1. Executive Summary & Mission

Closed-loop creative society: exclusive in culture, open in utility. SSO unifies creation, validation, mastering, and distribution under one self-governing technical standard.

- Operating principle: 'Automation as culture' — standards and math are the gatekeepers.
- Scope: ingest → validate → master → package → distribute → audit → analyze.
- Outcome: studio-grade, Apple/DDEX-compliant releases at indie speed with enterprise reliability.
- Differentiator: a multi-division system that pairs luxury-brand curation with tech-firm rigor.

Table of Contents

Auto-generated from detected section headings. Click via your viewer’s outline/bookmarks.

2. System Overview & Architecture Map

Modular, distributed, self-validating stack with strict metadata parity across ERN, MEAD, ADM, and TTML/ASS.

- Core modules: Digital Solutions (ingestion), Validator (QC & simulation), Mastering Chain, Distribution (VVS), Analytics, Security.
- Internal APIs link modules with versioned contracts; each node is independently deployable.
- Event-driven orchestration coordinates transitions, retries, and back-pressure handling.
- State model: immutable release objects, reproducible deterministic builds, audited transitions.

3. Core Philosophies: Automation as Culture

Every rule is executable; every check is testable; every exception is logged and learnable.

- Replace preference with policy: YAML/JSON rules define acceptance and packaging behaviors.
- Mathematical gates precede human review; humans tune the gates, not the output.
- CI/CD for media: each commit to a release object triggers validation and packaging jobs.
- Telemetry-first approach: logs, metrics, and traces inform continuous improvement.

4. The Creative Society Model

SSO behaves like a sovereign micro-economy: artists, engineers, validators, and distribution, all within one cultural OS.

- Divisions: SSO Digital Solutions, SSO Validator, SSO Artists, VVS Records (sandbox), Admin/Legal.
- Governance: policy registries, role-based permissions, and versioned operating procedures.
- Economic loop: content supply → quality enforcement → distribution → royalties → analytics → reinvestment.
- User roles: creators, operators, auditors, partners — all bound by a single metadata truth.

5. Digital Solutions / Ingestion Hub

Gateway for all assets and metadata; creates canonical packages for downstream modules.

- Metadata normalization: controlled vocabularies, profile selection, UTF-8/16 handling (with/without BOM).
- Schema enforcement: ERN/MEAD XSD validation; optional field coercion; cross-field constraints.
- Asset checks: audio sample rate/bit-depth, channel config, artwork dimensions, TTML/ASS timing sanity.
- Canonical package build: release object + version pinning + manifest IDs (ISRC/UPC parity).

6. DDEX Engine (ERN & MEAD)

Schema-aware generation, validation, and signing of ERN and MEAD XML for large catalogs.

- Profiles & controlled vocabularies enforce DSP compatibility and naming standards.
- Character set and normalization policies ensure cross-platform readability and searchability.
- XSD-driven validation with custom rule layers for partner-specific quirks.
- Diff-based resubmission: minimal change sets regenerate only impacted payloads.

7. SSO ADM Validator (Dolby/ADM/Apple QC)

YAML-driven checks for Dolby Atmos/ADM, Apple Digital Masters, ISRC/UPC parity, and asset readiness.

- ADM/BWF integrity: frame alignment, bed/object mapping sanity, true-peak and LUFS ranges.
- Apple Digital Masters readiness: noise floor, headroom, inter-sample peak detection.
- Metadata parity: embedded tags vs. manifests (ERN/MEAD); mismatch diff reports.
- Blocking thresholds vs. advisory thresholds: deterministic outcomes and operator transparency.

8. Granular Lyric Sync Engine

Musixmatch HTML → .ass → Apple TTML → MEAD; supports background-vocal tagging and word-level highlight.

- Inline parentheticals separated into bgv lines; timing normalized; whitespace standardized.
- ASS → TTML conversion preserves word-level highlighting and stanza structure.
- Apple compliance checks: timestamp bounds, no negative or overlapping intervals, line-length policy.
- MEAD mapping: text payload, language codes, role tags (lead/bgvs), territorial considerations.

9. Encoding & Localization Framework

UTF-8/16 normalization (with/without BOM), BCP-47 language tags, and region-aware metadata formatting.

- Language/script normalization for names, credits, and territory rights.
- Canonical casing and punctuation policies for artist names, featuring credits, and remixes.
- Right-to-left script handling and fallback glyph policies.
- Release title templating ensures cross-market readability and policy compliance.

10. QC Orchestrator (Pre-flight Automation)

Pre-flight validation across audio, art, metadata, and XML; enforces DDEX profiles and Apple Transporter checks.

- Job graph executes conditional checks: fail-fast for structural errors, soft-fail for advisory items.
- Image/audio spec validation (dimensions, DPI, color space; sample rate, bit depth).
- Rights completeness: contributors, publishers, splits, and mechanical licensing indicators.
- Operator console provides actionable diff, remediation guidance, and re-run controls.

11. Validator Architecture: Monte Carlo & QMC

Statistical immune system that predicts failure modes via Monte Carlo and quasi-Monte Carlo sampling.

- Parameter sampling over metadata drift, timing jitter, and partner-specific tolerances.
- Sobol/Halton sequences for deterministic coverage of edge cases; reproducible randomness.
- Outcome metrics: predicted reject probability, severity score, and root-cause candidates.
- Threshold learning from historical outcomes refines gating policies over time.

12. Simulation: Error Propagation & Bayesian Anomalies

Combines propagation modeling with Bayesian detectors to surface low-frequency, high-impact issues.

- Posterior updating on ingestion errors informs future priors for similar payloads.
- Time-series modeling on stream/royalty anomalies; alerts for outliers beyond credible intervals.
- Attribution heuristics localize anomalies to field-level or asset-level sources.
- Operator actions feed back into the learning loop to minimize false positives.

13. Mastering & Audio Chain

Stereo + Dolby Atmos (7.1.6/object) + binaural derivations with Apple Digital Masters compliance.

- Pre-master QC: DC offset, phase correlation, noise-floor profiling.
- Loudness normalization targets set per DSP with inter-sample peak checks.
- Version locking: immutable master artifacts per release; audit-traceable exports.
- Preview snippet generation with fades and watermark support for pre-release ops.

14. Metadata Embedding & Parity

Embed ISRC, UPC, and rights tags within ADM/BWF; ensure parity with ERN/MEAD manifests.

- Two-way diffing between embedded metadata and canonical manifests prevents drift.
- Checksum and signature policies catch truncation or unintended edits.
- Automated remediation prompts for single-source-of-truth maintenance.
- Export gates block deliveries with unresolved parity mismatches.

15. iTMPS / .itmsp Package Maker

Build Apple-ready .itmsp bundles with metadata, assets, TTML, motion covers, and iBooklets.

- Transporter pre-checks run locally to minimize remote rejects and roundtrips.
- Packaging templates enforce folder structure, naming, and aux-asset placement.
- Partial re-uploads supported via manifest diffs for speed and cost control.
- Delivery summaries provide itemized validation results for audit.

16. Spotify Delivery Packager

Create Spotify-compatible deliveries from the same canonical metadata; auto-map fields and territory rights.

- ISRC/UPC parity enforcement; contributor role mapping and localizations.
- Artwork and audio spec enforcement per Spotify profile.
- Regional availability matrices generated from a single rights graph.
- Error classification tailored to Spotify’s ingestion feedback patterns.

17. Amazon/YouTube/TIDAL Multi-Export

One source of truth → multiple packages: Amazon Music, YouTube Content ID feeds, TIDAL, etc.

- Partner-specific XML/CSV/JSON templates maintained with semantic versioning.
- Content ID pre-flags and reference policies integrated into packaging rules.
- Territorial rights and windowing schedules transformed into per-partner payloads.
- Q/A sampling of delivered assets verifies cross-partner consistency.

18. Motion Art & iBooklet Integration

Support animated covers/motion art and PDF iBooklets within packaging and partner constraints.

- Dimension/codec policies for motion assets; fallback static artwork logic.
- iBooklet embedding with TOC and rights notes; localized insert variants.
- Cross-check motion/iBooklet references in MEAD to avoid orphaned assets.
- Preview/thumbnail derivations for portals and storefronts.

19. Encoding & Localization (I18N)

Region-aware name formatting, language codes, and script normalization for global releases.

- BCP-47 tagging across lyrics, titles, and contributor names with script subtags where needed.
- Diacritics and transliteration policies for discoverability and legal names.
- Territorial exceptions (e.g., JP/KR) handled via policy packs.
- Automated QA checks ensure no garbled text or mojibake in downstream UIs.

20. QC Orchestrator (Governance)

Validation governance that unifies distributed checks under a single decision engine.

- Graph reducer composes pass/fail states from sub-checks into a final verdict.
- Explainability: every fail includes machine-readable reasons and human-readable guidance.
- Retry semantics and backoff policies reduce noisy flapping on transient errors.
- Operator overrides require justification and produce audit events.

21. Catalog Intelligence (Analytics)

SQL-backed dashboards unify multi-distributor data for streams, royalties, and asset health.

- Monte Carlo and Bayesian forecasts guide release timing and promotional windows.
- Anomaly alerts on sudden deltas in streams/royalties by territory or DSP.
- Cohort analyses by release type, genre, feature, and campaign tag.
- APIs expose aggregate insights to product and finance stakeholders.

22. Automation Pipelines (CI/CD)

Media-native CI/CD with event-driven jobs for ingestion, QC, packaging, and delivery.

- Idempotent job design allows safe replays; deterministic builds ensure reproducibility.
- Secrets management and per-environment configuration via encrypted stores.
- Canary deliveries validate novel policies with minimal blast radius.
- SLA/SLO definitions for pipeline stages and alerts on error budgets.

23. Webhooks & Private API

Event-based hooks (ingestion, QC, delivery, royalty updates) and private endpoints for bulk ops.

- Webhook signing and nonce checks prevent replay or spoofing.
- Bulk endpoints for package builds and backfill repairs with pagination and resume tokens.
- Fine-grained scopes on API tokens; per-tenant rate limits and quotas.
- Operator CLI bridges human workflows with the same API contract.

24. Security & Identity

Layered access control with least privilege, audit trails, and environment isolation.

- RBAC across creators, validators, admins; time-bounded elevation for critical fixes.
- Artifact signing and checksum verification at rest and in transit.
- Encrypted storage for rights/royalty data; PII minimization by design.
- Threat modeling covers supply chain, partner endpoints, and insider risks.

25. Error Management & Exception Handling

Classified errors (structural, parity, transport, partner) with consistent remediation patterns.

- Error catalogs map codes to steps; auto-suggested fixes lower MTTR.
- Dead-letter queues capture irreconcilable payloads with context for manual repair.
- Runbook automation opens templated incidents for repeatable actions.
- Continuous learning loop prunes noisy alerts and tunes thresholds.

26. Redundancy & Backup Policies

State replication and cold storage strategies to ensure resilience and recovery.

- Versioned artifact stores with geographic redundancy and integrity checks.
- Periodic drills validate restoration time and completeness.
- Policy for tombstoning vs. archiving superseded assets.
- Disaster recovery runbooks with contact trees and RPO/RTO targets.

27. Partnerships: Overview

SSO’s partnerships span standards bodies, rights organizations, and platform ecosystems; each integration maps to concrete technical obligations.

- Badges reflect compliance checkpoints, not marketing claims; each partner implies a specific test suite.
- Partnerships are encoded as policy packs within the Validator to guarantee ongoing conformity.
- Apple Partner, DDEX Member, IFPI, A2IM, NMPA, BMI/ASCAP, ISRC, TrustedSite/DMCA.
- This section details technical implications for each badge in practice.

28. Apple Partner Program

Operates within Apple’s complete media infrastructure: Music, iTunes, TV, Podcasts, and Books.

- iTMPS packaging via Transporter-prechecked .itmsp bundles with TTML lyrics and iBooklets.
- Apple Digital Masters compliance integrated into ADM Validator thresholds.
- Metadata field parity enforced against Apple-specific vocab and Transporter feedback loops.
- Delivery SLAs, error-class mappings, and automated re-try strategies modeled after Apple responses.

29. DDEX Membership & Standards

Active DDEX Member and committee participant, shaping global digital distribution standards.

- Internal XSD validators pinned per DDEX profile/version; diff alerts when schemas update.
- MEAD v1.x compliance for lyric metadata, contributor roles, and art/motion descriptors.
- ERN rights structures mapped to SSO’s canonical rights graph; partner adapters inherit this truth.
- XML signature/namespace policies ensure cross-vendor parsing stability.

30. A2IM Membership

Alignment with independent label innovation and fairness across DSP networks.

- Policy feeds influence Validator’s fairness and metadata integrity checks.
- Advocacy-driven updates prioritized in quarterly policy pack refreshes.
- Benchmarking against independent peers informs pipeline SLAs and analytics KPIs.
- Optional interoperability tests for indie-focused distributor endpoints.

31. IFPI Registration

Operates in alignment with IFPI for compliant catalog registration and industry standards.

- Release registration workflows include IFPI registry checks where applicable.
- ISRC issuance/verification integrated into ingestion; duplicates and gaps flagged.
- Mapping of recording rights to IFPI-aligned data fields enhances global portability.
- Audit reports evidence compliance for partners and legal stakeholders.

32. NMPA Affiliation

Upholds publishing integrity: songwriter rights, fair compensation, transparent licensing.

- Linkage between ISWC and ISRC improves royalty routing and conflict resolution.
- Publishing splits validated against MEAD and PRO records; anomalies escalated.
- Licensing notes captured in ERN extensions; territorial caveats enforced at packaging.
- Monthly reconciliation jobs check payout deltas against expected rights topology.

33. BMI & ASCAP Integration

Affiliate of both PROs to ensure every composition and performance is represented across global networks.

- PRO API synchronization keeps repertoires current; webhook updates feed Catalog Intelligence.
- Cue-sheet ingestion and work registration records attached to releases for auditability.
- Edge cases (co-writes, pseudonyms) standardized via contributor alias registry.
- Automated discrepancy reports highlight mismatched work codes or payees.

34. ISRC & Metadata Partnership

Every release carries registered ISRC for accurate tracking, royalties, and ownership verification.

- ISRC assignment policies enforce uniqueness and consistent encoding across assets.
- Cross-checks ensure no orphaned tracks without valid codes enter packaging.
- Checksum-based lineage links masters to their ISRCs across versions and remasters.
- Partner-specific validations ensure ISRC is consumed and echoed back correctly.

35. TrustedSite & DMCA

Security trust badge and DMCA protection integrated into legal and operational posture.

- Site scanning and certificate hygiene policies documented within Security & Identity.
- DMCA takedown workflows integrated into Webhooks API for rapid enforcement.
- Evidence packs auto-assembled (hashes, timestamps, manifests) for claims.
- Incident logs align to legal retention requirements and privacy constraints.

36. Data Flow Diagrams (Text Schematics)

Schematic representation of high-level flows for quick operator reference.

- Creator → Ingestion Hub → Validator → Mastering → QC Orchestrator → Packaging → Delivery.
- Lyrics: Musixmatch HTML → ASS (bgv split) → TTML → MEAD → DSP.
- Analytics loop: Delivery → Streams/Royalties → Catalog Intelligence → Policy tuning.
- Security wrap: AuthZ → Audit → Secret Mgmt → Artifact Signing.

37. YAML / JSON / XML Samples

Representative snippets to illustrate policy and payload structures.

- Validator policy (YAML): thresholds for LUFS, peaks, timing overlap, name formatting rules.
- ERN excerpt (XML): Release, ResourceList, Deal structures with namespaces.
- MEAD excerpt (XML): Lyrics, roles, and artwork descriptors.
- TTML sample: word-level spans with non-overlapping intervals and bgv lanes.

```
# Validator policy (YAML) example
loudness:
  target_lufs: -14
  max_true_peak_dbfs: -1.0
lyrics:
  allow_overlap: false
  max_line_chars: 120
  bgv_effect_tag: "bgv"
```

38. Metadata Schema Crosswalk

Field-by-field crosswalk linking ERN ↔ MEAD ↔ ADM/BWF ↔ TTML/ASS.

- Contributor roles and identifiers (IPI, ISNI, PRO IDs) mapped across schemas.
- Rights/territory representation normalized to SSO canonical model.
- Art/motion asset references aligned to file manifests and checksums.
- Validation matrix enumerates required/optional, max lengths, and regex policies.

39. Glossary of Protocols & Flags

Quick reference for acronyms, internal flags, and partner-specific terms.

- ADM, BWF, DDP, DDEX, ERN, MEAD, TTML, ISRC, ISWC, UPC, LUFS, QMC, MCMC.
- Internal flags: PARITY_FAIL, TIMING_DRIFT, RIGHTS_MISSING, TRANSPORTER_WARN.
- Policy packs: APPLE_V2025_01, SPOTIFY_V2025_02, YT_CID_V2025_01, TIDAL_V2025_01.
- Severity levels and suggested operator actions keyed to each flag.

40. Revision Control & Version History

Living document with semantic versioning; each change references policy updates and schema bumps.

- v1.3: platform expansions, GUI pages, API reference, extensive code samples, justified footer.
- v1.2: platform starts, justified footer paragraphs.
- v1.1: fully justified layout; typography and margin improvements.
- v1.0: initial 40-page release with validation & partnerships sections.

Spotify: Integration & Policy

Canonical metadata transforms into Spotify-compatible payloads with strict spec enforcement and error taxonomy alignment.

- Field mapping: contributors, roles, territories, explicitness, ISRC/UPC parity, display names vs legal names.
- Asset constraints: audio format, loudness, artwork size/ratio, canvas constraints.
- Error taxonomy mapped to SSO classes; conditional retry with manifest diffs.
- Playlist/algorithmic considerations encoded as non-functional metadata (advisory).

```
// Node.js: build Spotify package (pseudo-SDK)
const pkg = buildSpotifyPackage(release);
await transporter.upload(pkg);
```

Spotify: Delivery Spec & QA

Packaging, Transport, and storefront verification with parity checks.

- Transport via partner API or distributor bridge; verify ingestion receipts.
- Parity checks: titles, credits, timings, locales; sample pulls after go-live.
- Retry matrix for 4xx/5xx; backoff strategies and idempotent resubmits.
- Audit trail binds SSO release_id ↔ Spotify track/album IDs.

```
# Bash: verify ingestion receipts
sso cli spotify verify --release sso_2025_000341 --expect 12-tracks
```

Spotify: Analytics & Royalty

Usage ingestion → reconciliation → anomaly detection.

- Resolve resource IDs to ISRC; dedupe mis-attributed streams.
- Forecast trajectories using Bayesian updates; cohort by playlist exposure.
- Anomaly detection flags inorganic spikes; operator review with evidence.
- Royalty splits reconciled to MEAD/ERN rights graph.

```
-- SQL: find unexpected stream spikes
SELECT * FROM stream_spikes WHERE zscore > 4 AND platform='spotify';
```

TIDAL: High-Fidelity Delivery

Focus on high-res audio and detailed credits.

- Lossless/Atmos paths validated; LUFS/true-peak thresholds enforced.
- Role granularity: mixer, mastering, engineer, producer with identifiers.
- Locale-aware string rules; artwork motion policies.
- Post-delivery parity sampling and storefront QA.

```
tidal:
  audio:
    lufs_target: -14
    peak_dbfs: -1.0
  credits:
    required_roles: [PrimaryArtist, Mixer, MasteringEngineer, Producer]
```

TIDAL: Error Handling & Retries

Map TIDAL ingestion outcomes to SSO error classes.

- Structural vs advisory error split; escalate structural failures.
- Automatic manifest diffs for partial resubmissions.
- Transport retries gated by exponential backoff and jitter.
- Operator console shows human-readable remediation guidance.

```
# Pseudo-CLI
sso retry tidal --release sso_2025_000512 --only partial-failures
```

TIDAL: Analytics & Editorial

Analytics alignment with editorial signals and territory coverage.

- Cohort streaming by editorial placement, format type, and release age.
- Gap analysis for territories with under-indexed performance.
- Alerting for sudden parity drift between credits and storefront data.
- Feedback loop to packaging heuristics for future releases.

```
-- SQL: editorial lift estimate
SELECT release_id, (post_playlist - pre_playlist) AS lift FROM tidal_editorial_lift;
```

YouTube/CID: Reference & Policy

CID policies: monetize/track/block with time windows and territories.

- Generate clean references; tune match thresholds to minimize false positives.
- Policy graph supports derivatives/remixes; connect ISWC for publisher verification.
- Dispute/appeal workflows push evidence packets via webhooks.
- Auto-attach cue sheets and composer info to CID references.

```
curl -X POST https://api.sso/v1/youtube/cid -F package=@cid_00041.zip -H "Authorization: Bearer $
```

YouTube/CID: Music Delivery & Art Tracks

YouTube Music deliveries with parity to canonical metadata.

- Art track generation with consistent naming and credits.
- Lyric sync via TTML; territory windows respected.
- Content ID alignment across UGC and official channels.
- Storefront sampling to confirm parity and availability.

```
# JSON manifest excerpt
{"release_id":"sso_00041","tracks":[{"isrc":"US-SSO-25-141","title":"Midnight"}]}
```

YouTube/CID: Analytics & Rights

Usage analytics join with payout statements; detect CID mismatches.

- Match UGC claims to releases; identify policy conflicts.
- Estimate revenue impact of policy changes via simulations.
- Escalate disputed claims with evidence and timestamps.
- Alert on channel impersonation or mirrored uploads.

```
-- SQL
SELECT claim_id, confidence FROM cid_matches WHERE confidence < 0.65;
```

Amazon Music: Packaging & Availability

Windowing, prime/unlimited/free flags, and Alexa voice-surface readiness.

- Rights graph encodes availability tiers; storefront variants auto-generated.
- Artwork renditions for Echo/FireOS constraints.
- ADX feedback mapped to SSO error classes; partial rebuilds.
- Normalize titles for TTS searchability ('feat.' variants collapsed).

```
# YAML availability
availability:
  prime: true
  unlimited: true
  free: false
  window_end: 2026-01-31
```

Amazon Music: QA & Parity

Sampling and automated checks after ingestion.

- Title/credit parity checks; region availability matrix verification.
- Reconciliation of ADX IDs ↔ SSO release identifiers.
- Alerting for delayed storefront propagation.
- Auto-open incidents on repeated ingestion classes.

```
sso amazon verify --release sso_2025_000722 --regions US,CA,GB,DE
```

Amazon Music: Analytics & Voice

Voice-surface metrics and discoverability monitors.

- Track Alexa query success rates; detect mispronunciations.
- A/B test title variants for search fit while preserving canonical form.
- Join usage reports to royalties and detect anomalies.
- Recommendations for optimization shipped to packaging policies.

```
-- SQL
SELECT title, alexa_success_rate FROM voice_search WHERE rate < 0.90;
```

Deezer: Integration & Profile

Deezer payload mapping and editorial considerations.

- Contributor role adaptations where Deezer schemas differ.
- Image spec enforcement and locale title norms.
- Editorial tagging hints embedded as advisory metadata.
- Ingestion receipts archived for audit.

```
# JSON mapping
{"displayArtist":"Jayden Reo","roles":["PrimaryArtist"]}
```

Deezer: QA & Errors

Handle schema drift and error mapping.

- Detect schema version updates; alert and pin validators.
- Retry logic on transient upstream issues.
- Parity checks post-publish with storefront sampling.
- Consolidation of partner IDs ↔ canonical identifiers.

```
sso deezer sync --release sso_2025_000888 --assert-parity
```

Deezer: Analytics & Growth

Performance cohorts and territory growth plans.

- Analyze editorial impacts; correlate with campaign tags.
- Under-indexed territory detection and playbook suggestions.
- Forecast conversions from exposure to streams.
- Royalty reconciliation against splits.

```
-- SQL
SELECT territory, streams FROM deezer_daily WHERE streams > 50000;
```

Pandora: AMP & Radio

Pandora AMP integration and radio station mapping.

- Map mood/genre tags for radio seed fitness.
- AMP campaign configuration captured in metadata advisory fields.
- QA samples confirm audio and title parity.
- Royalties linked via ISRC and internal track IDs.

```
# AMP payload (pseudo)
{"campaign":"Launch Boost","budget":250,"targets":["US"]}
```

Pandora: Error Taxonomy & Retries

Translate Pandora feedback to SSO error classes.

- Structural errors escalate; advisory warnings logged for tuning.
- Partial resubmissions limited to affected tracks.
- Operator controls for manual overrides with justification.
- Backoff policies prevent hammering fragile endpoints.

```
sso pandora retry --release sso_2025_000931 --only structural
```

Pandora: Analytics

Radio spin analytics and conversion to follows/streams.

- Track station adds/removals; correlate with content changes.
- Detect dropoffs and recommend remedial actions.
- Cross-platform halo effect estimation.
- Royalty reconciliation and anomaly detection.

```
-- SQL
SELECT station_id, adds, removes FROM pandora_stations WHERE date BETWEEN :d0 AND :d1;
```

SoundCloud: Direct Upload & Monetization

Mapping metadata to SoundCloud; monetization flags and territories.

- Auto-populate credits and publishing notes.
- Fingerprinting linkage with YouTube CID to avoid conflicts.
- Artwork/track type/spec checks.
- Claim disputes logged via webhooks.

```
curl -X POST https://api.sso/v1/soundcloud/upload -F file=@track.wav
```

SoundCloud: Policies & QA

Public/private policies; preview and snippet controls.

- Snippet windows for pre-release; private link governance.
- Parity checks after publish; canonical title formats enforced.
- Version pinning for updates with diff-only uploads.
- Audit trail ensures reversibility.

```yaml
# YAML
soundcloud:
  privacy: private
  preview: 60s
  monetization: true
```

SoundCloud: Analytics

Join SoundCloud stats with cross-platform analytics.

- Detect follow/like ratios that underperform baseline.
- Flag suspicious activity and bot patterns.
- Correlate reposts with stream spikes.
- Map payouts to splits and resolve anomalies.

```
-- SQL
SELECT reposts, streams FROM sc_daily WHERE reposts/streams > 0.25;
```

TikTok/SoundOn: Delivery & Policies

Short-form delivery with clip windows and sound ownership policy.

- Clip timing windows enforced; meme-safe loops auto-generated.
- Ownership policy configured to allow/restrict derivatives.
- Link back to full track on streaming services.
- QA manual review for viral-readiness.

```
# JSON
{"clip_start_ms":15000,"clip_duration_ms":30000,"loop":true}
```

TikTok/SoundOn: Rights & Conflicts

CID conflicts with YouTube/SoundCloud handled via fingerprints.

- Shared fingerprint registry prevents double-claims.
- Publisher verification via ISWC links.
- Appeals workflow standardized; evidence packs auto-assembled.
- Monitoring for impersonation sounds.

```
sso tiktok resolve-conflicts --release sso_2025_001112
```

TikTok/SoundOn: Analytics

Viral signal detection and cross-platform uplift forecasts.

- Early-growth detectors trigger promotion suggestions.
- Cross-platform correlation to Spotify saves/streams.
- Creator-sound pairing recommendations.
- Anomaly alerts for stolen/duplicated audio.

```
-- SQL
SELECT sound_id, viral_score FROM tiktok_signals WHERE viral_score > 0.95;
```

Instagram/Facebook Music: Delivery & Stories

IG/FB music sticker enablement with lyric sync and territory windows.

- Ensure TTML alignment for sticker displays.
- Territory rights mirrored from canonical graph.
- Policy for UGC re-uploads documented.
- Sampling verifies display across app versions.

```
curl -X POST https://api.sso/v1/meta/music -F package=@meta_music_00031.zip
```

Instagram/Facebook Music: Rights & PRO

Publisher alignment via ISWC ↔ PRO records.

- Resolve name variants; aliases registry maintained.
- Escalate conflicts to catalog ops; freeze deliveries when required.
- Auto-sync updates when PRO repertoires change.
- Evidence stored for audit trail.

```yaml
# YAML
meta_rights:
  iswc_required: true
  pro_sync: [BMI, ASCAP]
```

Instagram/Facebook Music: Analytics

Stories usage signals linked to streaming outcomes.

- Measure saves to streaming app clicks.
- Correlate with release timing and editorial placements.
- Territory differentials inform future campaigns.
- Anomalies produce corrective tasks.

```
-- SQL
SELECT stories_uses, streaming_clicks FROM meta_usage WHERE date >= :d0;
```

Shazam: Fingerprint & Registration

Register fingerprints for recognition and linkouts to platforms.

- Maintain high-quality references; avoid noisy masters.
- Territory linkout configuration mirrors availability.
- QA check ensures correct artwork/title match.
- Analytics collected for recognition frequency.

```
curl -X POST https://api.sso/v1/shazam/register -F ref=@fp.wav
```

Shazam: Analytics & Lift

Estimate streaming lift from recognition spikes.

- Cohort by territory/time of day; correlate to playlist adds.
- Alert when recognition spikes lack streaming conversion.
- Recommend creative clip changes to improve recognition.
- Feed insights back to release strategy.

```
-- SQL
SELECT territory, recognitions, streams FROM shazam_daily;
```

Shazam: Conflict Handling

Handle duplicate fingerprints and misattributions.

- Detect duplicate ref collisions; decide canonical mapping.
- Appeal mismatched links with evidence.
- Monitor rogue fingerprints added by third parties.
- Retire outdated references properly.

```
sso shazam dedupe --release sso_2025_001341
```

Audiomack: Delivery & Monetization

Deliver releases with monetization flags and territory scopes.

- Artwork/audio spec enforcement; track status governance.
- Early access windows and premiere controls.
- QC parity checks after ingestion; storefront sampling.
- Analytics join to cross-platform dashboard.

```
# JSON
{"premiere":true,"premiere_end":"2025-12-01"}
```

Audiomack: Community & Editorial

Leverage editorial and community reposts with policy safeguards.

- Tag campaign IDs for attribution.
- Prevent metadata drift with version control.
- Validate repost authenticity; bot pattern detection.
- Automate incident creation for fraud signals.

```
sso audiomack verify --release sso_2025_001455 --editorial
```

Audiomack: Analytics

Engagement ratios and conversion funnels.

- Track save-to-stream ratios and completion percentages.
- Detect unusual bounce patterns.
- Forecast growth under campaign schedules.
- Payout reconciliation to splits.

```
-- SQL
SELECT saves/streams AS save_ratio FROM am_daily WHERE date BETWEEN :d0 AND :d1;
```

Boomplay: Delivery & Localization

Territory-specific localization and handset considerations.

- String norms for local scripts; BCP-47 tagging.
- Low-bandwidth renditions where needed.
- Rights matrices adapted to regional licensing.
- QA with storefront sampling.

```yaml
# YAML
localization:
  primary_lang: en
  secondary_langs: [yo, ha]
```

Boomplay: Analytics

Territory growth and device profile insights.

- Measure device splits; optimize artwork and previews.
- Campaign ROI by region.
- Detect reporting gaps; open incidents on missing days.
- Royalty reconciliation pipeline.

```
-- SQL
SELECT device_type, streams FROM boomplay_usage;
```

Boomplay: Ops & Errors

Endpoint stability and retry policy.

- Backoff with jitter; cap retries under SLO.
- Schema drift detection and validator pinning.
- Operator override gates with audit.
- Consolidate partner IDs to canonical graph.

```
sso boomplay retry --release sso_2025_001512 --max-retries 5
```

Anghami: Delivery & Arabic Script

Right-to-left (RTL) handling and metadata integrity.

- Normalize Arabic script; maintain canonical English transliterations.
- Artwork text readability for RTL contexts.
- Credits rendered properly in storefront preview.
- Availability mapped to territories.

```
# JSON
{"title_ar":"نداء منتصف الليل","title_en":"Midnight Call"}
```

Anghami: Analytics

Track engagement and growth in MENA region.

- Cohort by language preference; detect subtitle needs.
- Editorial boosts vs campaign tags.
- Royalty alignment with splits.
- Anomaly detection for sudden drop-offs.

```
-- SQL
SELECT lang_pref, streams FROM anghami_daily;
```

Anghami: Ops

Error handling and incident playbooks.

- Schema validation gates; alert on changes.
- Partial resubmissions and idempotency keys.
- Operator runbooks for common failures.
- Quarterly validator reviews for MENA partners.

```
sso anghami incidents open --template ANGHAMI_SCHEMA_DRIFT
```

Creator Portal: Release Wizard

Stepwise upload and validation with instant feedback loops.

- Stages: Assets → Metadata → Lyrics → Artwork → QC → Package.
- Inline ERN/MEAD hints; auto-fill from templates.
- Real-time checks: length, timing, casing; advisory warnings.
- ‘Ready to Package’ gate only after Validator pass.

```
# Pseudo-UI config (JSON)
{"steps":["assets","metadata","lyrics","artwork","qc","package"],"liveValidation":true}
```

Lyrics Sync Editor

Word-level sync with background-vocal lanes and Apple TTML export.

- Import Musixmatch HTML or ASS; auto-split parentheticals to bgv lanes.
- Mouse/keyboard timing adjustments; snap-to-grid.
- Export to Apple TTML with non-overlapping spans.
- Preview playback with dynamic highlighting.

```
# TTML example (excerpt)
<p begin="00:00:10.00" end="00:00:12.00">Hello <span begin="00:00:11.00" end="00:00:11.50">world<
```

QC Dashboard

Operator console for pass/fail states, diffs, and remediation guidance.

- Filter by error class: structural vs advisory.
- Drill-down to field-level mismatches.
- Retry buttons open with context preserved.
- All actions logged to audit trail.

```
sso ui qc --filter structural --since 7d
```

Packaging Queue Monitor

Observe package builds and deliveries in real time.

- Per-partner pipelines and SLA timers.
- Idempotent rebuilds for partial failures.
- Transport receipts and storefront checks.
- CSV export for ops reviews.

```
# CLI
sso ui queue --partner spotify --watch
```

Partner Delivery Monitor

Cross-partner status with parity sampling and alerts.

- Track IDs across platforms; detect drift.
- Alert rules for missing territories.
- Sampling cadence configurable per partner.
- CSV/PDF snapshot exports.

```
# JSON
{"partners":["spotify","apple","tidal"],"sampleRate":"hourly"}
```

Royalty Analytics Dashboard

Revenue insights with anomaly detection and cohort analysis.

- Join statements to ISRC/UPC graph.
- Detect outliers via z-scores and Bayesian priors.
- Growth projections for campaigns.
- Export plots for EPKs and decks.

```
-- SQL
SELECT isrc, revenue_usd FROM royalty_daily ORDER BY revenue_usd DESC LIMIT 50;
```

SSO Artists Entertainment | Apple Digital Masters | Spotify | TIDAL | Amazon Music | YouTube | Deezer | Pandora | SoundCloud | TikTok | Instagram

Discrepancy Resolver

Human-in-the-loop workflow for resolving ID mismatches and royalty conflicts.

- Suggest canonical merges; capture rationale.
- Re-run analytics after resolution.
- Lock writes during conflict states.
- Notify stakeholders of changes.

```
sso ui resolve --resource track:US-SSO-25-00001
```

Access Control Admin

RBAC editor with audit logging and time-bound elevation.

- Assign roles by division: Creator, Validator, Admin.
- Temporary elevation with reason and expiry.
- View audit logs and export CSV.
- SAML/OAuth federation.

```
# YAML
rbac:
  roles: [creator, validator, admin]
  elevation: {maxHours: 4}
```

API: Authentication & Tokens

OAuth2/SAML federation; token scopes & rotation policies.

- Scopes: releases:write, assets:read, delivery:manage, admin:*
- Short-lived access tokens; refresh via secure channel.
- JWT with audience/issuer checks; clock skew tolerance.
- HMAC for webhook verification.

```
POST /v1/auth/token
Authorization: Basic <client_id:secret>
grant_type=client_credentials
scope=releases:write assets:read
```

API: Releases

Create/update/read releases with immutable artifacts after lock.

- POST creates draft; PATCH allowed pre-lock; GET returns state.
- Lock endpoint seals artifacts; versioning increments on edits.
- Search with filters by artist, date, status.
- Webhooks on state changes.

```
POST /v1/releases
{ "title":"Midnight", "upc":"123...", "artist":"Jayden Reo" }
```

API: Assets

Upload audio/art/lyric files; checksum and media specs enforced.

- Chunked uploads with resume tokens.
- Checksums verified; reject on mismatch.
- Auto-extract technical metadata.
- Link to releases by manifest IDs.

```
PUT /v1/assets/{id}/chunks
Content-Range: bytes 0-1048575/5242880
```

API: Lyrics

TTML/ASS endpoints with validation and conversion.

- Upload TTML; server validates spans and overlaps.
- Convert ASS → TTML with bgv lanes preserved.
- MEAD lyric mapping helpers.
- Preview render endpoint for QA.

```
POST /v1/lyrics/convert
{ "input":"ass", "effects":["bgv-split","normalize-spaces"] }
```

API: Packaging

Build partner packages with policy packs and idempotent IDs.

- POST to /packages/{partner}; returns build ID.
- GET to fetch logs and artifacts.
- Retry partial failures only.
- Policy pack pinning by version.

```
POST /v1/packages/spotify?release=sso_2025_00041&policy=SPOTIFY_V2025_02
```

API: Delivery

Transport artifacts to partners; receipts and error mapping.

- Async delivery jobs with receipts archived.
- Webhook on success/failure; retry via job ID.
- Drift detection triggers post-delivery sampling.
- Audit trail binds external IDs.

```
POST /v1/deliveries
{ "partner":"tidal", "package_id":"pkg_00088" }
```

API: Partners

List partner capabilities and constraints; schema discovery.

- GET capabilities; includes media specs and metadata quirks.
- Version pinning per partner profile.
- Diff view when upstream schemas change.
- Simulator endpoint for pre-flight testing.

```
GET /v1/partners/spotify/capabilities
```

API: Webhooks

Signed events for state changes, errors, analytics updates.

- HMAC signatures with timestamp nonce.
- Replay protection with expiring windows.
- Event types: qc.failed, package.ready, delivery.succeeded, royalty.anomaly.
- Retry on 5xx with capped backoff.

```
POST https://yourapp/webhooks/sso
X-Signature: sha256=...
{ "event":"qc.failed", "release":"sso_2025_00041" }
```

API: Reports

Usage/royalty exports and ad-hoc queries.

- CSV/Parquet exports for BI stacks.
- Parameterized queries with limits/sanitization.
- Prebuilt anomaly reports.
- SLA on report generation tasks.

```
GET /v1/reports/royalties?from=2025-01-01&to=2025-01-31&format=csv
```

API: Catalog Intelligence

Forecasts, anomaly detection, and cohort analysis endpoints.

- POST forecast jobs; GET results with confidence intervals.
- Flag anomalies and attach evidence bundles.
- Cohort builder API for campaign analysis.
- Export to dashboard or CSV.

```
POST /v1/intelligence/forecast
{ "release":"sso_2025_00041", "horizon_days":30 }
```

API: Incidents

Open, track, and resolve incidents with templates and SLAs.

- Templates per partner; link to releases and jobs.
- Status transitions generate audit logs.
- Attach artifacts and operator notes.
- SLA timers and escalations.

```
POST /v1/incidents
{ "template":"ANGHAMI_SCHEMA_DRIFT", "release":"sso_2025_001512" }
```

API: Admin

RBAC, policy packs, and environment settings.

- Manage users, roles, and elevation windows.
- Pin/unpin policy packs; preview diffs.
- Rotate secrets; set quotas and rate limits.
- Export audit feed.

```
PATCH /v1/admin/policies/APPLE_V2025_01
{ "status":"pinned" }
```

Data Dictionary 1: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 1: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 2: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 2: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 3: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 3: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 4: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 4: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 5: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 5: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 6: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 6: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 7: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 7: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 8: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 8: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 9: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 9: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Data Dictionary 10: Core Fields

Reference of canonical fields across ERN/MEAD/ADM/TTML with constraints and mappings.

- Field group 10: contributor identifiers, roles, and display rules.
- Constraints: required/optional, max lengths, regex policies.
- Mapping to partner-specific fields and known quirks.
- Examples with valid/invalid cases.

```
# Table-like (markdown)
| Field | Required | Max | Pattern |
|------|----------|-----|---------|
| title | yes | 200 | ^[\w\s'!?,.&-]+$ |
| isrc  | yes | 12  | ^[A-Z]{2}-[A-Z0-9]{3}-[0-9]{2}-[0-9]{5}$ |
| upc   | yes | 12  | ^[0-9]{12}$ |
```

Ops Playbook: Incident Response

Standardized flow from detection → triage → remediation → postmortem.

- Severity matrix informs response time and staffing.
- Runbook steps include safeguards and rollback plans.
- Postmortem templates enforce learning and action items.
- KPIs: MTTA, MTTR, incident recurrence rate.

```
# Incident template (YAML)
severity: SEV-2
detect: qc.failed
owner: oncall-validator
rollback: true
```

Ops Playbook: DMCA Takedown

Evidence pack generation and legal workflow.

- Assemble hashes, manifests, and timestamps automatically.
- Notify parties with standardized templates.
- Track platform responses and timers.
- Close loop with audit and retention.

```
# Evidence pack (CLI)
sso dmca pack --release sso_2025_00041 --out evidence.zip
```

Ops Playbook: Disaster Recovery

Backup/restore drills and RPO/RTO targets.

- Routine restore tests validate backups.
- Documented contacts and responsibilities.
- Failover runbooks for partner outages.
- Communication plans for stakeholders.

```
# DR drill (CLI)
sso dr restore --date 2025-09-01 --env staging
```

Ops Playbook: Change Management

Policy for risky changes and canary rollouts.

- RFC proposals with blast radius estimates.
- Canary scope and abort criteria.
- Rollback checkpoints and metrics gates.
- Changelog announcements and archive.

```
# RFC header (md)
Title: Raise LUFS ceiling for TIDAL
Risk: Medium
Plan: Canary 5% of releases
```

Ops Playbook: SLA/SLO

Define SLAs for packaging, delivery, and response times.

- SLOs with error budgets and burn alerts.
- Partner-specific SLAs tracked separately.
- Dashboards and weekly reports.
- Escalation when budgets exceeded.

```
-- SQL
SELECT stage, error_budget_burn FROM slo_budget WHERE week = :wk;
```

Ops Playbook: Monitoring & Alerting

Telemetry signals and alert routing.

- Metrics, logs, traces shipped to a central sink.
- Alert rules with dedupe and suppression windows.
- On-call schedules and paging policies.
- Runbooks linked from alerts.

```
# Alert rule (YAML)
alert: DeliveryErrorsHigh
expr: rate(delivery_errors[5m]) > 5
for: 10m
labels: {severity: 'page'}
```