FLUX WIKI
OPEN SOURCE

NODE ARCHITECTURE SPECIFICATION FOR THE FLUX PROTOCOL — WHAT A FLUX NODE REQUIRES, WHAT MUST BE BUILT BEFORE PUBLIC RELEASE, AND THE OPEN SOURCE PHILOSOPHY.

flux.dantesisofo.com/wiki/open-source/

FLUX_WIKI_v2.0

JUNE 2026

FLUX DOCUMENTATION SYSTEM Layer 10 — OPEN SOURCE | open-source
flux.dantesisofo.com/wiki/open-source/

OPEN SOURCE

1. THE VISION

Any photographer should be able to run their own FLUX node.

Own their archive. Generate their own issues. Print them. Publish them. Avoid platform dependency entirely.

The FLUX protocol is open. The specification — 36 frames, canonical filename, 44-page PDF, manifest — is public. Any implementation that produces compliant output is a valid FLUX node.

The current implementation is Dante Sisofo's personal node. It runs on specific hardware, is configured for one photographer, and has hardcoded paths and identifiers. This document specifies what a generalized, portable node requires.

2. WHAT THIS IS NOT YET

This document is not an installer.

It does not package FLUX for public release. There is no setup.sh, no Docker image, no pip install flux. The current codebase contains hardcoded paths, single-photographer assumptions, and implementation details that assume specific hardware.

This document defines the requirements and assumptions that a future packaged system must satisfy. It is a specification, not a release.

The requirements here are accurate. The implementation is not yet portable.

3. CORE NODE REQUIREMENTS

Hardware minimum:

```
CPU:     x86_64 or ARM64 (Apple Silicon acceptable)
RAM:     8GB minimum; 16GB recommended for embedding generation
OS:      macOS 12+ or Linux (Ubuntu 22.04+)
Python:  3.11+
Scratch: 50GB local SSD for processing (not for canonical archive storage)
Network: local network for NAS mount; internet for S3 sync and geocoding
```

Storage minimum:

```
Archive drive:  1TB minimum (for a modest corpus)
Recommended:    NAS with mirrored drives (RAID-1 or equivalent)
Not acceptable: single unmirrored drive as canonical archive
```

Network minimum:

```
Local:    1GbE for NAS mount (10GbE preferred for large corpus operations)
Internet: standard broadband; S3 sync is bandwidth-bounded, not latency-bounded
```

4. SOFTWARE DEPENDENCIES

Python packages:

```
reportlab>=4.0        — PDF generation (cover, protocol page, images, contact sheet, manifest)
boto3>=1.28           — AWS S3 API (PutObject, GetObject, ListObjectsV2, DeleteObject, CopyObject)
Pillow>=10.0          — Image processing, JPEG handling, contact sheet generation
markdown>=3.5         — Wiki rendering
exifread>=3.0         — EXIF extraction from JPEG files
flask>=3.0            — Portal web server (local, not internet-facing)
requests>=2.31        — HTTP client for geocoding and weather API calls
numpy>=1.25           — Vector operations for embeddings
```

Optional (intelligence layer):

```
torch>=2.0            — PyTorch for CLIP inference
open-clip-torch>=2.20 — Open CLIP model weights
sqlite-vec            — SQLite vector search extension
```

System tools:

```
ImageMagick           — JPEG manipulation (optional; Pillow covers most cases)
exiftool              — Alternative EXIF reader (optional; exifread preferred)
```

Install:

```
pip install reportlab boto3 Pillow markdown exifread flask requests numpy
pip install torch open-clip-torch          # optional: intelligence layer
```

5. STORAGE ASSUMPTIONS

Canonical NAS folder structure:

```
/FLUX_ARCHIVE/
    /ORIGINALS/             — full photographic corpus (all frames, chronological)
        /YYYY/
            /YYYY-MM/
                /YYYY-MM-DD/
                    YYYY-MM-DD_HH-MM-SS_PhotographerName_R0000001.JPG
    /KEEPERS/               — selected keeper images (positive training class)
        /YYYY/
            /YYYY-MM/
                /YYYY-MM-DD/

/FLUX_SYSTEM/
    /scripts/               — all Python scripts
    /venv/                  — Python virtual environment
    /config/                — configuration files (see Section 6 below)

/FLUX_PUBLIC/
    /wiki/                  — generated wiki HTML + PDFs
```

```
    /issues/                — generated PDF issues (local copy before S3 sync)

/FLUX_METADATA/
    flux.db                 — SQLite metadata database
    /manifests/             — per-issue manifest.json files
    /backups/               — nightly database snapshots

/FLUX_EMBEDDINGS/
    flux_vectors.db         — SQLite-vec vector database

/FLUX_ISSUES/
    /FLUX_001/
        FLUX_001.pdf
        manifest.json
    /FLUX_002/
        ...

/FLUX_LOGS/
    ingest.log
    approve.log
    builder.log
    deploy.log

/FLUX_INBOX/
    [incoming photographs for processing]
```

Local scratch (on Mac mini SSD, not NAS):

```
/tmp/flux_processing/       — temporary files during PDF generation
/tmp/flux_issue_builder.lock
```

The canonical archive must be on mirrored storage. An unmirrored single drive is not acceptable for `/FLUX_ARCHIVE/`. Losing the archive is not recoverable from S3 — S3 holds published assets, not the complete corpus.

---

6. NAMING CONVENTIONS

The canonical filename format is part of the FLUX protocol. Any FLUX node must generate and respect this format.

```
YYYY-MM-DD_HH-MM-SS_PhotographerName_OriginalFilename.JPG
```

Rules: - Timestamp uses hyphens within date and time fields, underscore between date and time - PhotographerName is CamelCase, no spaces - OriginalFilename is the camera-generated filename, unchanged - Extension is always `.JPG` (uppercase) - Filenames are immutable once generated

A node that generates filenames in any other format is not FLUX-compliant.

---

7. ISSUE STANDARD

`FRAMES_PER_ISSUE = 36` is a protocol constant, not a configuration option.

A FLUX node that generates issues with any frame count other than 36 is not FLUX-compliant. A configuration file that sets `FRAMES_PER_ISSUE = 24` does not produce a

valid FLUX issue.

The PDF structure is also a protocol constant: 44 pages, fixed page layout (see DECISION-002). A node that generates 40-page PDFs or uses a different page layout is not FLUX-compliant.

These constants are not parameters. They are the protocol.

---

8. S3 COMPATIBILITY

The current implementation uses AWS S3. A self-hosted node could substitute any S3-compatible object storage service.

S3 API surface used by the FLUX implementation:

```
PutObject          — upload photographs, PDFs, manifests, catalog.json
GetObject          — download catalog.json, manifests, individual photographs
ListObjectsV2      — scan FLUX_CATALOG/ prefixes for ID generation; scan FLUX_ISSUES/ for issue inventory
DeleteObject       — cleanup during re-generation or retraction
CopyObject         — S3-side copy for efficient thumbnail duplication
```

Compatible services:

```
AWS S3             — reference implementation; currently in use
MinIO              — self-hosted, S3-compatible; suitable for fully local nodes
Backblaze B2       — S3-compatible API available; lower cost than AWS
Cloudflare R2      — S3-compatible; no egress fees
```

Any service that implements these five S3 API operations is drop-in compatible.

CloudFront (CDN) is separate from S3 and is not required for a local node. A local node that does not publish publicly does not need a CDN.

---

9. PRINT PIPELINE ASSUMPTIONS

The physical print pipeline assumes: - A laser printer accessible on the local network (AirPrint or IPP protocol) - PDF sent via system print command - No cloud print services; no internet dependency for printing - Human assembly: fold, staple (see DECISION-011)

```
# macOS print command
lp -d PrinterName -o fit-to-page FLUX_019.pdf

# Linux (CUPS)
lp -d PrinterName FLUX_019.pdf
```

Printer requirements: - Duplex printing (two-sided, for correct spread layout) - Letter or A4 paper (PDF is generated for Letter; A4 adaptation requires regeneration) - Laser preferred: inkjet produces inconsistent results at small text sizes

Gutter calibration is printer-specific. The current calibration is for a Brother laser printer. A different printer may require recalibration (see CHANGELOG v1.2, gutter compensation).

---

10. WHAT NEEDS TO BE BUILT BEFORE OPEN SOURCE RELEASE

The current codebase is not ready for public release. These items must exist:

1. Configuration file
   Replace all hardcoded paths, bucket names, CloudFront IDs, and
   photographer names with a config.yaml or config.toml that is
   generated during setup.

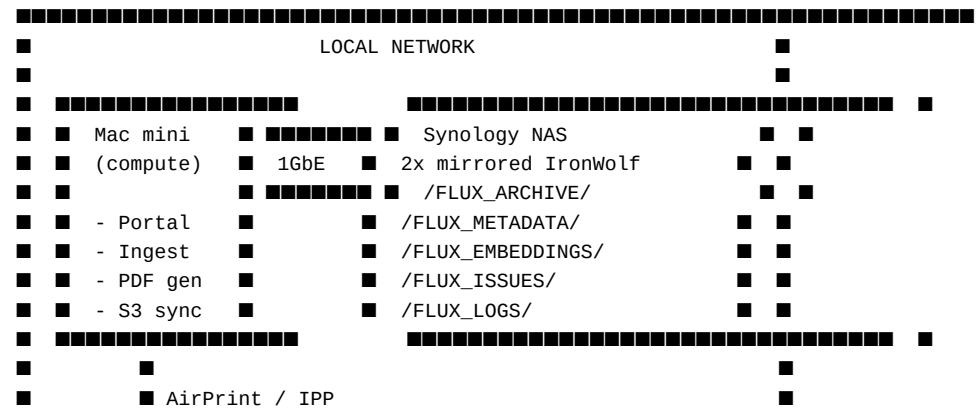
2. Setup script
   setup.sh or setup.py that:
   - Creates the NAS folder structure
   - Initializes the SQLite database
   - Configures S3 credentials and bucket
   - Sets photographer name and canonical filename prefix
   - Tests all connections (S3, printer, NAS mount)

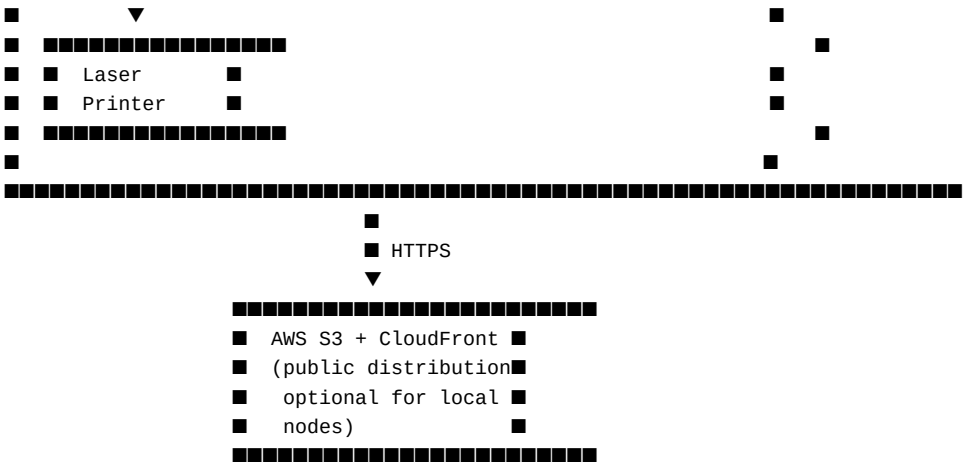
3. Multi-photographer support
   The current implementation assumes one photographer (Dante Sisofo).
   A generalized node supports one photographer per installation.
   Multi-photographer support (e.g., a shared node) requires:
   - Per-photographer namespaces in the database
   - Per-photographer S3 prefixes
   - Per-photographer canonical filename prefixes

4. Documentation for non-technical setup
   The current wiki assumes technical knowledge.
   A public release requires:
   - Step-by-step setup guide (macOS and Linux)
   - Troubleshooting guide
   - FAQ

5. Test suite
   No tests currently exist. A public release requires:
   - Unit tests for canonical filename generation
   - Unit tests for PDF page count verification
   - Integration tests for S3 upload/download
   - End-to-end test for full issue generation pipeline

11. RECOMMENDED ARCHITECTURE FOR SELF-HOSTED NODES





Minimum viable node (no cloud): - NAS or large external drive (mirrored) - Any Mac or Linux machine with Python 3.11+ - Local laser printer - No S3 required (PDFs stored locally only)

Full node (with public distribution): - NAS (mirrored) - Mac mini or equivalent compute - Laser printer - AWS S3 + CloudFront (or S3-compatible equivalent)

---

12. OPEN SOURCE PHILOSOPHY

The FLUX protocol is open.

The specification is public: 36 frames, canonical filename format, 44-page PDF structure, manifest schema, contact sheet geometry. Any implementation that produces compliant output is a valid FLUX node.

The implementation (the specific Python scripts in this repository) is currently unlicensed. A public release will include an explicit open source license (MIT or Apache 2.0 — decision deferred to release time).

What is part of the protocol (open, standard, immutable): - `FRAMES_PER_ISSUE = 36` - PDF page count: 44 - PDF page layout (as specified in DECISION-002) - Canonical filename format - Contact sheet geometry (6×6) - Manifest format (2 columns × 18 rows)

What is implementation detail (can vary between nodes): - Storage backend (S3 vs. MinIO vs. Backblaze vs. local) - Database engine (SQLite vs. PostgreSQL) - Web portal implementation - Embedding model (CLIP variant, dimensions) - Keeper model architecture - Print pipeline specifics

A node that uses a different storage backend but generates 36-frame, 44-page PDFs with compliant filenames is a valid FLUX node.

A node that generates 24-frame PDFs is not.

SEE ALSO

Document	Layer	Relationship
PROTOCOL	Layer 2 — Protocol	The canonical specification this layer aims to open-source
DECISIONS LOG	Layer 9 — Governance	DECISION-001 through DECISION-011 define the protocol constants
BOOTSTRAP	Layer 4 — Infrastructure	Current hardware architecture that the open-source node spec generalizes
ROADMAP	Layer 8 — Roadmap	Timeline for open source release work

FLUX_WIKI_v2.0 — flux.dantesisofo.com/wiki/open-source/