Really Small Web — Conception, Architecture, and MVP Scope

Version 0.02 | November 23, 2026

Developed by Pablo Murad — November 23, 2026 — pablo@pablomurad.com — https://rsw.works

Experimental Project Specification. This document is an experimental specification of the Really Small Web project. It is a work in progress and has not been published as an RFC or approved as an Internet Standard.

License: specifications CC0 1.0 (public domain); future reference implementation and official tools MIT.

Conception, architecture, and MVP scope document

Document version: 0.02

Idealized and developed by Pablo Murad

Pablo Murad — November 23, 2026

Contact: pablo@pablomurad.com

Status: consolidated experimental proposal; implementation not started.

Language: Brazilian Portuguese (original) | Encoding: UTF-8

really small web.

rsw://

Small files. Independent worlds.

A web you can carry, collect, preserve and connect.

Really Small Web (RSW) is a protocol and ecosystem proposal idealized and developed by Pablo Murad.

This document succeeds the original Cartnet conception, numbered 0.01 in the project history (the original file had the 0.1 header). It consolidates the subsequent decisions, delimits the MVP, and records both the pending questions and the deliberately deferred features. It is not a normative specification, nor does it claim the existence of functional software. The command, format, and interface examples are proposals. The version of this document is not necessarily the version of the protocol, of the file format, or of the executable, which will have their own cycles.

Contents

1. Central idea and motivation
2. Manifesto and principles
3. Nomenclature and ecosystem
4. Layered architecture
5. PNG cartridge format
6. Content, limits, and profiles
7. Manifest and metadata
8. Identity, signature, and trust
9. rsw:// addressing and resolution
10. Hosting and publication
11. Local library, navigation, and discovery
12. Versioning, preservation, and mirroring
13. Readers: graphical and terminal
14. Security and threat model
15. Minimalist stack and resource goals

16. CLI, storage, and code organization
17. Communication and future applications
18. Specifications, licenses, and governance
19. Use cases and references
20. MVP: closed scope
21. Acceptance criteria and tests
22. WHAT IS STILL MISSING
23. WHAT WAS LEFT OUT OF THE MVP
24. Roadmap and next steps
25. Final vision

01. Central Idea and Motivation

What if an entire website could be distributed as a PNG image?

Not a screenshot, but the site itself: pages, texts, styles, local resources, identity, metadata, and links.

An ordinary viewer shows the cover image. An RSW reader interprets the additional content and presents a small navigable space. A person can save that file, receive it by email, copy it to a USB drive, host it, or keep it in a library. The basic content works without a server and without an active internet connection.

Cartridges can also reference one another. The network emerges from links, from publication indexes and, in the future, from signed events, messages, and synchronization. The PNGs are passive objects: communication is performed by the clients and services that implement the protocol.

The initial inspiration came from PICO-8, which distributes games in PNG cartridges. RSW combines the culture of file collecting and sharing with autonomy, portability, preservation, and decentralized navigation.

A conventional ZIP would be simpler for storing pages. The choice of PNG does not aim to improve the ZIP's efficiency: it gives the package a visual identity and allows it to be recognized and shared as a digital object, even outside an RSW reader.

The differentiator is not hiding code inside an image. It is the combination of self-contained content, verifiable identity, its own addressing, links between publications, hosting freedom, and offline reading.

02. Manifesto and Principles

- Offline-first: the essential content is in the cartridge.
- No central authority, account, or service is mandatory to read.
- No domain is required to own an RSW identity.
- Any static HTTPS hosting can participate in the network.
- Anyone can create, copy, distribute, and archive cartridges, respecting their licenses and applicable rights.
- Public protocol, documented and implementable by third parties.
- No proprietary platform is a functioning requirement.
- No blockchain, cryptocurrency, or economic token is mandatory.
- No telemetry by default. External connections must be transparent.
- Received content is always untrusted, even when signed.

- The reader's interface is separate from the presentation of cartridges.
- Extensions are optional; small readers may implement only the profiles they want and clearly indicate their limitations.
- Simplicity, low consumption, and accessibility are verifiable goals.
- Compatibility with the conventional web without dependence on a single implementation, hosting, or distribution technology.

Decentralization does not guarantee anonymity, security, availability, or absolute censorship resistance. HTTPS, for example, does not hide from all participants the fact that a file was requested. Preservation requires that someone keep copies, and privacy requires measures of its own.

RSW is inspired by the Small Web, IndieWeb, pubnix, tildeverse, Gemini, Gopher, the demoscene, zines, and cartridge culture. It does not need to compete with the entire conventional web: it intends to be a small-scale alternative, capable of coexisting with it.

03. Nomenclature and Ecosystem

```
RSW / Really Small Web   Project and network name.
RSW Protocol             Open set of specifications.
rsw://                   Proposed URI scheme.
.rsw.png                 Cartridge extension convention.
rsWT                     Proposed private chunk type.
RSW Core                 Central processing library.
rsw                      Unified executable and CLI.
RSW Browser              Graphical reader and local library.
RSW Text                 Markdown content profile.
rswd / rsw serve         Optional network service, in a future stage.
RSW Directory            Independent community directories.
RSW-SPEC                 Set of numbered technical documents.
RSW Tiny                 Proposed seal for cartridges up to 64 KiB.
```

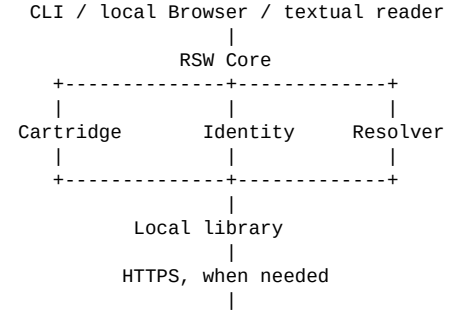
Other future tools: Pack, Inspect, Validator, Forge, Gateway, Explorer, Archive, and SDK. They are feature or project names, not services that need to exist separately from the start.

The research on name, trademark, domain collisions, and URI registration remains pending. The internal choice of RSW does not prove exclusivity.

04. Layered Architecture

```
Layer 1 - Cartridge Layer
    PNG, ZIP, manifest, internal resources, content format.
Layer 2 - Identity Layer
    Keys, signatures, integrity, versions, and stable identity.
Layer 3 - Transport Layer
    Local files and HTTPS; additional optional transports.
Layer 4 - Network Layer
    Resolution, links, indexes, discovery, mirrors, synchronization.
Layer 5 - Social Layer
    Events, comments, followers, messages, and moderation.
Layer 6 - Application Layer
    Application runtime, isolated storage, and capabilities API.
```

Layers 1 to 4, in the subset of local files, HTTPS, indexes, and links, are sufficient for the MVP. Layers 5 and 6 are future.



```
        Independent static hostings
```

The core does not depend on the graphical interface. A third party can implement a client in Python, Rust, C, Lua, or another language, as long as it follows the corresponding specifications. It does not need to use the official core.

05. PNG Cartridge Format

Two techniques were considered in the original conception:

A. Private ancillary chunks: binary data inside the PNG structure. B. LSB: data encoded in the least significant bits of the pixels.

PICO-8 uses the second approach: two bits in each RGBA channel, eight bits per pixel. Its 160 x 205 pixel image offers 32,800 raw bytes, subject to PICO-8's own internal layout and limits.

RSW chooses the first approach for the MVP. It does not need to alter the pixels or set fixed dimensions. LSB remains a future experiment.

Illustrative structure:

```
midnight.rsw.png
    PNG signature
    IHDR
    [other valid PNG chunks]
    IDAT (compressed cover pixels)
    rsWT (bytes of a ZIP file)
    IEND
```

The rsWT chunk was chosen to follow the PNG naming convention: first character lowercase, ancillary; second lowercase, private; third uppercase, reserved bit; fourth uppercase, not safe for indiscriminate copying by editors that do not know the chunk when critical chunks are modified. Its exact location, cardinality, and validation rules will be defined in RSW-0001. Proposal: a single rsWT chunk after IDAT and before IEND; reject duplicates.

Proposed internal structure:

```
payload.zip
|-- manifest.json
|-- signature.sig        (when signed)
|-- index.md             (or index.html)
|-- style.css            (optional)
|-- pages/
|-- assets/
|   |-- logo.png
|   `-- photo.jpg
`-- data/
    `-- posts.json
```

The packer preserves the cover image, adding the chunk and the corresponding CRC. An ordinary PNG reader can ignore the ancillary chunk. The RSW reader extracts its content and validates each component.

The PNG structural limit of almost 2 GiB per chunk is not an RSW acceptance limit. The implementation must reject excessive entries before allocating memory or decompressing files. The PNG CRC is useful for structural integrity, but it is not a cryptographic signature.

One must not depend on data appended after IEND, on polyglot files, or on automatic code execution. The file must remain a valid PNG according to the applicable specification.

Social networks, converters, editors, and CDNs may strip the chunk or recompress the image. A cover that remains visible may have lost its cartridge. Sharing must preserve the original bytes; the reader must distinguish an ordinary PNG from an RSW PNG. An exported ZIP can serve as a content backup, not as a substitute for the authenticated cover.

06. Content, Limits, and Profiles

Consolidated project limits (engineering values to be formalized):

```
Tiny cartridge:                  up to 64 KiB
Recommended cartridge:           up to 256 KiB
Maximum complete cartridge MVP:  2 MiB (2,097,152 bytes)
Enlarged limit in a future stage: 10 MiB, if approved
Maximum expanded ZIP:            32 MiB
Maximum number of entries:       512
Maximum manifest:                16 KiB
Text encoding:                   UTF-8
```

The 10 MiB discussed earlier do not constitute a second accepted limit in the MVP: the MVP operational limit is 2 MiB for the complete PNG, including cover, ZIP, and overhead. Entry counts, cover dimensions/pixels, per-file limits, depth, internal image resources, and processing costs still need to be defined. Resource goals do not replace the format's mandatory checks.

Compatibility profiles:

```
RSW Text        Markdown and links, without arbitrary active HTML.
RSW Static      HTML and CSS, images and other allowed local resources.
RSW Interactive Isolated JavaScript and own storage (future).
RSW Connected   RSW API and reader-mediated communication (future).
```

MVP: Text and Static. The selection of a safe subset of Markdown and HTML/CSS, MIME types, and relative paths still needs to be formalized. No script, automatic remote resource, or active form is accepted as an MVP execution feature. The reader's interface may use its own JavaScript; this does not enable JavaScript from cartridges.

Semantic types, different from the execution profiles: site, page, profile, journal, zine, gallery, book, directory, webring, feed, data, app, game, archive. Initially they are metadata; the existence of an app or game type does not grant permission to execute code. Special rules for each type may be added only when necessary.

All content needed for reading must be in the package. An external link may exist, but it will be treated as explicit navigation, not an invisible dependency. Missing resources must be flagged without automatically fetching arbitrary URLs.

07. Manifest and Metadata

The manifest identifies the cartridge, its contents, and connections. Illustrative example (hash and key values are placeholders, not valid data):

```
{
  "protocol": "rsw/0.01",
  "name": "midnight-journal",
  "title": "The Midnight Journal",
  "description": "A little corner of the network.",
  "type": "journal",
  "profile": "static",
  "version": 3,
  "author": "pablo",
  "public_key": "ed25519:BASE32_DA_CHAVE",
  "entrypoint": "index.html",
  "created_at": "2026-11-23T00:00:00Z",
  "license": "CC-BY-4.0",
  "previous_release": "sha256:HASH_ANTERIOR",
  "files": {
    "index.html": "sha256:HASH_DO_HTML",
    "style.css": "sha256:HASH_DO_CSS"
  },
  "peers": [
    "rsw://alice.example/zine",
    "rsw://bob.example/profile"
  ],
  "capabilities": ["discovery", "updates"],
  "permissions": [],
  "tags": ["journal", "smallweb", "photography"]
}
```

The rsw/0.01 string is provisional, not a published protocol version. RSW-0002 will fix mandatory and optional fields, types, limits, names, extensibility, and the behavior toward unknown fields. It will also define whether the file list will be an object or an array and how to treat duplicate manifests, equivalent paths, and extra metadata.

Capabilities describe features; permissions describe requested accesses. No manifest field grants access by itself. In the MVP there is no script-permission API; incompatible requests must be ignored with a warning or rejected according to the final specification.

The manifest and the signature cannot list their own hash circularly. The list must cover all relevant resources, excluding the reserved files per an explicit rule. Extra or unauthenticated files must not be presented as signed content.

08. Identity, Signature, and Trust

Consolidated proposal:

```
Identity: Ed25519.
Hash: SHA-256.
Textual public keys: Base32, details to be fixed.
Manifest: UTF-8 JSON.
Candidate canonicalization: JCS (RFC 8785).
Separate signature: signature.sig.
```

One key per author is sufficient for the first experiment. The friendly name does not prove authorship. The stable identifier of a publication is formed by the author's key and the cartridge's name, independent of the hosting domain. Cartridges with equal names and distinct keys are independent publications.

The signed manifest must contain the file hashes and bind the identity to the content. The exact procedure, signature domain prefix, bytes, signature representation, Base32 encoding, and error handling belong to RSW-0003. Do not improvise canonicalization with json.Marshal or with simple key sorting: interoperability requires exact rules and test vectors. If JCS requires a small, auditable dependency, the zero-dependency goal will not take priority over cryptographic correctness.

Distinguish three references: the original PNG hash, the raw ZIP hash, and the file hashes. For the MVP, it is proposed that rsw://sha256/<hash> points to the SHA-256 of the complete PNG, computed externally. The ZIP hash may be used for inspection, but not as a synonym of the image hash. This choice must be frozen in the specification. To authenticate the cover as part of the authorship, define a separate commitment that does not include the signature itself circularly; until then, do not claim that the content signature authenticates the pixels or external metadata.

Trust states:

```
TRUSTED   Valid signature and key locally trusted by the user.
VERIFIED  Valid signature, unknown or not yet trusted key.
UNSIGNED  No signature; local import and reading with a warning.
INVALID   Incorrect signature or integrity; reject, do not render.
```

Authenticated publication on the network requires a signature. Anonymous local cartridges are allowed, but they cannot update a signed project nor occupy another person's key address. A valid signature is not synonymous with an identified real person, a safe site, or benign content.

Private keys stay outside the PNG, with restricted permissions and user-controlled backups. An offline recovery key, rotation, and revocation are future goals; the MVP must at least document backup, key loss, and compromise risk. There is no magical recovery nor universal revocation of already-copied content.

09. rsw:// Addressing and Resolution

Three proposed modalities:

```
rsw://pablomurad.com/journal
rsw://key/&lt;public-key&gt;/journal
rsw://sha256/&lt;hash&gt;
```

The first locates a project by a domain. The second identifies a publication by key and name. The third is a reference to an exact, immutable file. A historical version could be selected by rsw://pablomurad.com/journal?v=1, syntax not yet normative.

The browser interprets the rsw:// scheme internally. It does not need to be recognized by common browsers nor require a new network port. The URI grammar, normalization, characters, IDNA, escapes, ports, segments, fragments, and the reserved key and sha256 namespaces must be fixed to avoid ambiguities with domain names. Relative internal links need clear resolution rules inside the cartridge.

Proposed resolver flow:

1. Interpret and validate the address.
2. Consult the local library and known records.
3. For the domain mode, fetch the HTTPS index if the policy allows.
4. For key/hash, consult local maps and previously known sources, such as indexes, peer links, or authorized mirrors.
5. Validate the download URL, limits, and redirects.
6. Download the file, check the expected hash, manifest, and signature.
7. Record the location and store the original PNG.
8. Open the content in an isolated environment.

In offline mode, never make connections. If the cartridge is not stored, indicate so. For key or hash references still unknown, there is no guaranteed global discovery: the user can import a copy or provide a source. The hash contains no location.

The MVP needs no own DNS, DHT, relay, or naming authority. Provisional scheme registration per RFC 7595 may be evaluated later; it is not a prototype prerequisite.

10. Hosting and Publication

Any common HTTPS hosting suffices, for example Nginx, Caddy, Apache, or a static hosting. No dedicated RSW server is mandatory.

```
example.org/
|-- .well-known/
|   `-- rsw.json
`-- rsw/
    |-- profile.rsw.png
    |-- journal.rsw.png
    `-- zine.rsw.png
```

The /.well-known/rsw.json index is an experimental convention not yet registered; rules and any registration will be considered in RSW-0006. Conceptual index example:

```
{
  "protocol": "rsw/0.01",
  "name": "pablo",
  "public_key": "ed25519:BASE32_DA_CHAVE",
  "cartridges": [
    {
      "name": "journal",
      "version": 3,
      "url": "/rsw/journal.rsw.png",
      "sha256": "HASH_DO_PNG"
    }
  ]
}
```

The authenticated index must have a separate signature or a signed envelope per the defined rule, including version, hash, identity, and publication location. HTTPS authenticates the domain according to the TLS infrastructure; it does not replace the author's signature. An index with multiple authors does not automatically acquire authorization to update their projects.

Author flow:

```
create local content -&gt; select cover -&gt; generate manifest -&gt;
```

```
compute hashes -&gt; sign -&gt; create ZIP -&gt; insert rsWT -&gt; validate -&gt;
publish original PNG -&gt; generate/publish signed index -&gt; share.
```

Reader flow:

```
open file or address -&gt; verify -&gt; import -&gt; present -&gt;
follow links by explicit action -&gt; check updates manually.
```

Serve the original file as image/png, with X-Content-Type-Options: nosniff; offer download preserving the bytes. Do not depend on automatic image processors. Publishing third-party content requires separation from authenticated-services origin and an abuse policy.

11. Local Library, Navigation, and Discovery

The library is the base of the offline experience. It stores the exact PNG and separate metadata: key, name, version, hash, origin, import date, trust, references, and preferences. A cartridge gains no access to the global catalog or to other cartridges' data.

Initial discovery is limited to links, explicit peers, and webrings. A publication can reference friends, directories, archives, zines, and other content by the same author. The client presents these links and allows opening what is cached or fetching the reference on the network.

The first network already exists when three independent publications reference each other. It needs no chat, comments, or real-time messages. Discovering a link does not mean downloading or executing everything it references. Recursive downloads stay disabled in the MVP.

Community indexes, feeds, favorites, history, and publication subscriptions can extend the experience later. The MVP needs only a library, listing, import, export, manual update checks, and navigation between cartridges.

12. Versioning, Preservation, and Mirroring

A published version is an immutable object identified by hash. Updates are new complete PNGs, not changes to a signed copy. The library can preserve the versions it imports; permanence depends on the storage and the participants' will.

Each new version can reference the previous hash (previous_release). A signed index binds the stable name to the most recent known version. The client records the largest authenticated version already observed, avoids automatic regressions, and signals conflicts, such as two versions with the same number and distinct hashes. The specification must fix the exact treatment of these cases and of a first query that receives old data.

A version cannot be considered current merely because it has the author's greatest declared date: clocks can be wrong. The version counter and chaining need their own rules. There is no global consensus, blockchain, or universal proof that an announcement is the most recent. An offline reader only reports what it knows.

Seed Library (future): a participant may choose public cartridges they want to make available on their server. These mirrors do not alter the files, and the recipient verifies the original hash and signature. The original server can disappear while other copies continue to exist. Sharing must be opt-in and respect licenses, privacy, local policies, and applicable removal requests.

In the MVP, manual copies and import/export already allow preservation. Automatic replication, deltas, advanced historical catalogs, and global mirror resolution are out of the initial scope.

13. Readers: Graphical and Terminal

RSW Browser: a local web interface embedded in the Go executable. The rsw browse command starts a loopback service and opens the conventional browser. Do not develop an own rendering engine or depend on Electron.

```
+------------------------------------------------------------+
| &lt;  &gt;  Update | rsw://example.org/midnight-journal          |
```

```
+-----------------+-----------------------------------------+
| LIBRARY         | THE MIDNIGHT JOURNAL                    |
| Journal         | Cover, text, pages and links            |
| Zines           | Author: verified key                    |
| Archives        | Version: 3 | Size: 42 KiB               |
| Import PNG      | Connections: Alice, Bob                 |
+-----------------+-----------------------------------------+
| Offline ready | 3 local cartridges                      |
+---------------------------------------------------------+
```

Essential functions: open/import a PNG, open a URI, display cover and data, verify the signature, list the library, navigate pages and references, check for updates manually, export the original, and activate offline mode. The privileged interface is not part of the imported content. Drag-and-drop, favorites, tags, detailed history, and version visualization are possible improvements, not conditions for the first demonstration.

Textual reader: especially useful for SSH, pubnix, tildes, old computers, and environments without a graphical interface. RSW Text will present Markdown and a link list without an HTML engine. The inspection CLI is part of the MVP; simple textual reading is desirable, but a full TUI inspired by Lynx or Offpunk is a post-MVP extension.

Conceptual examples:

```
rsw read rsw://example.org/journal
rsw links
rsw library
rsw offline
```

In the MVP, rsw:// works inside the reader. Operating-system registration, browser extensions, a Tauri app, and mobile clients may be evaluated later. Do not confuse the reader's address bar with Chrome's or Firefox's address bar.

14. Security and Threat Model

The PNG is passive. JavaScript inside a chunk is not executed by an ordinary viewer just by existing. A malformed file can, however, exploit image-library vulnerabilities or cause excessive consumption. A custom reader can also create risks when it interprets or executes content insecurely.

The MVP does not execute third-party code. A signature does not eliminate the need for isolation. The reader treats images, ZIPs, manifests, HTML, CSS, Markdown, and network data as untrusted inputs.

Threat	Required or proposed protection
Malformed PNG, invalid CRC	Strict parser, limits, safe rejection.
Excessive decompression	Count real bytes and limit resources.
Path traversal	Validated paths; nothing outside the root.
Symlinks and duplicates	Reject special entries and collisions.
Tampering	Verifiable hashes and signatures.
Active HTML, XSS, scripts	Sanitization, sandbox, CSP, separation.
External tracking	Block remote subresources by default.
SSRF and redirects	Destination restrictions and explicit policy.
DNS rebinding	Validate Host, origin, and network destinations.
CSRF on the local API	Session token, Origin, safe methods.
Replay and downgrade	Sequence verification and known hash.
Version conflicts	Warning and deterministic policy.
Key theft	Separate storage, restricted access.
Resource abuse	Time, memory, disk, and rate limits.
Spam and unwanted content	Local moderation and filters (future network).

Rules for package processing:

- Validate declared sizes before allocating; enforce limits during reading and decompression, without trusting only the ZIP header.
- Reject absolute names, .. segments, dangerous characters and separators, duplicate entries, symlinks, device files, and name collisions after normalization, including on Windows systems.
- Do not extract using paths provided directly by the cartridge. Prefer controlled access to internal resources or a dedicated temporary directory, with explicit export and no file overwriting.

- Verify the presence of the manifest, entrypoint, version, supported types, hashes, and signature before rendering. Fail closed.
- Also limit internal image dimensions and decoding cost, ZIP depth, nested files, and local disk occupation.
- Unknown files are not executed or interpreted as HTML.

Rules for the reader and the local interface:

- Bind the interface server to loopback by default, with Host and Origin verification and protection of privileged operations. Loopback alone is not a security boundary.
- Isolate imported content from the privileged interface, with a separate origin or an origin-isolated sandbox iframe without scripts and without allow-same-origin; do not give access to the reader's API or keys.
- Apply a restrictive CSP and an allowlist of types and resources. Block scripts, inline events, forms, iframes, plugins, automatic navigation, automatic downloads, and external subresources.
- HTML and CSS can also create connections and navigations; do not trust only JavaScript being disabled. Treat links in a mediated way.
- Define a policy for SVG, CSS url()/@import, meta refresh, external links, fonts, and images. Active SVG cannot be treated as inert without appropriate measures; when in doubt, exclude it from the initial profile.
- Do not expose terminal APIs, personal files, native IPC, or secrets.
- No destructive action, publication, or sensitive connection started by a cartridge without authorized user intervention.

Rules for the HTTPS resolver:

- Use HTTPS and normal certificate validation; do not silently accept TLS errors. Define redirects, timeouts, and maximum size.
- Do not reach private networks, loopback, link-local, metadata services, or other destinations reserved at the request of remote content. Local instances and pubnix may require an explicit, controlled test mode; it must not disable the protection globally.
- Validate destinations after DNS resolution and redirects, preventing initial checks from being bypassed by address changes.
- Offline mode blocks RSW-generated traffic, including updates.

This document does not guarantee that a sandbox is sufficient. Isolation must be designed and tested on supported browsers and platforms. Before a public version, perform a security review, fuzzing, negative file tests, and rendered-content escape tests. No reader should promise that every malicious file will be detected.

15. Minimalist Stack and Resource Goals

Decision of this version: Go for the core, CLI, and local server. HTML, CSS, and vanilla JavaScript for the interface. Filesystem + JSON initially. SQLite only when storage complexity justifies it.

Component	Technology
Language and binary	Go
HTTP and HTTPS	net/http, crypto/tls
CLI	flag and simple subcommands
ZIP	archive/zip
PNG	image/png + dedicated chunk parser
Binaries and CRC	encoding/binary, hash/crc32
JSON	encoding/json + JCS per RFC 8785
Hash	crypto/sha256
Signature	crypto/ed25519, crypto/rand
Embedded interface	embed, html/template
Frontend	HTML, CSS, vanilla JavaScript
Markdown	defined subset or small library
Storage	files and JSON indexes

```
Tests                           testing, httptest, Go fuzzing
Public hosting                  any static HTTPS server
Proposed license                MIT (code), CC0 (specifications)
```

Go allows reusing standard libraries for most operations. The image/png package does not provide a general API to preserve private chunks; the core needs an appropriate parser/serializer. Do not invent an insecure crypto parser, HTML sanitizer, or JCS implementation just to reach zero dependencies. Use small, auditable, pinned libraries when needed for correctness and security.

Initial goals, not yet measured:

```
CLI + local reader executable:   up to 10 MiB, when feasible.
Idle local service RAM:          up to 32 MiB, excluding browser.
Core external dependencies:      zero, if technically viable.
Typical cartridge:               up to 256 KiB.
Tiny cartridge:                  up to 64 KiB.
Mandatory containers:            zero.
```

The public version must record platform, compilation flags, workload, and benchmark conditions. Binary size and real consumption depend on implementation decisions. They are goals, not promises.

Intended initial platforms: Linux, Windows, and macOS. FreeBSD and OpenBSD are desirable possibilities, mainly for pubnix. Effective availability requires per-platform builds and tests.

Out of the initial stack: React, Vue, Next.js, Electron, Node.js as a mandatory runtime, PostgreSQL, MongoDB, Redis, Kubernetes, blockchain, microservices, and complex queues. Rust/Tauri remains a future desktop client option, without replacing the Go decision for the MVP. Docker may be offered for convenience, never mandatory.

16. CLI, Storage, and Code Organization

A single rsw executable with subcommands. Examples not yet implemented:

```
rsw init midnight-journal
rsw keygen
rsw pack ./midnight-journal --cover cover.png --out journal.rsw.png
rsw inspect journal.rsw.png
rsw verify journal.rsw.png
rsw unpack journal.rsw.png --out ./extracted
rsw import journal.rsw.png
rsw fetch rsw://example.org/journal
rsw open rsw://example.org/journal
rsw library
rsw browse
rsw update journal
```

Generating a publication index must be possible via the CLI or an equivalent tool; the name and arguments are yet to be defined. rsw serve, rsw mirror, rsw sync, and global search are planned for later.

Proposed repository organization:

```
really-small-web/
|-- cmd/rsw/main.go
|-- internal/
|   |-- cartridge/
|   |-- manifest/
|   |-- identity/
|   |-- resolver/
|   |-- transport/
|   |-- storage/
|   |-- discovery/
|   |-- security/
|   `-- server/
|-- web/
|   |-- index.html
|   |-- style.css
|   `-- app.js
|-- spec/
|-- examples/
|   |-- journal/
|   |-- zine/
|   `-- webring/
|-- testdata/
```

```
|-- go.mod
|-- Makefile
|-- README.md
`-- LICENSE
```

Do not create empty packages in anticipation. The structure must grow as responsibilities appear. Keep the core independent of the CLI and the interface, with testable functions and documented errors.

Suggested local storage (platform-dependent path):

```
rsw-data/
|-- objects/sha256/&lt;hash&gt;.rsw.png
|-- indexes/library.json
|-- indexes/locations.json
|-- config.json
`-- keys/                (restricted permissions; never served)
```

Write objects and indexes atomically where the system allows; protect against concurrent writes; do not accept arbitrary file paths through the local interface. Hash-addressed storage facilitates deduplication and version preservation. Define a cache-cleaning policy, disk limits, and key backup separately.

17. Communication and Future Applications

The network is initially formed by connections between cartridges. Active communication can be added without rewriting the format or altering the original PNG. Each occurrence would be a separate, signed event referenced to the identity or to a cartridge's hash.

Conceptual operations (not normative verbs or implemented commands):

```
DISCOVER   Discover identities and cartridges.
FETCH      Obtain an identified file.
HELLO      Present an identity/session.
ANNOUNCE   Announce a publication or new version.
FOLLOW     Follow an author or project.
MESSAGE    Send a message.
COMMENT    Comment on a publication.
SYNC       Synchronize information.
VERIFY     Verify locally.
```

Future event types: announce, update, follow, message, comment, reference, mirror, reaction, revoke. Proposed message envelope:

```
{
  "protocol": "rsw/0.01",
  "type": "message",
  "id": "HASH_DO_EVENTO",
  "from": "CHAVE_DO_REMETENTE",
  "to": "CHAVE_DO_DESTINATARIO",
  "created_at": "2026-11-23T18:40:00Z",
  "body": "hello from another cartridge",
  "signature": "ASSINATURA"
}
```

This example describes a conceptual message, not a secure private message. It is necessary to define which fields enter the hash and signature, excluding circular fields, and to implement deduplication, replay, limits, expiration, moderation, transport, synchronization, and privacy. Private messages need end-to-end encryption with an established, auditable construction, not an improvised cryptographic mechanism.

A future API, mediated by the reader, could offer:

```
rsw.identity.get()
rsw.cartridge.info()
rsw.network.discover()
rsw.network.follow(author)
rsw.network.send(recipient, message)
rsw.storage.get(key)
rsw.storage.set(key, value)
rsw.events.subscribe("message")
```

Candidate permissions:

```
render.static       HTML/CSS and internal resources.
script.execute      Isolated JavaScript.
```

```
storage.local       The cartridge's own storage.
network.discover    Authorized discovery.
network.message     Authorized communication.
network.external    Limited external connections.
```

The application layer requires explicit authorization, origin and storage isolation, CPU/memory limits, permission revocation, and its own review. Never expose private keys directly. A signing API, if necessary, must be restricted and require consent. The reader's JavaScript is not a cartridge's JavaScript.

Optional servers may exchange events over HTTPS or WebSocket. P2P, DHT, and alternative transports are extensions, not network assumptions. The event schema will not be defined before stabilizing the static format, identity, and resolver.

18. Specifications, Licenses, and Governance

Normative documentation must be independent of the software. Numbered proposals, inspired by Nostr's NIPs, allow community evolution. Provisional document map:

```
RSW-0001  PNG cartridge format
RSW-0002  Manifest and metadata
RSW-0003  Identity and signatures
RSW-0004  rsw:// URI scheme
RSW-0005  Address resolution
RSW-0006  Static hosting
RSW-0007  Discovery, links, and directories
RSW-0008  Network events                    (future)
RSW-0009  Messages and comments             (future)
RSW-0010  Replication and mirroring         (future)
RSW-0011  Security and permissions
RSW-0012  JavaScript API                    (future)
RSW-0013  Updates and versioning
RSW-0014  Message encryption                (future)
RSW-0015  Key rotation and revocation       (future)
RSW-0016  Compatibility and extensions
```

RSW-0011 and the basic subset of RSW-0013 must accompany the first specifications, despite their later numbers. A future extension cannot make a central service mandatory for the static core.

Project documents:

```
MANIFESTO.md, SPECIFICATION.md, GOVERNANCE.md, SECURITY.md,
CONTRIBUTING.md, COMPATIBILITY.md, ROADMAP.md, LICENSE.
```

Proposed licenses: MIT for the reference implementation; CC0 for specifications and, at the authors' choice, test examples. Cartridge licenses are independent and must be respected.

[Editorial note — documentation release: these licenses were formalized by the creator of the project. The specifications (RSW-0001 to RSW-0016) are dedicated to the public domain under CC0 1.0, and the MIT license is recorded for the future reference implementation and official tools. See the Governance page and the CC0-1.0.txt / MIT-LICENSE.txt files in the downloads area. The text above is preserved as the historical v0.02 wording.]

Simple initial governance: public repository, reviewable changes, numbered proposals, a state per proposal (draft, experimental, stable, obsolete), and a compatibility policy. Do not promise stable standards before interoperability and version-rule review.

19. Use Cases and References

Use cases:

- Portable personal blogs and journals.
- Profiles with biography, links, and public keys (no private keys).
- Zines, magazines, books, and small collectible catalogs.
- Galleries of art, photography, and ANSI/ASCII archives with SAUCE data.

- Directories, webrings, and libraries for pubnix/tilde communities.
- Postcards with separately encrypted letters (future).
- Applications, games, forums, and restricted messengers (future).
- Preservation of websites and shareable historical versions.

Suggested initial experiment: three authors and three hostings, a blog, a zine, and a directory. Projects such as noctem.cafe and ASCII.zone could serve as real use contexts, without becoming RSW dependencies.

Technical references and inspirations (study list, not a compatibility audit or performed implementation):

```
PNG Specification - https://www.w3.org/TR/png/
    Container and chunk structure.
PICO-8 - https://www.lexaloffle.com/pico-8.php
    PNG cartridge distribution culture.
Excalidraw - https://excalidraw.com/
    Exportable images that can carry editable data.
TiddlyWiki - https://tiddlywiki.com/
    Self-contained documents and applications.
Go standard library - https://pkg.go.dev/std
    Base of the proposed implementation.
RFC 8032 - https://www.rfc-editor.org/rfc/rfc8032.html
    EdDSA, including Ed25519.
RFC 8785 - https://www.rfc-editor.org/rfc/rfc8785.html
    JSON Canonicalization Scheme.
RFC 3986 - https://www.rfc-editor.org/rfc/rfc3986.html
    Generic URI syntax.
RFC 7595 - https://www.rfc-editor.org/rfc/rfc7595.html
    URI scheme registration guidelines.
RFC 8615 - https://www.rfc-editor.org/rfc/rfc8615.html
    well-known URIs.
MDN iframe - https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe
    Rendering isolation and sandbox.
MDN CSP - https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
    Content security policies.
Nostr - https://github.com/nostr-protocol/nips
    Identities and events signed over relays. Nostr uses Schnorr
    on secp256k1, not Ed25519. It requires adaptation or a verifiable binding.
Blossom - https://github.com/hzrd149/blossom
    Optional blob storage by hash.
IPFS - https://ipfs.tech/
IPNS - https://docs.ipfs.tech/concepts/ipns/
    Content-addressed distribution and mutable references.
Hypercore - https://github.com/holepunchto/hypercore
    Verifiable P2P logs and replication.
IndieWeb - https://indieweb.org/
    Publication autonomy and content ownership.
Gemini - https://geminiprotocol.net/
    Inspiration from small-scale protocols and clients.
```

None of these projects is a mandatory dependency of the RSW network.

20. MVP: Closed Scope

Objective: demonstrate a small portable, signed, interconnected, and offline-navigable web, without a central server or third-party runtime.

INSIDE THE MVP (mandatory):

- Initial specification of format, manifest, and identity.
- Conventional PNG with rsWT chunk and internal ZIP; 2 MiB limit.
- RSW Text and Static: Markdown and HTML/CSS with safe local resources.
- Packaging, extraction, inspection, validation, and signing.
- Local Ed25519 key; SHA-256 hashes; local trust.
- Unsigned cartridge for local import, with a clear indication.

- rsw:// resolver by domain, key, and known hash, without a mandatory global search for unknown addresses.
- Static HTTPS hosting and a signed publication index.
- Library of original PNGs with local storage and offline mode.
- Import, export, and sharing of the original files.
- Local web reader with isolated interface and imported content.
- Navigation through links, peers, and webrings; manual update checks.
- Complete versions, update verification, and automatic-regression prevention per a documented minimal policy.
- Installation and usage documentation; fixtures and security tests.
- Three cartridges from distinct identities and hostings.

The inspection and administration CLI is mandatory. Minimal textual reading is desirable; a full TUI is not a blocking criterion. It is not necessary to develop a public daemon, gateway, or native app.

The first network is not a social network. The existence of resolvable links among three independent authors already demonstrates communication in the sense of discovery and navigation. Active events constitute the next stage.

21. Acceptance Criteria and Tests

Completion checklist (all still pending implementation/testing):

- [ ] Create and protect a local cryptographic identity.
- [ ] Pack a small Markdown or HTML/CSS site inside a PNG.
- [ ] Open the cover normally in conventional PNG viewers.
- [ ] Extract and validate the content without altering the original PNG.
- [ ] Verify authorship and hashes; distinguish unsigned, invalid, and verified.
- [ ] Publish PNG and signed index on a common HTTPS hosting.
- [ ] Resolve rsw:// addresses in the MVP's modalities.
- [ ] Navigate among three independent cartridges and their internal links.
- [ ] Import and export byte-identical copies.
- [ ] Disconnect the network and navigate all imported cartridges.
- [ ] Take one hosting down and open the corresponding stored copy.
- [ ] Check a signed update, detect regression and conflict.
- [ ] Reject invalid entries and test known threat classes.
- [ ] Validate limits and the absence of unauthorized external requests.
- [ ] Publish fixtures and expected results for interoperability.
- [ ] Document how to create, verify, and publish a cartridge.

Suggested minimal fixtures:

```
valid-minimal.rsw.png
valid-text.rsw.png
valid-signed.rsw.png
valid-unsigned.rsw.png
valid-multilingual.rsw.png
```

```
valid-webring.rsw.png
invalid-signature.rsw.png
invalid-manifest.rsw.png
invalid-checksum.rsw.png
invalid-path.rsw.png
invalid-duplicate-chunk.rsw.png
invalid-extra-file.rsw.png
invalid-size.rsw.png
invalid-zip.rsw.png
unsupported-version.rsw.png
expected-results.json
```

Also create canonicalized-manifest, key, signature, hash, URI, and index vectors. Use exclusively test keys. Include fixtures of conflicting updates, downgrades, and malformed network responses. Exercise PNG/ZIP parser, manifest, and resolver fuzzing.

Interoperability test: an independent implementation or a small reference verifier in another language must be able to validate the vectors. The official library passing its own tests is not, alone, proof of interoperability.

Definition of done: the mandatory criteria pass, the MVP rules are published, there are no known untreated critical failures, the limitations are documented, and the demonstration works without the maintainer's private services. This is not equivalent to a formal security audit nor a guarantee of vulnerability absence.

22. What Is Still Missing

The MVP conception phase is closed. We do not need new features to start. Missing are normative decisions, implementation, and validation. The proposals below are not implemented and must not be confused with completed tasks.

P0 - SPECIFICATION BLOCKERS (before or during the first prototype)

- [ ] RSW-0001: define the rsWT chunk bytes, position, cardinality, CRC, ZIP structure, allowed compression, PNG validation, absence of data after IEND, treatment of foreign files, and errors.
- [ ] Fix size, dimension, and pixel limits, decompression, entries, names, time, and memory; exact-limit tests.
- [ ] RSW-0002: manifest schema, mandatory/optional fields, profile/type values, UTF-8, paths, entrypoint, tags, extensions, duplicate fields, number types, and behavior toward new fields.
- [ ] RSW-0003: exact signed bytes, signing domain, JCS, Base32, signature.sig representation, file hashes, exclusions, and treatment of unlisted resources. Decide cover authentication.
- [ ] Fix the semantics of the PNG SHA-256 and internal hashes, previous references, and independent verification vectors.
- [ ] RSW-0004: rsw:// grammar, domains, reserved namespaces, paths, fragments, query/version, normalization, and internal-link rules.
- [ ] RSW-0005 and 0006: resolution algorithm, cache priority, known sources, signed indexes, domain/author binding, TLS policy, HTTP, relative URLs, redirects, and error handling.
- [ ] RSW-0007: minimal peers and webrings convention and manual discovery.
- [ ] RSW-0011: threat model, safe Markdown and HTML/CSS profiles, isolation architecture, network policy, local API, files, keys, malicious inputs, and rejection rules.
- [ ] RSW-0013: minimal versioning, monotonicity, conflicts, first queries, old versions, and absence of newness information.
- [ ] RSW-0016: version negotiation, field extension, and compatibility.
- [ ] Decide the real initial protocol/format version, separate from v0.02 (conception-document version) and from the program version.

P1 - IMPLEMENTATION (not started)

- [ ] Create repository, licenses, philosophical manifesto, documentation, and CI.
- [ ] Implement PNG, ZIP, and manifest parser/serializer with limits.
- [ ] Implement identity, signatures, and verification tools.
- [ ] Implement CLI, hash-addressed objects, and persistent local catalog.
- [ ] Implement resolver, index generation, and secure HTTPS client.
- [ ] Implement local reader, static navigation, and offline mode.
- [ ] Implement unit, integration, and fuzzing tests and builds.
- [ ] Create three independent demonstration cartridges and hostings.

P2 - VALIDATION AND PUBLICATION PREPARATION

- [ ] Measure executable size, RAM, opening time, disk use, and performance on low-resource machines; adjust the budget.
- [ ] Test reader security and isolation on supported browsers, redirects, limits, concurrency, and failure recovery.
- [ ] Publish fixtures, verify interoperability, and record failures.
- [ ] Document key backup, import/export, and limitations.
- [ ] Test Linux, Windows, and macOS and prepare verifiable executables.
- [ ] Verify name, trademark, domain, and repository availability before public release; study URI/well-known registration.
- [ ] Define the contribution process and responsible disclosure.

Non-blocking questions that remain open for the long term: identity recovery/rotation, revocation, event privacy, moderation, global discovery, community governance, new platform support, and replication policy. They must not enlarge the MVP.

23. What Was Left Out of the MVP

Deliberately excluded from the first implementation, and why:

- JavaScript from cartridges, WebAssembly, plugins, and application runtimes: they need isolation, an execution budget, and their own review.
- rsw.* API and interactive permissions: they depend on the safe runtime.
- Chat, messages, comments, reactions, and distributed followers: they require an event protocol, moderation, and additional synchronization.
- End-to-end private messages: key management and privacy need an independent cryptographic specification.
- Relays, federation, public rswd, and WebSocket transport: the MVP uses static HTTPS, without a mandatory dedicated service.
- P2P, DHT, libp2p, IPFS/IPNS, and Hypercore: unnecessary complexity to prove the first network; possible optional transports.
- Nostr and Blossom integration: possible future adaptation, not a requirement.
- Automatic Seed Library, replication, network-discovered mirrors, block/delta synchronization: manual copies suffice initially.
- Global directories, indexers, distributed search, automatic backlinks, and advanced feeds: links and webrings prove discovery.

- Public gateway, visual Forge creator, Explorer, and independent SDK: adoption tools that can reuse the stabilized core.
- Tauri desktop app, mobile clients, extensions, and rsw:// operating-system integration: use the conventional browser.
- Advanced TUI, complete visual history, tags, favorites, and sophisticated historical-files interface: experience improvements.
- Mandatory SQLite, database services, and containers: use the filesystem.
- Key rotation, distributed revocation, recovery key, and Web of Trust: one key and local trust suffice for the prototype.
- LSB encoding in pixels and alternative image formats: keep a single transport mechanism for interoperability.
- Cartridges above 2 MiB: test the Really Small Web goal before considering larger profiles (such as the future 10 MiB limit).

Not planned as project requirements at any phase: blockchain, cryptocurrency, mandatory central account, advertising, or telemetry by default. Future extensions are not implementation commitments: they will be evaluated according to need, security, and impact on simplicity.

24. Roadmap and Next Steps

The versions below are proposed conceptual milestones, not releases already published or promised dates. The real version scheme will be decided at the start of the technical specification.

```
0.01  Original conception (Cartnet / Web Cartridge).
0.02  Really Small Web: consolidated architecture and delimited MVP.
0.03  Initial formal specifications of the cartridge and security.
0.04  Go packer, extractor, and validator.
0.05  Identity, signatures, and interoperability vectors.
0.06  rsw:// addressing, resolver, hosting, and offline reader.
0.07  Discovery, webrings, manual updates, and MVP demonstration.
0.08  Experimental event network, after the MVP.
0.09  Isolated application runtime, after security review.
0.10  Broader experimental public version, according to results.
1.00  Stable protocol, only after interoperability and maturity.
```

The MVP is planned within the deliveries up to 0.07, and may be published as an experiment before the social extensions. It is not necessary to wait for 0.10 nor for JavaScript to release the static reader.

Immediate execution order:

1. Write RSW-0001 - Cartridge Format, together with the limits and basic security requirements.
2. Write RSW-0002 and RSW-0003 - manifest, identity, and signatures.
3. Create independent test vectors and a pack/unpack/verify prototype.
4. Close URI, resolution, hosting, updates, and minimal discovery.
5. Implement the local library and the safe static reader.
6. Test three authors, three hostings, and offline navigation.
7. Review, document, measure resources, and release the MVP.

Avoid writing all future specifications before validating the core. The format and the identity model are important foundations; they may be revised during the experimental phase, with recorded changes.

25. Final Vision

Imagine receiving from a friend the image of a small cartridge, opening it, and finding an entire blog. The blog points to a zine, the zine to a library, the library to a small personal diary. Each publication is independent and

can be kept on your computer.

One author's server disappears. Your copy remains available. You can verify the version and the signature, share the original file, and discover new spaces when you return to the network.

RSW intends to create this experience with the smallest possible number of components: an open format, a simple addressing scheme, verifiable identities, common hosting, and a small reader. Active communication can evolve later, without destroying the fundamental properties.

The conceptual scope of the MVP is closed. The specification and the software remain pending. The next delivery is RSW-0001, not a new feature list.

```
Portable by design.
Offline by nature.
Independent by principle.
Connected by choice.

The web, in cartridges.
```

Developed by Pablo Murad November 23, 2026 | pablo@pablomurad.com

END OF DOCUMENT - RSW v0.02

Author's Address

Pablo Murad
Independent
pablo@pablomurad.com
https://pablomurad.com
Official project website: https://rsw.works

Small files. Independent worlds.