SSIS Blueprint #1 v1.4 — Audio Pipeline (Polished + Clarified, No Other Changes)

0) What we are building

SSIS Audio Pipeline is an offline-first, CPU-only, resumable processing spine that turns raw audio into durable, versioned artifacts for downstream ML and product features—designed for environments where power loss is normal, not exceptional.

Primary artifacts produced

- AudioAsset: canonical identity + source metadata
- Normalized WAV: canonical audio derivative (22050 Hz mono PCM WAV)
- FeaturePack: compressed HDF5 containing:
    - log-mel spectrogram
    - YAMNet embeddings via ONNX Runtime CPU
- Segments: speech/music/noise regions + derived silence
- PreviewCandidate: selected preview window with heuristic confidence + fallback mode
- PipelineJob telemetry: per-stage logs, metrics, error taxonomy, attempts
- FeatureSpec registry: lookup table linking short aliases → full feature specs

Non-negotiable design doctrine

1. Idempotency at every stage (safe to rerun after crash/power loss)
2. Atomic publish for every artifact:
    - write temp → flush (best-effort fsync) → rename to final
3. Stage locks with stale-lock reclamation:
    - lock key prevents duplicate work after restarts
4. Versioned contracts from day one:
    - every artifact has schema_id and version
5. Resilience tests in the spec:

- kill mid-write, restart, verify correctness

---

1) Locked decisions (v1.4)

Orchestration & persistence

- Task orchestration: Huey
- Queue backend: SQLite (durable queue state offline)
- Metadata DB: SQLite (assets, jobs, locks, artifact index, feature specs)
- Default concurrency: 1 worker (safe baseline; configurable)

Audio + ML

- Canonical audio derivative: 22050 Hz mono PCM WAV
- Embedding model: YAMNet via ONNX Runtime CPU
- Segmentation model: inaSpeechSegmenter (speech/music/noise)
- Silence: derived (gaps + optional VAD assist)
- Feature storage: HDF5 + gzip (single writer discipline)
- Preview selection: heuristic v1 (boundaries + energy variance + embedding variance), fallback to intro

Artifact evolution & reprocessing

- Never overwrite artifacts. New configs/models → new FeatureSpec → new artifacts.
- Feature configs are referenced by a short alias to avoid long identifiers.

---

2) Scope and boundaries

In scope (Blueprint #1)

- Ingest: local path + file upload
- Normalize: decode → canonical WAV
- Feature extraction: mel + YAMNet(ONNX) → HDF5

- Segmentation: inaSpeechSegmenter + silence derivation
- Preview selection: heuristic + fallback
- Observability: SQLite jobs + JSON logs + error taxonomy
- Resilience test harness (power cut simulation)

Out of scope (Blueprint #1)

- Training custom models / weak supervision
- CLAP/MERT semantic search
- Transcription/diarization (optional later)
- Distributed multi-node processing
- UI/permissions system

---

3) Repository layout (authoritative)

```
ssis/
  audio_pipeline/
    README.md
    pyproject.toml

    specs/
      audio_asset.schema.json
      pipeline_job.schema.json
      feature_pack.schema.json
      segments.schema.json
      preview_candidate.schema.json
      ingest_local_request.schema.json
      preview_feedback.schema.json        # optional future contract

    app/
      config.py
      db.py
      models.py                  # AudioAsset, PipelineJob, StageLock, ArtifactIndex, FeatureSpec
      schemas.py
      huey_app.py
      orchestrator.py
      utils/
        hashing.py
        audio_meta.py
        atomic_io.py
        checkpoints.py
        paths.py
```

```
services/
  ingest_api/
    main.py
  worker_decode/
    run.py
  worker_features/
    run.py
    yamnet_onnx/
      yamnet.onnx
      yamnet.onnx.sha256
  worker_segments/
    run.py
  worker_preview/
    run.py

data/
  audio/{asset_id}/
  features/
  segments/
  preview/
  logs/

tests/
  contract/
  e2e/
  resilience/
```

---

4) Canonical paths and publish rules

Canonical storage paths

- Original audio copy:
    - data/audio/{asset_id}/original.<ext>
- Normalized canonical derivative:
    - data/audio/{asset_id}/normalized.wav
- Feature packs (versioned by feature_spec_alias):
    - data/features/{asset_id}.{feature_spec_alias}.h5
- Segments:
    - data/segments/{asset_id}.segments.v1.json
- Preview:
    - data/preview/{asset_id}.preview.v1.json

Atomic publish rule (mandatory)

Every artifact write must:

1. write to temp path in same directory
2. flush + best-effort fsync
3. rename temp → final (the publish boundary)

HDF5 special rule

- Never write directly to final .h5.
- Always write to .h5.tmp, then rename to .h5 only when fully valid.
- If a power cut occurs mid-write, the temp file may remain and must be ignored/cleaned on restart.

Single-writer rule

Only worker_features writes HDF5. All other stages write JSON only.

---

5) FeatureSpec: readable IDs + short aliases (with immutability)

Why

Readable feature_spec_id strings can become long; we keep them human-auditable while using short aliases for filenames and locks.

FeatureSpec fields

- feature_spec_id (human-readable, canonical string)
- feature_spec_alias (short stable alias) = first 12 chars of sha256(feature_spec_id)
- created_at, notes

Example

- feature_spec_id: mel64_h10ms_w25ms_sr22050__yamnet1024_h0.5s_onnx
- feature_spec_alias: a1b2c3d4e5f6

Table:

feature_specs

- alias (pk)
- feature_spec_id
- created_at
- notes

Locked immutability rule (v1.4)

feature_spec_alias is frozen at first registration:

- If alias does not exist → insert (alias → feature_spec_id)
- If alias exists and spec matches → no-op
- If alias exists but spec differs → hard error FEATURE_SPEC_ALIAS_COLLISION (and log job failure)

---

6) Data contracts (spine contracts)

Every artifact includes:

- schema_id (e.g., segments.v1)
- version (e.g., 1.0.0)
- asset_id
- computed_at
- recommended: pipeline_version, worker_version

FeaturePack contract (v1.4 must include)

- feature_spec_alias
- feature_spec_id
- embedding_backend: "onnxruntime"
- embedding_model_id: "yamnet"
- embedding_model_hash: "sha256:<...>"
- dataset pointers:
    - hdf5_uri: data/features/{asset_id}.{alias}.h5#/<dataset>

---

7) Database entities (minimum viable)

Tables

1. audio_assets
2. pipeline_jobs
3. stage_locks
4. artifact_index (recommended)
5. feature_specs

Stage locks (v1.4)

Lock key:

- (asset_id, stage, feature_spec_alias|null)

Rules:

- Acquire before stage begins
- Release at end
- Reclaim stale lock if older than TTL (e.g., 10 minutes)
- Log all lock acquisition/reclamation in pipeline_jobs.metrics_json

---

8) Pipeline stages (authoritative execution plan)

Stage A — Ingest (FastAPI)

Input: local path or upload

Actions:

- copy file into SSIS-managed path
- compute sha256 hash
- extract best-effort metadata (duration/channels/sr)
- enforce idempotency via (owner_entity_id, content_hash)

Outputs:

- AudioAsset row
- PipelineJob(stage="ingest")
- enqueue orchestrator start

Failure codes: INGEST_FAILED, FILE_NOT_FOUND, HASH_FAILED

Stage B — Decode + Normalize (worker_decode)

Input: AudioAsset.source_uri

Goal: produce canonical wav

Outputs:

- data/audio/{asset_id}/normalized.wav (atomic publish)
- PipelineJob(stage="decode")

Resilience:

- process in chunks
- checkpoint every ~60s processed audio
- restart resumes safely

Error codes: CODEC_UNSUPPORTED, FILE_CORRUPT, FILE_TOO_SHORT

Stage C — Feature extraction (worker_features)

Input: normalized wav

Goal: mel + embeddings saved in HDF5

FeatureSpec resolution

- Determine the active feature_spec_id for v1.4 (locked default).
- Compute feature_spec_alias and upsert into feature_specs under immutability rule.

Outputs

- data/features/{asset_id}.{alias}.h5 (atomic publish via .tmp)

- FeaturePack metadata record (DB and/or JSON)
- PipelineJob(stage="features")

Config (locked default FeatureSpec)

- Log-mel: 64 mel, 25ms window, 10ms hop, sr=22050
- YAMNet embeddings: 1024-D, hop=0.5s, onnxruntime CPU

Validation

- no NaN/Inf → else FEATURE_NAN
- shape checks

Error codes: FEATURE_NAN, MODEL_OOM, FEATURE_EXTRACTION_FAILED, FEATURE_SPEC_ALIAS_COLLISION

Stage D — Segmentation (worker_segments)

Input: normalized wav

Model: inaSpeechSegmenter → speech/music/noise

Silence: derived from gaps and optional VAD assist

Output

- data/segments/{asset_id}.segments.v1.json (atomic publish)
- PipelineJob(stage="segments")

Confidence semantics (clarified)

- confidence is heuristic, not calibrated logits.
- Include confidence_type: "heuristic_v1".

Error codes: SEGMENTATION_FAILED

Stage E — Preview selection (worker_preview)

Inputs

- normalized wav

- segments JSON
- embeddings from HDF5 for a specific feature_spec_alias

Locked FeatureSpec selection rule (v1.4)

Preview reads embeddings from the FeaturePack matching the active feature_spec_alias, chosen deterministically:

1. If config SSIS_ACTIVE_FEATURE_SPEC_ALIAS is set, use it if that FeaturePack exists for this asset.
2. Else use the pipeline default feature_spec_alias (v1.4 shipped default).
3. If the selected FeaturePack does not exist, fail fast with FEATUREPACK_MISSING and the orchestrator must (re)run features.

Algorithm v1

1. identify candidate boundaries:
    - pause boundaries (>200ms low-energy)
    - segment boundaries in speech-heavy regions
2. generate candidates (default 60s)
3. score candidates:
    - energy variance (RMS variance)
    - embedding variance (diversity proxy)
4. select best above threshold else fallback:
    - intro = first non-silent window

Output

- data/preview/{asset_id}.preview.v1.json (atomic publish)
- PipelineJob(stage="preview")

Error codes: PREVIEW_LOW_CONF (not fatal; triggers fallback), FEATUREPACK_MISSING

---

9) Orchestrator semantics (Huey + SQLite)

Responsibilities

- Determine required stages based on artifact existence + artifact index
- Acquire stage locks and dispatch tasks in order
- Enforce retries/backoff
- Log every attempt in pipeline_jobs
- Dead-letter repeated failures

Retry policy (locked)

- 3 attempts
- delays: 60s, 300s, 900s
- dead-letter after failures with error taxonomy

Idempotency rules (locked)

- If final artifact exists, stage is skipped.
- Temp artifacts are ignored and cleaned.
- If a lock is held and not stale, stage is not duplicated.

---

10) Observability & debuggability (must ship in v1.4)

Required stage metrics

- ingest: file size, hash time, format guess
- decode: output duration, chunk count, resample time
- features: inference time, mel/embedding shapes, NaN/Inf count, spec alias/id
- segments: segment count, class distribution, flip rate
- preview: candidate count, best score, fallback_used, spec alias used

Pipeline-level health

- end-to-end latency (ingest → preview ready)
- success rate

- backlog depth
- error breakdown by code

Minimal error taxonomy (locked)

- CODEC_UNSUPPORTED
- FILE_CORRUPT
- FILE_TOO_SHORT
- FEATURE_NAN
- MODEL_OOM
- SEGMENTATION_FAILED
- PREVIEW_LOW_CONF
- FEATUREPACK_MISSING
- FEATURE_SPEC_ALIAS_COLLISION

---

11) Resilience tests (required, in blueprint)

Contract tests

- validate API + artifact JSON against JSON schemas

End-to-end smoke test

- ingest → orchestrator → decode → features → segments → preview

Load-shedding simulation tests (must ship)

Test harness must:

- kill worker mid-write:
    - during decode
    - during HDF5 .tmp write
- restart pipeline and assert:

- no corrupt final artifacts
- temp files ignored/cleaned
- stale locks reclaimed
- pipeline completes successfully
- idempotency prevents duplication

---

12) Re-extraction and versioning policy (explicit)

When feature config changes

- Create a new feature_spec_id
- Generate new feature_spec_alias
- Produce new HDF5 artifact {asset_id}.{alias}.h5
- Keep prior artifacts intact
- ArtifactIndex tracks which FeaturePacks exist per asset

No backfill is forced; backfill becomes an explicit batch job later.

---

13) Optional future contract: Preview feedback (for learning later)

Define early so later blueprints can plug in without redesign.

Contract: preview.feedback.v1 (optional)

- asset_id
- preview window
- listen/skip metrics
- user actions
- timestamp

Blueprint #1 does not require collection.

---

14) Implementation plan (exact build order)

1. Contracts + DB primitives
    - schemas in /specs
    - DB models: AudioAsset, PipelineJob, StageLock, FeatureSpec, ArtifactIndex
    - utilities: atomic_io, checkpoints, hashing, paths
2. Ingest API
    - local + upload
    - idempotency enforcement
    - enqueue orchestrator
3. Orchestrator (real)
    - stage planning + dispatch
    - locks + stale lock reclamation
    - retries + dead-letter
4. Decode worker
    - canonical wav
    - chunking + checkpointing
    - atomic publish
5. Features worker (Option B ONNX)
    - mel + embeddings via onnxruntime CPU
    - HDF5 .tmp atomic publish
    - FeatureSpec upsert with immutability enforcement
    - validation + telemetry
6. Segments worker
    - inaSpeechSegmenter + silence derivation
    - heuristic confidence + post-processing
7. Preview worker

    - deterministic FeatureSpec selection rule
    - candidate scoring + fallback
8. Resilience harness
    - kill/restart tests + lock reclamation assertions
9. MVP acceptance
    - offline CPU run
    - deterministic artifacts
    - safe restart after interruption
    - job telemetry sufficient for debugging

This document is only Blueprint #1 — the SSIS Audio Pipeline module.

It's 1 out of the 3.

- ✅ Blueprint #1 (this one): Audio Pipeline — ingest → normalize → features (mel + YAMNet ONNX) → segments → preview + observability/resilience.
- ⏭️ Blueprint #2 (next): LLM Safety + Query Compiler — natural language → typed/validated command plans (no direct execution).
- ⏭️ Blueprint #3 (after): Real-time Scoring + Explainability — fast scoring + reason codes + feedback loops.

All three will later be linked into one complete SSIS system, but the doc I just wrote to you is specifically the module blueprint for #1.