# The Sovereign Monograph: Pipelines of Digital Autonomy

## The Sovereign Scholarly Monograph: Reclaiming Intellectual Autonomy and Aesthetic Excellence through High-Standard Digital Distribution Pipelines

zeba.academy

Zeba Academy

**First Published by Zeba Academy and Zeba Books.**

**Publication Year:** 2026

**Document Series:** Zeba Academy Blueprints -- Sovereign Systems Technical Directives

**Purpose:** This series of blueprint directives is authored to combat the "enshittification" and unnecessary bloat of modern software. Our goal is to reclaim sovereign control over our systems by bridging the gap between deep academic theory and high-stakes industrial implementation. We believe that software should be fast, permanent, and most importantly, understandable to the person who owns and uses it.

**Principal Architect: Sufyan bin Uzayr,** Google Cloud-Certified Professional DevOps Engineer.

**Core Stack:** Linux, Rust, Zig, C++, Flutter, and PHP.



**Email:** hello@zeba.academy

# The Sovereign Monograph: Pipelines of Digital Autonomy

## *The Sovereign Scholarly Monograph: Reclaiming Intellectual Autonomy and Aesthetic Excellence through High-Standard Digital Distribution Pipelines*

## The Monograph as a Computable Intellectual System

Today's scientific monograph should be seen as a computable, schema-driven digital object, meaning a structured electronic document that can be automatically processed by computers using defined data templates (schemas). This design allows computers to read and handle their information, not just for people to read. The main goals include machine interoperability, where different computer systems can exchange and make sense of each other's data, semantic extraction, which refers to computers identifying structured meanings and relationships in the text, and protocol-level integration, where automated systems manage how data is shared across scholarly platforms.[1]

A sovereign monograph is a digital data format that consists of multiple clearly defined sections. Each section, or "layer," can be accessed separately using a multi-layered architecture. These components are connected by persistent identifiers, which are unique, permanent tags that reference information, and by transformation pipelines, which are automated processes

[1] Journal Article Tag Suite - https://jats.nlm.nih.gov/ - Accessed: 19 March 2026

for converting and linking data across layers. This setting ensures that all components of the monograph work properly.

- **Canonical XML** (using BITS or a JATS extension), which is a structured markup language for scientific content, enforces structural semantics (how document parts relate) using XSD (XML Schema Definition).
- **The Identifier Layer includes DOI** (Digital Object Identifier, used by Crossref/DataCite for persistent links), ISBN (International Standard Book Number), and fragment-level URIs (Uniform Resource Identifiers for referencing parts of documents) for sub-document resolution.
- **Metadata layers include ONIX 3.0** (a standard format for book industry metadata), MARC21 (a widely used library cataloging format), and Crossref XML (a metadata format for managing the citation graph).
- **The distribution layer includes OAI-PMH** endpoints, RESTful APIsPIs, and schema.org JSON-LD exposure.
- **Preservation Layer: OAIS-aligned** (Open Archival Information System, an archival framework) archiving packages such as CLOCKSS, Portico (digital preservation services), and BagIt (a packaging format for storage).

## Implementation Model: Canonicalization Pipeline

The production workflow must establish a singular authoritative source of truth. This will eliminate formatting discrepancies and the resulting inconsistencies.

```
Author Input (DOCX, Markdown, or LaTeX)
↓
Pandoc Parsing Engine →
Abstract Syntax Tree (AST)
↓
Custom XSLT / Lua Filters (semantic normalization)
BITS XML (checked against the XSD schema)
```

Key constraints:

- Clearly label all structural elements (chapters, sections, figures, references) per project style guidelines and requirements.
- During AST transformation, inline semantics such as citations, cross-references, and footnotes must be preserved.
- XML validation forms a non-negotiable constraint in the pipeline (the build fails if the schema is broken).

This canonical XML is the source for deterministic file conversions. These include EPUB, HTML5, PDF/UA, ONIX, and Crossref deposits.[2]

## Structural Addressability and Fragment Resolution

The monograph must provide fine-grained addressability, including citation, indexing, and sub-document-level retrieval.

The hierarchical model follows this structure:

```
/book
/chapter[id]
```

[2] JATS (Journal Article Tag Suite) - https://jats.nlm.nih.gov/extensions/bits/ - Accessed: 19 March 2026

`/section[id]`

`/subsection[id].`

The fragment identifiers use DOI-based URIs, such as `https://doi.org/10.xxxx/monograph#ch2.sec3.` The following actions are needed:

- Add `XML:id` properties to all structural nodes.
- Maintain a consistent ID scheme across all output formats, and do not allow IDs to vary between formats.
- Create anchor maps in HTML and EPUB. This enables direct linking.

This enables:

- section-level citation indexing.
- It supports deep linking in systems for finding things
- It also allows the creation of a graph node for each structural unit.

## Computability and Graph Compatibility

The XML must be semantically enhanced and machine-parsable to be included in knowledge graphs.

DOIs must be linked to elements, and ORCID identifiers must be included in elements. must match restricted vocabularies such as LCSH and BISAC.Step after processing:`BITS XML → Entity Extraction → RDF/JSON-LD → Graph ingestion`

Without this normalization, the monograph is still hard to understand for:

- Pipelines for NLP

- Adding to the citation graph
- Systems for disambiguating entities

## System-Level Implications

- If structural encoding isn't enforced, the entire text can't be indexed (only PDFs can).
- Loss of citation detail
- Graph nodes that aren't connected (no connections between entities)
- Failure to ingest into library and aggregator systems

## Final Thoughts

The sovereign monograph must be organized as a system with multiple levels, each grouping information in a specific order. It should use rules (called schema validation) to ensure the information is structured consistently. Each piece of data should have a unique reference, making it easy to point to specific information. Rather than being a traditional document that simply flows from beginning to end, it serves as a flexible structure for knowledge that many systems can use together. The design allows for changes to be made in clear, predictable ways, makes it easy for computers to read, and ensures the information is always available across different computer networks.

# The Open Access Paradox: Engineering Discoverability Pipelines

Open Access (OA) eliminates access controls at the network layer; however,

it does not ensure integration with discovery, indexing, and citation frameworks. Discoverability is not a feature of openness; rather, it results from a complete pipeline, adherence to the rules, and information that machines can read.

## Main Idea

A monograph that cannot be automatically located, managed, or identified using unique identifiers is effectively invisible within academic systems.[3]

## Minimum Discoverability Stack (System Requirements)

The following interoperable layers must be present in a production-grade pipeline:

- Layer for resolving identifiers: DOI (Crossref/DataCite), ISBN, and URI fragments
- ONIX 3.0, Crossref XML, and Dublin Core are all examples of metadata serialization layers.
- The harvesting layer has OAI-PMH endpoints with multiple metadata prefixes.
- Web Semantics Layer: schema.org JSON-LD is built into HTML

[3] ORCID Integration Guidelines - https://info.orcid.org/documentation/ - Accessed: 19 March 2026

- Content Parsing Layer: Full-text XML (BITS) made available as HTML/EPUB

It is important that each layer is automatically created from the canonical XML, not manually edited.[4]

## Step-by-Step Implementation

### DOI Registration (Event-Driven Automation)

As a post-validation hook in the build pipeline, DOI minting must be added:

```
on: XML_validation_success
POST https://api.crossref.org/works
{
"DOI": "10.xxxx/monograph",
"type": "book",
"title": "...",
"author": [
{ "given": "...", "family": "...", "ORCID":
"https://orcid.org/..." }
]
}
```

Requirements:

- Give each chapter a DOI, such as/ch1 or /ch2.
- Keep DOI-to-URL resolution going through landing pages.

[4] Crossref DOI API & Metadata Schema - https://www.crossref.org/documentation/ - Accessed: 19 March 2026

- Keep DOI mappings in an internal registry, such as a database or a JSON index.

### Metadata Transformation Pipeline (Deterministic Generation)

All metadata formats must use BITS XML, a standard for encoding journal articles, and must follow specified transformation rules (procedures for converting data formats).

```
BITS XML
    ├── XSLT → ONIX 3.0 (distribution feeds)
    ├── XSLT → Crossref XML (DOI deposit)
    └── XSLT → Dublin Core (OAI-PMH exposure)
```

Important enforcement:

- ORCID has to be standardized (https://orcid.org/...)
- Controlled vocabularies were used to map subjects to:
- BISAC (business systems)
- LCSH (library systems)
- Ensure an abstract and keywords. Index both items.

Validation:

```
if missing_required_fields:
fail build
```

### OAI-PMH Endpoint Deployment (Harvesting Interface)

Set up a compliant endpoint that shows multiple schemas:

```
GET /oai?verb=ListRecords&metadataPrefix=oai_dc
GET /oai?verb=ListRecords&metadataPrefix=marc21
```

Architecture:

- PostgreSQL / XML-native DB (eXist-db) for the backend
- Service Layer: REST wrapper that makes XML answers
- Pagination: how to handle resumptionTokens for big datasets

This makes it possible to automatically take in:

- Library discovery layers like Primo and EDS
- Services for aggregating and indexing

**Schema.org Injection (Search Engine Parsing Layer)**

Embed JSON-LD in HTML landing pages generated from XML:

```
{
"@context": "https://schema.org",
"@type": "Book",
"name": "...",
"author": [{ "@type": "Person", "name": "...",
"identifier": "ORCID" }],
"identifier": "https://doi.org/10.xxxx/monograph"
}
```

Putting it into action:

- Put in at the HTML transformation stage.
- Check with structured data testing tools.

- Make sure it matches the DOI metadata.

### Full-text Machine Parsing (Layer for Showing Content)

PDF is just kept as a way to show things. To use a machine, you need:

```
BITS XML → HTML5 (semantic)
BITS XML → EPUB 3 (reflowable)
```

Constraints:

- hierarchy kept with IDs
- has captions and metadata elements turned into DOI URLs. Optional:
- Make an XML search index (ElasticSearch/Solr) for full-text searching

## Failure Modes (System-Level)

- Only PDF dissemination (no XML or HTML)
- There are no persistent identifiers (e.g., DOI or ORCID).
- No OAI-PMH endpoint (can't be harvested)
- HTML that is flat or doesn't have any semantic meaning (no structural parsing)
- Metadata that doesn't match between formats

## Final Thoughts

Publication alone does not guarantee that something can be easily found. Instead, using automated methods that follow common standards - such as clear information about the work, trackable identifiers, and full text that

computers can understand, makes content discoverable. A sovereign monograph is not simply made available to the public; it is built so that different search systems can find, list, and access it.[5]

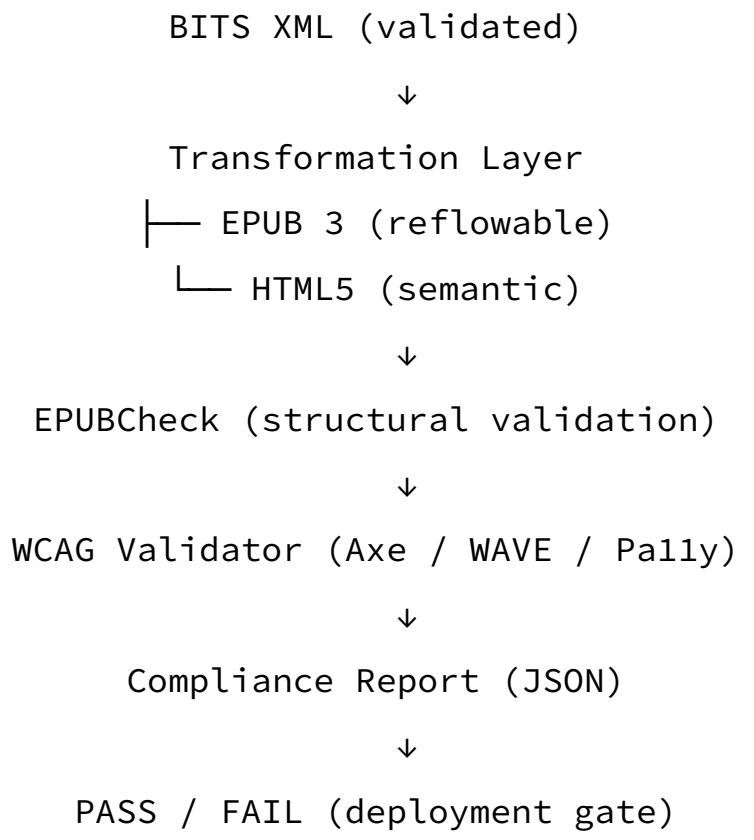
[5] WCAG 2.1 Guidelines (W3C) - https://www.w3.org/TR/WCAG21/ - Accessed: 19 March 2026

# Standards of Excellence: Accessibility and Mobile Systems

Accessibility and mobile compliance must be viewed as non-negotiable validation layers in the publishing pipeline. Automated checks, schema restrictions, and build-time gating must ensure that these layers are always enforced. These are not changes on the editorial side; they are requirements that can be checked by machines and that directly affect ingestion, indexing, and compliance with the law.

## Accessibility Pipeline (WCAG 2.1 as Build Constraint)

Accessibility validation needs to be a defined step within the automated development and deployment process, not something addressed only after production:



Putting it into action:

- Add validation tools to CI (like GitHub Actions and GitLab CI)
- Builds fail on:
- No alt text
- Wrong order of headings
- Violations of ARIA roles
- Keep reports for audit trail purposes.

## Implementation Requirements

### Semantic Structure Enforcement (Schema-Level)

At the XML schema level (XSD or RELAX NG), all structural semantics must be followed:

- Elements must follow a hierarchy (h1 → h2 → h3)
- You must use and to make lists (no inline formatting workarounds)
- Tables must have header associations
- Figures must be clearly stated

```
<fig id="fig1">
<caption>...</caption>
<graphic xlink:href="..."/>
</fig>
```

Validation rule:

```
if invalid_structure:
reject build
```

### Alt-text Injection (Mandatory Metadata)

Every non-textual asset must include alternative descriptions:

```
<alt-text xml:lang="en">Descriptive summary of
figure</alt-text>
```

Plan for automation:

- Create placeholder alt-text while changing XML.
- Use NLP heuristics to flag descriptions that are missing or of poor quality.
- Finally, before the final build, you need to get confirmation from the editor.

Optional improvement:

- AI-assisted alt-text creation with human verification

### PDF/UA Generation (Accessible Binary Output)

PDF generation must originate from structured XML - not visual layout tools:

```
BITS XML → XSL-FO → PDF/UA (ISO 14289 compliant)
```

Technical limits include the

- tagged content tree, which helps set the reading order based on logical structure rather than visual flow.
- In addition, these limits require mapping all Unicode glyphs

- using only built-in fonts with no external dependencies.

Validation tools:

- PAC (PDF Accessibility Checker)
- veraPDF

## Mobile-first Delivery (Rendering Layer Optimization)

The main way to consume must work on every device and be customizable.

**Putting into action**

```
BITS XML
├── EPUB 3 (primary distribution format)
└── HTML5 (responsive rendering)
```

What you need:

- CSS media queries for layouts that change based on the screen size
- Typography that can be scaled (rem/em units)
- Navigation that can be collapsed (TOC, footnotes)

## Performance Optimization (Network-Level Constraints)

Performance has a direct effect on:

- Ranking in search engines
- Keeping users
- Scores for mobile usability

Implementation:

- **Lazy Loading**:

  `<img loading="lazy" ... />`

- **Distribution of CDN:**

- Use edge nodes like Cloudflare and AWS CloudFront to serve static files.

- **TTFB Improvement:**

- Use caching layers such as Redis and Varnish.

- Pre-render XML to HTML

### Progressive Web App (Optional but Strategic)

Enhance delivery through PWA capabilities:

```
Service Worker
      ↓
Cache API (offline access)
      ↓
IndexedDB (state persistence)
```

What it has:

- Reading without being online
- Start tracking your position.
- Navigation like an app

## System-Level Outcome

Accessibility and mobile compliance serve as:

**Indexing Signals** (search engines put stuff that is easy to find first)

**Legal Requirements** (in many places, WCAG compliance is required)

**Ingestion Constraints** (libraries won't accept formats that don't follow the rules)

What happens when you fail:

- Not being able to use discovery platforms.
- Lowered position in search engines
- Not working with assistive technology.

## Final Thoughts

Standards of excellence must be enforced as machine-validated limitations incorporated in the publishing pipeline. A sovereign monograph meets compliance not by manual effort but by enforcing schemas, automating validation, and deterministic transformation. This makes sure that accessibility and mobile readiness are built-in features of the system, not changes made after the fact.

# The Global Knowledge Graph: Integration as Infrastructure

Modern discovery systems function as federated, identifier-driven knowledge graphs rather than as standalone databases. So, a monograph needs to be converted from hierarchical XML to data structures that can be accessed via a graph and have semantic links.[6]

## Graph Construction Pipeline

```
BITS XML
↓
Entity Extraction (NER + rule-based parsing)
↓
Identifier Normalization (ORCID, DOI, ISNI)
↓
RDF / JSON-LD Serialization
↓
Graph Database Ingestion
```

## Step-by-Step Implementation

### Entity Resolution (Identifier Binding)

All entities must use unique, globally resolvable identifiers that are consistent across all records:

➔ ORCID URI (`https://orcid.org/...`)

➔ DOI resolution (`https://doi.org/...`)

[6] RDF 1.1 Concepts (W3C) - https://www.w3.org/TR/rdf11-concepts/ - Accessed: 19 March 2026

➔ ISNI or ROR numbers

Validation rule:

```
if identifier_missing:
flag entity as unresolved
```

### RDF Transformation (Semantic Serialization)

Use XSLT or mapping frameworks to turn XML into triples:

```
BITS XML → RDF/XML or JSON-LD
```

Example triple model:

```
<Author>  —[schema:creator]→  <Monograph>
<Monograph> —[cito:cites]→ <DOI>
```

Limitations:

- Use standard vocabularies like schema.org, Dublin Core, and CiTO.
- Make sure that all nodes have the same URI.

### Graph Storage (Execution Layer)

Select based on the model of the query:

- **Neo4j** → property graph (Cypher queries, quick traversal)
- **Blazegraph / Fuseki ->** RDF triple storage (SPARQL queries)

Ingestion:

```
bulk_load(RDF triples) → index nodes → build
relationships
```

### Metadata Layer Synchronization

Graph integration must align with external systems:

```
BITS XML
├── ONIX → commercial platforms
├── MARC21 → ILS / library catalogs
└── Crossref → citation graph
```

To avoid graph fragmentation, all layers must have the same identifiers.

## Discovery System Ingestion

External systems take in data through:

- OAI-PMH (collecting metadata)
- REST APIs (JSON/XML feeds)
- Crossref event data (connecting citations)

## System Constraints and Failure Modes

- Missing ORCID → duplicate author nodes
- Broken citation edges because of a missing DOI
- Identifiers that don't match up → graph nodes that don't belong to anything

## Result

- A well-integrated monograph turns into:
- You can use SPARQL or Cypher to query

- Able to be resolved (via persistent URIs)
- Composable (connected to outside graphs)

It changes from a static file into a first-class node in a globally distributed scholarly knowledge graph. This enables automated discovery, citation expansion, and semantic interoperability.[7]

## Preservation and Longevity: Engineering Digital Permanence

Preservation should not be a one-time occurrence, but rather an ongoing, verified, and automated activity. The goal is to ensure that bit-level integrity, format survivability, and platform independence remain intact for a long time.[8]

### Archival Packaging (BagIt Compliance Layer)

All preservation units must follow the BagIt standard to make sure that they can be transported safely and reproduced:

```
/bag
/data
book.xml
book.epub
book.pdf
manifest-sha256.txt
bag-info.txt
```

[7] Pandoc Documentation - https://pandoc.org/ - Accessed:19 March 2026
[8] Progressive Web Apps - https://web.dev/progressive-web-apps/ - Accessed: 19 March 2026

Requirements for implementation:

- Make SHA-256 checksums for all payload files.
- Keep the tag manifest for checking metadata.
- Version bags that use identifiers that can't be changed

## Integrity Verification (Automated Fixity Checks)

Preservation requires scheduled validation:

```
cron:
run checksum_verification()
compare stored_hash vs computed_hash
if mismatch:
trigger alert + restore from replica
```

Tools:

- Libraries for validating BagIt
- Fixity audit logs are kept in systems that only allow appending

## Distributed Storage Architecture

Set up a multi-node replication model:

- Main: storage for local objects
- Second, cloud storage (buckets that work with S3 and have versions)
- Tertiary: Archival networks like CLOCKSS and Portico

Replication policy:

```
min_replicas = 3
geo_distribution = required
```

Make sure:

- Redundancy in geography
- Separate areas of administrative control

## Format Migration Strategy (Forward Compatibility Layer)

Preservation must consider format in advance. outdatedness:

```
if format_deprecated:
transform(BITS XML → new schema)
regenerate EPUB/PDF derivatives
```

Main idea:

- XML is still the best format for keeping things safe.
- All derivatives are reconstructible artifacts.

## Ingestion into Preservation Networks

Send structured packages to:

- CLOCKSS (dark archive, release based on a trigger)
- Portico (service for managing preservation)

You must include in your submission:

- Source XML

- Formats that can be rendered (EPUB, PDF)
- Full metadata (Crossref, ONIX)

## Trigger and Access Model

When the following happens, preserved content is turned on:

- The publisher is not active or has shut down.
- A catastrophic breakdown of the platform

Access is controlled by:

- Conditions that are already set
- Policies for archival networks

## Rules and Governance

- **Version Immutability:** yArtifacts that have been published can only be added to
- **Audit Trails:** All changes are recorded with timestamps and hashes.
- **Withdrawal Policies:** Not deletion, but tombstone metadata controls them

## Result

A preserved monograph is:

- **Fixity-verified** (assured bit integrity)
- **Stored in more** than one place (multi-node resilience)

- **Format-migratable** (works with the future)

So, preservation is not only passive storage; it is an active system of monitoring, validation, and controlled transformation.

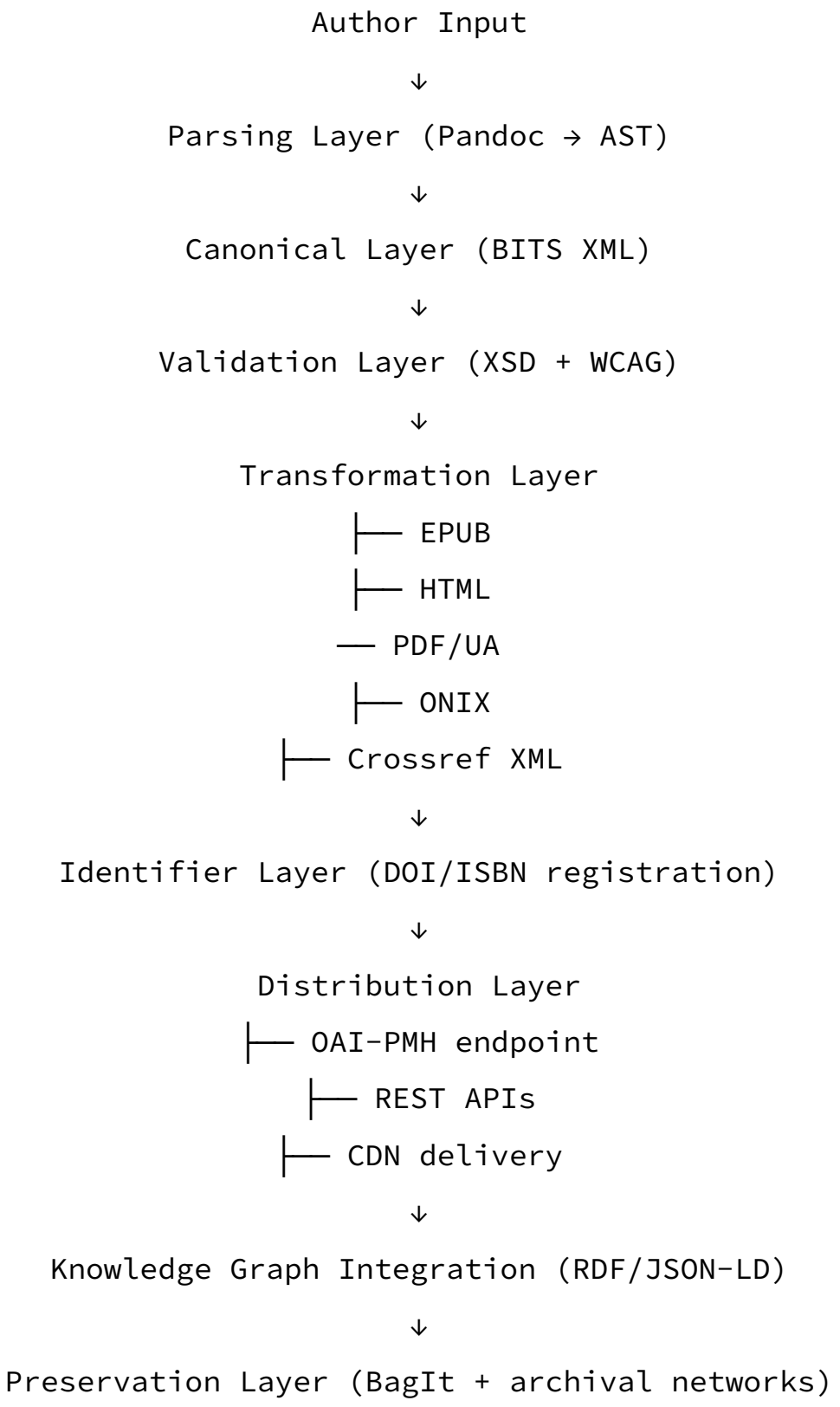
# Closing Remarks: Engineering the Independent Academic Press

The independent academic press needs to be built as a modular, automated, and API-driven publishing system. Modern scholarly infrastructure doesn't rely on manual workflows or on methods that focus on formats.

## Core Infrastructure Stack

- XML-first production (BITS as the main layer)
- Automated metadata pipelines (ONIX, Crossref, MARC)
- Distribution based on APIs (OAI-PMH, REST endpoints)
- Deployment in containers (Docker/Kubernetes orchestration)
- The integration of validation layers guarantees that XML and metadata adhere to schemas and industry standards before dissemination.
- Persistent identifier services (DOI, ORCID) are intimately integrated with information pipelines to enable worldwide resolvability.
- The use of search indexing services (such as Elasticsearch/OpenSearch) to improve discoverability across platforms.
- Support for event-driven processing enables real-time updates and synchronization across the production, metadata, and distribution levels.

## Reference Architecture (End-to-End Pipeline)



## Model for Scalability and Deployment

- Stateless microservices for every step of the pipeline
- CI/CD pipelines start the build, validation, and distribution process.
- Horizontal scaling through container orchestration

Example:

docker-compose / Kubernetes:

- xml-processor

- metadata-generator

- api-server

- oai-provider

## Optional Layer for AI-Augmented Enhancements

- Recognizing named entities for semantic tagging
- Automatic parsing of citations and finding DOIs
- Graph traversal-based recommendation systems

## Final Place

The sovereign monograph isn't a paper; it's a system of information that has been put together, checked for accuracy, and shared.

If this architecture is not put into place, it will:

- Fragmentation of metadata

- Failure to discover
- Dependency on the platform

Sovereignty is attained solely when the monograph transforms into a knowledge object that can be acted on by machines, follows standards, resolves identifiers, and is interconnected around the world.

**Zeba Academy** is a specialized technical research and training initiative dedicated to the principles of Sovereign Systems Engineering. Founded by Sufyan bin Uzayr - an author and university instructor as well as Google Cloud-Certified DevOps Engineer - Zeba Academy serves as a bridge between deep academic theory and high-stakes industrial implementation.

We reject the "enshittification" of modern software. Our core mission is the promotion of Anti-Bloat Architecture through the mastery of:

- **Systems Languages:** Using Rust, Zig, and C++ to build high-performance foundations that prioritize memory safety and deterministic execution.
- **SRE & DevOps:** Professional-grade automation via Google Cloud, Terraform, and Immutable Infrastructure to eliminate manual "toil" and operational fragility.
- **High-Performance Interfaces:** Utilizing Flutter for cross-platform development to deliver near-native mobile experiences without the lag of standard web-based wrappers.
- **Lean Web Publishing:** Reclaiming WordPress and PHP by stripping away the "slop", using Redis object caching and Unix sockets to transform standard platforms into high-speed, GEO-stable engines for modern publishing.
- **Legacy Modernization:** Applying memory-safe paradigms and modern build systems to century-old computational problems and aging C codebases.

Zeba Academy doesn't just teach code; we architect reliability. By merging the analytical rigor of Historical Research with the precision of Google-Certified Cloud Engineering, we provide our "Operatives" with the directives necessary to build systems that are safe, fast, and permanent.

Website:- https://zeba.academy

زیبا

Zeba Academy

zeba.academy