SCHEMA 

SCHEMA Manual

(0-1-20250630-Schumacher.Personal-SCHEMAM)

Creator: **Joshua Schumacher**
Editor(s): **Joshua Schumacher**

Members:
Joshua Schumacher

_____
_____
_____
_____

_____ Print Off

0 - Public

06/30/2025-09/28/2025

Security Clearance: 0 - Public
Name: SCHEMA Manual
ID: 0-1-20250630-Schumacher.Personal-SCHEMAM
Dates: (Started): 06/30/2025
Description: Unified metadata and structuring system designed to bring order, traceability, and permanence to digital content across time, platforms, and organizational contexts. It standardizes how files are named, how versions evolve, how archives are structured, and how documents are formatted, forming the backbone of the Legacy Matrix.

Table of Contents

Summary

“Schumacher Content Hierarchical Encoding & Management Architecture”

SCHEMA — *Schumacher Content Hierarchical Encoding & Management Architecture*

SCHEMA is a unified metadata and structuring system designed to bring order, traceability, and permanence to digital content across time, platforms, and organizational contexts. It standardizes **how files are named, how versions evolve, how archives are structured, and how documents are formatted**, forming the backbone of the **Legacy Matrix**.

1. SCHEMA: File Identification

Purpose:

Encodes critical metadata directly into file names, enabling instant identification of an item's security level, type, origin, creation date, and contextual tags, without requiring external systems or databases.

Format:

[SEC]-[TYPE]-[CREATION_DATE]-[ORG].[SUBORG]-[TITLE][+DOWNLOAD_DATE][+TAGS].[EXT]

Benefits:

- Self-describing file names
- Cross-system portability
- Intuitive structure for archiving and searching
- Security level classification at a glance

2. SCHEMA: Revision Tracking

Purpose:

Tracks both the evolution of personal projects and their corresponding repository state (e.g., GitHub), while clearly distinguishing between internal progress and public releases.

Format:

v[MAJOR].[MINOR].[PATCH].[REPO]-[DATE]-[STAGE]

Benefits:

- Dual versioning for personal project + repository
- Chronologically anchored development states
- Clear separation between working versions and external visibility
- Ideal for changelogs, codebases, and milestone tracking

3. SCHEMA: Archive Management

Purpose:

Organizes all content into annual, version-anchored archive blocks housed within the **Legacy Matrix**. Each archive reflects the SCHEMA version active when the year began.

Format:

```
Legacy Matrix/
└── Archive.XXXX+YYYY@SCHEMAvX.X.X/
```

Shorthand Variants:

- A324S205 (Compact)
- ARC3-2024@S2.0.5 (Readable)

Benefits:

- Chronological segmentation of digital history
- Immutable archives per SCHEMA version
- Clean separation of annual workloads
- Ready for branding, compression, and publishing

4. SCHEMA: Document Formatting

Purpose:

Defines typographic and structural standards for documents created within SCHEMA. Ensures that formatting, metadata, and hierarchy are consistent across all textual content.

Features:

- SCHEMA ID blocks and embedded metadata
- Public-level formatting base
- Defined header hierarchy
- Print/digital-ready layouts
- Compatible with versioning and archival layers

Benefits:

- Professional and consistent output
- Easy parsing of authorship, security, and structure
- Seamless integration into SCHEMA archives and references

In Summary

SCHEMA is more than a file-naming convention; it's a content philosophy.
It bridges identity, structure, time, and evolution into a modular system that can scale with personal projects, legacy archives, or even enterprise-level information systems.

Whether you're tracking a prototype, versioning a release, formatting a report, or archiving a year’s worth of intellectual output, **SCHEMA makes the structure part of the substance.**

File Identification

“Provides a structured, metadata-rich naming convention for digital files. Encodes essential attributes—such as security level, file type, creation date, ownership, and title—directly into the filename, enabling efficient categorization, sorting, and retrieval across projects and archives.”

ID Format Specification

Format Structure

[SEC]-[TYPE]-[CREATION_DATE]-[ORG].[SUBORG]-[TITLE][+DOWNLOAD_DATE][+TAGS].[EXT]

Component Breakdown

Component	Description
SEC	**Security Clearance** – Indicates the security level
TYPE	**Item Type** – The general classification
CREATION_DATE	**Creation Date** – The original date the item was created. If unknown, use the ID creation date. Format: YYYYMMDD.
ORG	**Organization** – The entity responsible or associated with the item. Use your name if personal.
SUBORG	**Sub-Organization** – Further narrows down the association (e.g., department, team). Use Name.Personal for personal projects.
TITLE	**Title Abbreviation** – A concise abbreviation of the item's name (e.g., Smart Dog Collar → SDC).
+DOWNLOAD_DATE	*(Optional)* – The date the file was downloaded (if applicable). Format: +YYYYMMDD.
+TAGS	*(Optional)* – Any additional custom tags for sorting, filtering, or metadata. Format: +TAG1_TAG2.
EXT	**File Extension** – The actual file type (e.g., .pdf, .png, .mp4, .zip).

Example

0-1-20250630-Schumacher.Personal-SCHEMADF.pdf

This represents a **public document** (0-1) created on **June 30, 2025**, by **Schumacher** under their **Personal** archive. The item is titled **SCHEMA: Document Formatting** (SCHEMADF), and the file is in **PDF** format.

Downloaded File Example

0-1-20250630-Schumacher.Personal-SCHEMADF+20250724.pdf

Breakdown

Component	Value	Meaning
0	**Public**	File is accessible to everyone
1	**Word Document**	Item type is a Word-based document
20250630	**Creation Date**	Originally created on **June 30, 2025**
Schumacher	**Organization**	Created by or associated with **Schumacher**
Personal	**Sub-Org**	Personal project
SCHEMADF	**Title Abbreviation**	Short for **SCHEMA: Document Formatting**
+20250724	**Download Date**	This specific file was downloaded on **July 24, 2025**
.pdf	**File Extension**	File is in **PDF** format

Item Type Codes

Code	Type	Description
11	**Network**	Messages, packages, correspondence, or delivery records
10	**Organization**	Companies, institutions, groups, associations, or their structural records
9	**Mail**	Interconnected systems, infrastructures, connections, or information flows
8	**Machine**	Physical constructs, engineered devices, hardware systems, and robotics
7	**Framework**	Conceptual structures, organizational systems, or guiding models for processes and development
6	**Person**	Individuals, profiles, biographies, or related personal records
5	**Protocol**	Standards, procedures, rules, or communication formats
4	**Software**	Executables, scripts, source code, applications
3	**Spreadsheet**	Excel files, Google Sheets, data tables
2	**PowerPoint**	Slide decks or presentations
1	**Word Document**	Text documents, reports, written formats
0	**Other**	Miscellaneous or undefined types

Security Clearance Levels

Level	Label	Access Scope
5	**Top Secret**	Creator only
4	**Uxor**	Creator and their partner
3	**Familia**	Creator’s nuclear family
2	**Familiaris**	Creator’s extended family
1	**Amicus**	Includes all above, plus trusted friends
0	**Public**	Open access to everyone

Clearance Code Protocol

This defines Clearance Code Formats, scaling from readable at low levels to high-entropy with error checks at high levels.

Level 0 — Public

Scope: Open access to everyone
Code Tier: *No code required* (implicit access)

- Example: Public-facing files, shared documentation.

Level 1 — Amicus

Scope: Trusted friends + all lower levels
Code Tier: Tier 1
Format: LDD-LDD

- Example: R42-K71
- Entropy: ~17.5 million
- Use: Lightweight identifiers for files/projects shared with trusted circles.

Level 2 — Familiaris

Scope: Extended family + all lower levels
Code Tier: Tier 2
Format: LLDD-LLDD

- Example: XZ42-QK97
- Entropy: ~4.56 billion

Use: Broader family access; durable but still human-friendly.

Level 3 — Familia

Scope: Nuclear family + all lower levels
Code Tier: Tier 3
Format: LLLL-DDDD-LL

- Example: WQXZ-4921-NK
- Entropy: ~8.2 trillion
- Use: Private archives and Legacy Matrix materials for closest family.

Level 4 — Uxor

Scope: Creator + partner + all lower levels
Code Tier: Tier 4
Format: LLLL-DDDD-LLDD

- Example: KZQM-1284-YH47
- Entropy: ~820 trillion
- Use: Shared control with partner, administrative power, high-trust.

Level 5 — Top Secret

Scope: Creator only
Code Tier: Tier 5
Format: LLLL-DDDD-LLDD-CHK

- Example: QZRT-1742-KC99-TOP
- Entropy: ~29.5 quadrillion (with checksum)
- Use: Absolute root-level control. The checksum prevents transcription errors and underscores uniqueness.

Notes

- **Format = clearance**. No need to *read* the level, the code's structure conveys it.
- **Prefixes are random**. Prevents inference of scope/project.

- **Hierarchical Access**: Higher levels include lower levels, consistent with SCHEMA manual rules.

Revision Tracking

“Defines a unified, metadata-rich versioning format that distinctly separates a personal project's core version from its associated GitHub repository version. This enables transparent tracking of both internal development progress and public-facing repository updates, without conflating the two.”

Revision Tracking

Format Structure

v[MAJOR].[MINOR].[PATCH].[REPO]-[DATE]-[STAGE]

Components:

Segment	Description
v	Prefix denoting a versioned asset
MAJOR	Core project major version – for fundamental or breaking changes
MINOR	Core project minor version – for feature-level additions
PATCH	Core project patch version – for fixes and minor updates
REPO	**GitHub Repository Version** – 4-digit number incrementing independently
DATE	Release date in YYYY.MM.DD format
STAGE	Development stage: alpha, beta, rc, stable, etc.

Examples:

- v0.0.3.0007-2025.07.21-beta
  → Core version **0.0.3**, GitHub repository version **0007**, released **July 21, 2025**, at the **beta** stage.

- v0.0.0.0001-2025.07.24-alpha
  → Initial version **0.0.0**, GitHub repo version **0001**, released **July 24, 2025**, as an **alpha** build.

Rationale:

- The 3-part core version (MAJOR.MINOR.PATCH) tracks **your personal project's internal evolution.**
- The 4-digit segment (REPO) tracks **the GitHub repository's upload/versioning history**, allowing for commits, releases, or iterations that do not affect the core version.
- The explicit date and stage provide audit-friendly metadata for releases, prototypes, and deployments.

This format is especially suited for dual-tracking workflows, where private development and public deployment diverge in timing or pace.

Archive Management

“Long-term structural system used to organize all digital assets under the Legacy Matrix. Defines how yearly archives are created, labeled, and managed—anchored to the version of SCHEMA active at their time of origin.”

Archive Management

Folder Format Structure:

```
Legacy Matrix/
└── Archive.XXXX+YYYY@SCHEMAvX.X.X/
└── [Projects, Files, and Substructures]
```

Format Breakdown

Segment	Description
Archive.XXXX	**Archive Number** – Chronological index (zero-padded, e.g. 0003)
+YYYY	**Creation Year** – The year the archive was initialized
@SCHEMAvX.X.X	**Starting SCHEMA Version** – The SCHEMA version at archive creation
Legacy Matrix	The root folder containing all annual archives

Archives are immutable in version. Once an archive is created under a specific SCHEMA version, it is never retroactively updated—even if SCHEMA evolves. Each archive is a time capsule of its year and environment.

Example Archives

Folder Name	Meaning
Archive.0001+2025@SCHEMAv0.0.1	Archive 1, created in **2025**, started with **SCHEMA v0.0.1**
Archive.0002+2026@SCHEMAv0.0.6	Archive 2, created in **2026**, started with **SCHEMA v0.0.6**

Legacy Matrix Shorthand Style Guide

To simplify referencing archives across interfaces and systems, **two shorthand formats** are used:

Format Definitions

Compact Format: A324S205

- A = Archive prefix
- 3 = Archive number (from Archive.0003)
- 24 = Year (2024)
- S205 = SCHEMA v2.0.5 → 2 0 5

Use For:

- File/folder names
- Command-line tools
- Internal logs
- QR/barcodes
- Spreadsheet references

Readable Format: ARC3-2024@S2.0.5

- ARC3 = Archive 3
- 2024 = Anchor year
- @S2.0.5 = SCHEMA version, with periods for clarity

Use For:

- Dashboards
- Documentation headers
- Reports and slide decks
- Visual overlays or UI tags

Usage Rules

Context	Format	Example
Folder Names	A324S205	/Projects/A324S205/
CLI / Databases	A324S205	--archive A324S205
UI Labels	ARC3-2024@S2.0.5	Archive Viewer: ARC3-2024@S2.0.5
URLs / Short Links	A324S205	https://legacymatrix.net/a324s205/
Printed Tags / QR Codes	A324S205	*(Compact for physical labels)*
Reports / Headers	ARC3-2024@S2.0.5	LEGACY MATRIX // ARC3-2024@S2.0.5

Naming Convention Reminder

A324S205

- A → Archive
- 3 → Archive.0003
- 24 → Year 2024
- S205 → SCHEMA v2.0.5

ARC3-2024@S2.0.5

- Archive 3
- Year 2024
- SCHEMA version 2.0.5

Document Formatting

"Defines the visual and structural standards for all textual documentation within the SCHEMA system. It establishes baseline formatting for headers, metadata blocks, file identifiers, and layout consistency—ensuring that every SCHEMA-based document is immediately recognizable, accessible, and interoperable across archives and interfaces."

Document Formatting

Purpose

Serves as the **default formatting model** for all documents stored, versioned, or referenced through SCHEMA's identification framework. It encodes metadata visibly within the document body and maintains typographic discipline across headers, sections, and metadata fields.

Features

- Embedded SCHEMA ID block
- Aligned metadata: security level, item type, creation date, organization, and title abbreviation
- Defined header structure (Header 1, Header 2, and normal text)
- Structured for both **print and digital use**

Document Example

Name: SCHEMA: Document Formatting
ID: 0-1-20250630-Schumacher.Personal-SCHEMADF
Date: Started 2025-06-30
Level: 0 - Public