting	Notes
Source URL *(required)*	What the assets are grounded in
Asset types	short video, cover, static image, carousel, story asset
Target duration	5–120s (default 30)
Budget cap	USD per run (default $25)
Publish channels *(optional)*	Passed through for the downstream service

How a run works

Every stage is **automatic** — this service produces an approval-ready package rather than pausing for you:

1. **Normalize the request** — a typed brief (never inflates duration or budget).
2. **Media intake** — inventory the source media (no invented assets).
3. **Blueprint** — hook, scenes, narration, CTA, and asset mapping.
4. **Provider plan** — budget-aware provider selection and a cost estimate.
5. **Generate & validate** — produce assets with provider + storage receipts, then verify they're playable and within budget.
6. **Package** — assemble the final production package.

Where you approve

Nothing internal — it emits an **approval-ready package** (with `approvalStatus` and review/publish channels) that the **parent service** or downstream owner reviews and publishes.

What you get

A `creative_production_package`: the request, script, generated assets, a validation report, the provider decision log, and a cost report — delivered over Tredy's service **handoff protocol** so a downstream service can pick it up.

Compliance Guard

A regulated compliance workpaper — questions, answers, evidence with integrity status, and governed sign-off.

Compliance Guard is Tredy's **compliance workpaper**: the governed surface where you review a set of compliance questions, inspect their supporting evidence, and take it through sign-off and export.

When to use it

You need to demonstrate compliance against a framework — working through obligations, attaching evidence, and producing a signed, exportable record.

What the workpaper shows

- **Questions** — each with a prompt, an answer, a status, reviewer notes, and attached **evidence**.
- **Evidence integrity** — every evidence item is `verified`, `pending`, or `failed`. Integrity is *never implied* when the source reports pending or failed.
- **Statuses** — `approvalStatus`, `evidenceIntegrity`, `signatureStatus` (unsigned / pending / signed), and `exportStatus`.

How you use it

1. **Find gaps** — surface incomplete or failed questions first.
2. **Inspect evidence** — open each item; evidence links open through a **governed action** (sensitive URLs aren't exposed directly).
3. **Sign off** — request review, apply a signature, and export — all governed steps.

> **Note:** Compliance Guard is produced by an authoring **playbook** (a gap-analysis playbook such as coverage, policy, security-control, regulatory, contract, or vendor). See Authoring a service.

Policy Update

Policy-change redlines with maker–checker sign-off — current vs proposed, with cited findings and a confirmed control mapping.

Policy Update delivers a **redline diff set** for a policy change: the current vs

proposed document, a reconciled control mapping, and cited findings — governed by a

maker–checker lifecycle so nothing is accepted without a second signer.

When to use it

A policy or guideline is changing and you need a defensible record of *what* changed,

how significant each change is, and *who confirmed it*.

What the outcome shows

- **Current vs proposed** documents, each with controls and obligations that carry a **citation** (document, control, quote).
- **Pairing** — how current and proposed map together, with a confidence score and a confirmation state.
- **Reconciliation** — control mappings (matched / added / removed / unresolved) with counts.
- **Findings** — each classified (editorial, scope changed, obligation added/removed, strengthened/weakened), with a severity, a confidence, and whether it **requires human validation**.
- **Provenance** — analysis version and SHA-256 hashes of the sources.

Maker–checker lifecycle

The review moves through **pairing review → finding review → owner review**, and the

gates are enforced:

- Unconfirmed pairings **stop at pairing review**.
- Any finding that requires human validation **stops at finding review**.
- Confirming a pairing records the **actor's id and a timestamp** — that's the checker.
- Every citation is validated against the real control text.

What you get

A signed policy diff set with a redline canvas, the change list, and a decision bar —

review each finding, confirm the pairing, and approve.

Regulatory Horizon

Continuous regulatory monitoring — detect changes across authoritative sources and deliver a signed liaison brief.

Regulatory Horizon continuously monitors regulatory sources, detects what changed against a prior baseline, and delivers a **signed liaison brief** — not just another alert.

When to use it

You need to stay ahead of regulatory change across jurisdictions and hand decision-makers a grounded brief on what moved and what to do about it.

What you need

- **Jurisdictions** *(required)* — what to monitor.
- **Regulatory domain** *(optional)* — narrows the scope.

How a run works

1. **Discover & monitor** — find and fetch authoritative sources.
2. **Extract** — pull the local requirements from each source.
3. **Baseline & detect** — retrieve the prior baseline and detect changes against it.
4. **Draft the brief** — an executive summary, jurisdictions monitored, changes detected, local requirements, and liaison actions.
5. **Verify** *(you sign off)* — a reviewer confirms the brief is grounded.
6. **Distribute** *(gated)* — the signed brief is distributed as evidence of notification.

Where you approve

Two gates: **monitoring sign-off** (at verification) and **distribution sign-off** — distribution is the only side-effecting step and is held until a reviewer approves the signed package.

What you get

A signed **regulatory monitoring brief** (PDF + Excel). Every detected change cites both the prior baseline and the current source, and each run stamps an **as-of** date. If sources are insufficient, it says so explicitly rather than issuing a false all-clear.

Digital Footprints

A triaged feed of your organization's public-web exposure, with what changed since the last approved scan.

Digital Footprints scans the public web for mentions and exposure of your organization and delivers a **triaged feed** — ranked by risk, with the original sources and a clear picture of what changed since the last scan.

When to use it

You want ongoing visibility into what's publicly visible about your org, with the riskiest items surfaced first and a clear action on each.

What the feed shows

- **Findings** — each with body, tone, category, an editorial label, a **severity**, a **primary action** (e.g. removal), the **source** (name/domain), and a link **preview**.
- **Highlights & high-risk count** — the items that need attention first.
- **Delta summary** — what changed compared with the previous approved scan (or "first scan").

How you use it

1. **Triage** — scan by source, category, tone, or action; high-risk items are discoverable without opening every card.
2. **Open the source** — original URLs open via a **governed action** (never fabricated from display text).
3. **Act & approve** — each finding carries a primary action; a reviewer triages and approves the scan.

> **Note:** "What changed" is the **delta vs the previous approved scan** — that's how recurring runs stay one living feed instead of a pile of separate reports.

Organizations

Members, roles, and org-level settings.

Everything in Tredy is scoped to an **organization** (orgs can also be nested with parent/child relationships).

Roles

Role	Typical use
`ORG_SUPER_ADMIN`	Full control of the org
`ORG_ADMIN`	Administer members and settings
`ORG_MANAGER`	Manage services and work
`ORG_VIEW_EDITOR`	View and edit content
`ORG_MEMBER`	Standard member
`ORG_REVIEWER`	Review and sign off on outcomes

Job titles

Members can carry a **job title** (e.g. "CISO") that is separate from their permission role — used for attribution and reviewer assignment.

Billing

How Tredy is priced.

Tredy billing has two parts, both backed by Stripe.

Subscription tier

Your organization is on a **subscription tier**: `free`, `starter`, `team`, `business`, `pro`, `premium`, `enterprise`, `unlimited`, or `custom`. The tier sets your included entitlements and plan-level limits.

Outcome units

Delivered work is metered in **outcome units**. Each period your org receives **outcome grants**, and consumption is tracked in an **outcome ledger** (debits and credits with a running balance). Usage-based costs are recorded in a usage ledger and priced per tier and meter.

> **Note:** Manage your plan and payment method from your organization's billing settings.

API Introduction

Trigger service runs and read outcomes programmatically.

> **Caution:** The Service API is in preview. Endpoints and auth are subject to change, and are not yet covered by the published OpenAPI spec.

Tredy's Service API lets you trigger runs, track their progress, and fetch outcomes from your own systems. All routes are mounted under `/api`.

Services

Method	Path	Purpose
`GET`	`/api/services`	List services available to your org
`GET`	`/api/services/{serviceSlug}`	Get a service definition
`GET`	`/api/services/{serviceSlug}/intake`	Get the service's intake schema

Runs

Method	Path	Purpose
`POST`	`/api/services/{serviceSlug}/runs`	Start a run
`POST`	`/api/services/{serviceSlug}/runs/bulk`	Start multiple runs
`GET`	`/api/services/{serviceSlug}/runs/{runUuid}`	Get a run
`GET`	`/api/services/{serviceSlug}/runs/{runUuid}/status`	Run status
`GET`	`/api/services/{serviceSlug}/runs/{runUuid}/events`	Run event stream
`POST`	`/api/services/{serviceSlug}/runs/{runUuid}/cancel`	Cancel a run
`POST`	`/api/services/{serviceSlug}/runs/{runUuid}/pause`	Pause a run
`POST`	`/api/services/{serviceSlug}/runs/{runUuid}/resume`	Resume a run

Human-in-the-loop

Method	Path	Purpose

`GET`	`/api/services/{serviceSlug}/runs/{runUuid}/human-loop`	Pending decision
`POST`	`/api/services/{serviceSlug}/runs/{runUuid}/human-loop/answer`	Answer it

Outcomes

Method	Path	Purpose
`GET`	`/api/services/{serviceSlug}/outcome-preview`	Preview the outcome
`GET`	`/api/services/{serviceSlug}/artifacts/{artifactId}/export`	Export an artifact

> **Tip:** A separate legacy **Developer API** (documents, workspaces, system) is documented via OpenAPI under `/v1/*`. The Service API above is the current, service-oriented surface.