**Work Speaks Protocol (WSP)**

**A receipts-first protocol for independent projects: Proof Trails + Community Coordination**

**White Paper (v1.0)**  
**Status:** Public Release  
**Commit Date (UTC):** 2025-12-26  
**Public Release Date (UTC):** **[FILL AT PUBLIC RELEASE]** (YYYY-MM-DD)

**Author / Steward:** Kent Nimmo  
**Contact:** absoluterelativityproject@gmail.com  
**Canonical Home:** <https://absoluterelativity.org>  
**Provenance Identity (Project Wallet):** 0x1F06ea3554aE665e713a637eD136a5065C9cD787

**License:**

* Text/content: CC BY 4.0
* Templates/examples: CC0 1.0
* Code/tooling (if published): MIT

**Date note (readers and auditors):**  
Commit Date is when the canonical ReleasePack hash for this white paper was timestamped under the project’s provenance identity. Public Release Date is when the ReleasePack and this PDF were made publicly available (e.g., on the project website and mirrors).

**Self-Verifying Release (How to verify *this* white paper)**

This white paper is released as a Work Speaks Protocol artifact. The canonical release bundle and its verification record are published in the Proof Trail.

To verify this release:

1. Go to the Proof Trail entry for this white paper release.  
   It will be published in the project’s Artifacts Index and referenced from the Canonical Home when the site goes live.
2. Download the release bundle (zip) from the official mirrors listed there.
3. Compute the SHA-256 hash of the downloaded zip file.
4. Confirm the hash matches the published hash in the Proof Trail entry.
5. Confirm the hash appears in the published timestamp record and is attributable to the project’s provenance identity.

Provenance identity (official timestamp spine): 0x1F06ea3554aE665e713a637eD136a5065C9cD787  
(See the Proof Trail entry for this release for timestamp method and reference.)

**Important boundary:**  
This verification proves that the release bundle existed by a certain time and that the bundle bytes match the declared hash. It does not prove that the contents are correct.

**How to read this document**

This document has two roles:

1. **Vision and rationale** — why WSP exists and what it enables.
2. **Specification** — how to implement a verifiable Proof Trail and (optionally) the token/community layer.

Normative keywords are used as follows:

* **MUST** = required
* **SHOULD** = strongly recommended
* **MAY** = optional

**Scope and disclaimers**

Work Speaks Protocol is credibility infrastructure. It does not guarantee correctness of conclusions, prevent private experimentation, or replace peer review and replication.

WSP may describe token-based community participation and coordination. Nothing in this document is financial, legal, or tax advice. WSP does not require any promise of profit, ownership, dividends, or guaranteed returns.

Peer review and editorial correspondence may be confidential depending on venue policy. WSP supports policy-aware disclosure through PUBLIC, REDACTED-PUBLIC, and HASH-ONLY milestone records as described in Section 7.

Contents

[**1. Introduction** 3](#_Toc217203838)

[**1.1 The problem WSP addresses: trust and coordination are the bottleneck** 3](#_Toc217203839)

[**1.2 What Work Speaks Protocol is** 3](#_Toc217203840)

[**1.3 How this white paper is organized** 3](#_Toc217203841)

[**2. The Vision** 3](#_Toc217203842)

[**2.1 An independent researcher movement: credibility without permission** 4](#_Toc217203843)

[**2.2 Why the token layer is central, not optional** 5](#_Toc217203844)

[**2.3 AI as the missing medium between technical work and the public** 6](#_Toc217203845)

[**3. Why This Is Needed Now** 7](#_Toc217203846)

[**3.1 The institutional bottleneck (without turning this into a rebellion)** 7](#_Toc217203847)

[**3.2 The legitimacy gap for independent work** 8](#_Toc217203848)

[**3.3 Why “publication acceptance” cannot remain the only legitimacy path** 9](#_Toc217203849)

[**4. How WSP Works at a Glance** 10](#_Toc217203850)

[**4.1 The three layers of Work Speaks Protocol** 10](#_Toc217203851)

[**4.2 Failure modes WSP is designed to avoid** 12](#_Toc217203852)

[**4.3 The end-to-end lifecycle timeline (what a WSP project actually does)** 13](#_Toc217203853)

[**4.4 What WSP can prove and what it cannot prove** 14](#_Toc217203854)

[**5. Definitions and Principles** 15](#_Toc217203855)

[**5.1 Core definitions** 15](#_Toc217203856)

[**5.2 Non‑negotiable principles** 17](#_Toc217203857)

[**5.3 Scope notes: what WSP does and does not claim** 19](#_Toc217203858)

[**6. Proof Trail Specification** 20](#_Toc217203859)

[**6.1 Identity anchors (what must be publicly declared)** 20](#_Toc217203860)

[**6.2 Bundles and hashing** 21](#_Toc217203861)

[**6.3 Timestamping the hash** 22](#_Toc217203862)

[**6.4 The Artifacts Index: the mapping layer that makes the Proof Trail usable** 23](#_Toc217203863)

[**6.5 Pack taxonomy (standard bundle types)** 24](#_Toc217203864)

[**6.6 Interoperability (how WSP composes with existing infrastructure)** 27](#_Toc217203865)

[**6.7 Minimum Proof Trail compliance (baseline)** 27](#_Toc217203866)

[**7. Submission Integrity and Journal Accountability** 28](#_Toc217203867)

[**7.1 Why submission integrity matters (what it solves)** 28](#_Toc217203868)

[**7.2 SubmitPack specification (pre-submission commitment)** 29](#_Toc217203869)

[**7.3 DecisionPack specification (policy-aware outcome record)** 31](#_Toc217203870)

[**7.4 RevisionPack(s) and resubmission trail** 33](#_Toc217203871)

[**7.5 Policy-aware transparency guardrails** 33](#_Toc217203872)

[**8. Compliance Levels (Bronze / Silver / Gold)** 34](#_Toc217203873)

[**8.1 Why tiers exist (honest claims + fast auditing)** 34](#_Toc217203874)

[**8.2 Bronze — Timeline integrity (Proof Trail foundation)** 35](#_Toc217203875)

[**8.3 Silver — Rules + inputs integrity (pre-commitment discipline)** 36](#_Toc217203876)

[**8.4 Gold — End-to-end evidence chain (reproducibility + submission integrity + audits)** 37](#_Toc217203877)

[**8.5 How to publish a compliance declaration** 39](#_Toc217203878)

[**9. The Token Layer: What it is and what it is not** 41](#_Toc217203879)

[**9.1 Token as participation and coordination mechanism** 41](#_Toc217203880)

[**9.2 Receipts-first norms: how WSP avoids token-only credibility** 42](#_Toc217203881)

[**9.3 Project identity linking: token ↔ project ↔ wallet ↔ artifacts** 44](#_Toc217203882)

[**9.4 Disclosures and boundaries (risk + integrity)** 45](#_Toc217203883)

[**10. Incentives, Governance Primitives, and Community Design** 47](#_Toc217203884)

[**10.1 Incentivizing audits and replication** 47](#_Toc217203885)

[**10.2 Reputation through work: auditors and replicators as first-class actors** 49](#_Toc217203886)

[**10.3 Governance: lightweight by default, scalable if needed** 51](#_Toc217203887)

[**10.4 Anti-manipulation posture (realistic, not utopian)** 52](#_Toc217203888)

[**11. Implementation Guide** 54](#_Toc217203889)

[**How to actually run Work Speaks Protocol in the real world** 54](#_Toc217203890)

[**11.1 Minimal viable adoption** 54](#_Toc217203891)

[**11.2 Canonical infrastructure stack** 56](#_Toc217203892)

[**11.3 Website requirements** 57](#_Toc217203893)

[**11.4 Human-readable and machine-readable index** 58](#_Toc217203894)

[**11.5 Security and operational continuity** 59](#_Toc217203895)

[**12. Case Studies** 61](#_Toc217203896)

[**Proof by demonstration** 61](#_Toc217203897)

[**12.1 Case Study 0 — This white paper as a WSP artifact** 61](#_Toc217203898)

[**12.2 Case Study 0B — Submission integrity demonstration** 63](#_Toc217203899)

[**12.3 Case Study 1 — Absolute Relativity implementation and end-to-end pipeline** 64](#_Toc217203900)

[**12.4 Living case-study updates: the website is the canonical record** 66](#_Toc217203901)

[**13. Roadmap and Ecosystem Vision** 66](#_Toc217203902)

[**13.1 From “one project” to a decentralized independent research network** 66](#_Toc217203903)

[**13.2 Tooling roadmap** 68](#_Toc217203904)

[**13.3 Integrations and future improvements** 71](#_Toc217203905)

[**14. FAQ / Objections** 72](#_Toc217203906)

[**14.1 “Isn’t this just Zenodo / OSF / a DOI / GitHub releases?”** 73](#_Toc217203907)

[**14.2 “Couldn’t someone just timestamp fake hashes and pretend it proves something?”** 73](#_Toc217203908)

[**14.3 “How do I verify a WSP project if I’m not technical?”** 74](#_Toc217203909)

[**14.4 “Doesn’t public participation (via tokens) create manipulation or speculation problems?”** 74](#_Toc217203910)

[**14.5 “What about peer review confidentiality? Am I allowed to publish decision letters or reviewer reports?”** 75](#_Toc217203911)

[**14.6 “Is WSP anti-academia or anti-journal?”** 76](#_Toc217203912)

[**14.7 “Does WSP prove a theory is correct?”** 76](#_Toc217203913)

[**14.8 “What stops a project from rewriting history by just starting over?”** 76](#_Toc217203914)

[**15. Conclusion / Call to Action** 77](#_Toc217203915)

[**15.1 If you want to adopt WSP: start with one honest milestone** 77](#_Toc217203916)

[**15.2 If you want to contribute to a WSP ecosystem: become an auditor, replicator, or curator** 78](#_Toc217203917)

[**15.3 A note to institutions: WSP is compatible with the scientific ideal** 78](#_Toc217203918)

[**15.4 The core invitation** 79](#_Toc217203919)

[**Appendix A — Glossary** 80](#_Toc217203920)

[**Normative keywords** 80](#_Toc217203921)

[**A** 80](#_Toc217203922)

[**B** 81](#_Toc217203923)

[**C** 81](#_Toc217203924)

[**D** 82](#_Toc217203925)

[**H** 82](#_Toc217203926)

[**I** 82](#_Toc217203927)

[**M** 82](#_Toc217203928)

[**P** 83](#_Toc217203929)

[**R** 83](#_Toc217203930)

[**S** 84](#_Toc217203931)

[**T** 84](#_Toc217203932)

[**V** 85](#_Toc217203933)

[**W** 85](#_Toc217203934)

[**Appendix B — Bundle Templates** 86](#_Toc217203935)

[**Pack structures, required files, and MANIFEST.json schema** 86](#_Toc217203936)

[**B.1 General bundle conventions (applies to every Pack)** 86](#_Toc217203937)

[**B.2 README.md template (recommended minimum content)** 87](#_Toc217203938)

[**B.3 MANIFEST.json schema (bundle-level metadata)** 87](#_Toc217203939)

[**B.4 Pack templates (recommended folder structures)** 89](#_Toc217203940)

[**B.5 Redaction and disclosure templates** 96](#_Toc217203941)

[**B.6 Minimal “starter bundles” (copy-ready suggestions)** 97](#_Toc217203942)

[**Appendix C — Naming and Versioning Conventions** 98](#_Toc217203943)

[**Stable identifiers, clear filenames, and credibility-preserving version discipline** 98](#_Toc217203944)

[**C.1 Design goals** 98](#_Toc217203945)

[**C.2 Versioning rules (the credibility contract)** 99](#_Toc217203946)

[**C.3 Filename conventions for packs (recommended default)** 100](#_Toc217203947)

[**C.4 Artifact ID conventions (recommended)** 101](#_Toc217203948)

[**C.5 Relationship declarations (how to make evolution legible)** 102](#_Toc217203949)

[**C.6 Index file naming conventions (recommended)** 103](#_Toc217203950)

[**C.7 Changelog and correction conventions** 103](#_Toc217203951)

[**C.8 Multi-venue submissions and parallel tracks** 104](#_Toc217203952)

[**C.9 Minimal checklist (before publishing any milestone)** 105](#_Toc217203953)

[**Appendix D — Artifacts Index Schema** 105](#_Toc217203954)

[**Machine-readable Proof Trail mapping (wsp\_index.json and wsp\_index.csv)** 105](#_Toc217203955)

[**D.1 Index design goals (why this schema exists)** 106](#_Toc217203956)

[**D.2 Canonical files and publishing requirements** 106](#_Toc217203957)

[**D.3 wsp\_index.json top-level structure (recommended)** 107](#_Toc217203958)

[**D.4 Artifact entry object (required fields + recommended fields)** 108](#_Toc217203959)

[**D.5 Canonical entry schema (recommended)** 109](#_Toc217203960)

[**D.6 HASH‑ONLY entry requirements (policy-aware transparency)** 110](#_Toc217203961)

[**D.7 wsp\_index.csv schema (columns and mapping)** 111](#_Toc217203962)

[**D.8 Validation rules (what tooling SHOULD check)** 112](#_Toc217203963)

[**D.9 Minimal guidance for projects adopting this schema** 113](#_Toc217203964)

[**Appendix E — Verification Guide** 114](#_Toc217203965)

[**Beginner verification, auditor verification, and how to report discrepancies** 114](#_Toc217203966)

[**E.1 What you need (minimal)** 114](#_Toc217203967)

[**E.2 Beginner verification (PUBLIC milestone)** 115](#_Toc217203968)

[**Verify one milestone end-to-end in minutes** 115](#_Toc217203969)

[**E.3 How to verify timestamp records (method-agnostic rules)** 117](#_Toc217203970)

[**E.3.1 Common timestamp patterns and what to check** 117](#_Toc217203971)

[**E.4 Auditor verification (deeper checks)** 118](#_Toc217203972)

[**E.4.1 Proof Trail completeness check (“Explain Every Hash”)** 118](#_Toc217203973)

[**E.4.2 Bundle internal integrity checks (PUBLIC / REDACTED‑PUBLIC bundles)** 119](#_Toc217203974)

[**E.4.3 Relationship graph sanity check** 119](#_Toc217203975)

[**E.4.4 RunPack reproducibility / inspection (when applicable)** 120](#_Toc217203976)

[**E.4.5 Submission integrity verification (when present)** 120](#_Toc217203977)

[**E.5 Publishing verification as an AuditPack (recommended output format)** 120](#_Toc217203978)

[**E.5.1 Minimal AuditPack contents (recommended)** 120](#_Toc217203979)

[**E.5.2 Auditor standards (culture)** 121](#_Toc217203980)

[**E.6 Reporting discrepancies (responsible disclosure and corrective workflow)** 121](#_Toc217203981)

[**E.6.1 Discrepancy severity categories (practical)** 121](#_Toc217203982)

[**E.6.2 What to include in a good discrepancy report** 122](#_Toc217203983)

[**E.6.3 Preferred remediation behavior (what a WSP project should do)** 122](#_Toc217203984)

[**E.7 Security notes (verify safely)** 122](#_Toc217203985)

[**Appendix F — Checklists and Quick Reference** 123](#_Toc217203986)

[**Practical checklists for projects, auditors, and communities** 123](#_Toc217203987)

[**F.1 Bronze / Silver / Gold compliance checklists** 123](#_Toc217203988)

[**F.2 “Publish a milestone” checklist (applies to any Pack)** 125](#_Toc217203989)

[**F.3 Pack-specific mini checklists (quick expectations)** 126](#_Toc217203990)

[**F.4 Submission integrity checklist (policy-aware)** 127](#_Toc217203991)

[**F.5 Artifacts Index maintenance checklist (continuous operations)** 128](#_Toc217203992)

[**F.6 Operational security checklist (protect the credibility spine)** 129](#_Toc217203993)

[**F.7 Community norms quick reference (receipts-first culture)** 129](#_Toc217203994)

[**Appendix G — Public Statement Templates** 130](#_Toc217203995)

[**Compliance declaration, provenance statements, and policy-aware submission language** 130](#_Toc217203996)

[**G.1 WSP Compliance Declaration Template (Bronze / Silver / Gold)** 130](#_Toc217203997)

[**G.2 WSP Provenance Statement Template (for papers, releases, or README files)** 132](#_Toc217203998)

[**G.3 Submission Integrity Statement Template (policy-aware)** 133](#_Toc217203999)

[**G.4 Token / Community Statement Template (receipts-first participation)** 133](#_Toc217204000)

[**G.5 Wallet / Identity Migration Notice Template (continuity under key rotation)** 134](#_Toc217204001)

[**Appendix H — Threat Model and Risk Considerations** 135](#_Toc217204002)

[**What WSP defends against, what it cannot, and how to handle incidents** 135](#_Toc217204003)

[**H.1 Security objectives** 135](#_Toc217204004)

[**H.2 Adversary model** 136](#_Toc217204005)

[**H.3 What WSP mitigates well** 137](#_Toc217204006)

[**H.4 What WSP cannot prove (and does not claim to)** 137](#_Toc217204007)

[**H.5 Common failure modes and recommended mitigations** 138](#_Toc217204008)

[**H.6 Incident response and continuity** 141](#_Toc217204009)

[**H.7 Summary: what makes WSP robust** 142](#_Toc217204010)

**1. Introduction**

Work Speaks Protocol (WSP) is a standard for building **public credibility and community coordination** around serious work—especially independent research—by combining two things that are usually separate:

1. a **Proof Trail**: a public, verifiable record of what a project declared and when (artifacts packaged, hashed, timestamped, and mapped so anyone can audit), and
2. a **Community Token layer**: a simple mechanism for public participation, coordination, and support around work that can be inspected.

WSP is built for a near-future environment where powerful AI tools are widely available, the volume of claims and “discoveries” will explode, and credibility cannot depend primarily on credentials, closed networks, or institutional affiliation. In that environment, credibility must become closer to **infrastructure**—something legible and auditable—rather than something granted mostly through reputation and gatekeeping.

WSP is not anti-academia and not a replacement for peer review, preprint servers, DOI repositories, or code hosting. It is designed to **interoperate** with existing infrastructure, while adding a missing layer: a unified, auditable “chain of declared work” that ties a project’s claims to specific immutable bundles and shows how those bundles evolved over time.

**1.1 The problem WSP addresses: trust and coordination are the bottleneck**

AI is rapidly lowering the barrier to doing serious intellectual work. More people can now:

* read and synthesize technical literature faster,
* run large-scale analyses and simulations,
* write coherent technical documents,
* and build tools that previously required institutional resources.

This shift is empowering, but it creates a new problem: when output becomes cheap, **verification becomes scarce**.

In an AI-saturated world, the bottleneck is no longer “can someone generate a convincing narrative?” The bottleneck becomes:

* **Credibility:** can outsiders tell whether a project is behaving with integrity, not quietly rewriting its past, and making claims anchored to real artifacts?
* **Auditability:** can a third party verify what data, code, and configurations produced a result—without relying on trust in the author’s credentials?
* **Coordination:** can attention, replication, critique, and support flow toward the most promising work—especially high-variance novelty—without depending entirely on institutional filters?

Traditional institutions solve parts of this problem (peer review, credentialing, grant allocation, editorial standards), and they remain valuable. But they are not designed to scale to a world where millions of independent projects can produce “research-like” output. WSP is an attempt to make legitimacy formation more scalable by anchoring it to **public receipts** and a transparent record of evolution.

**1.2 What Work Speaks Protocol is**

Work Speaks Protocol is a **token-backed standard for credibility and coordination** around independent projects in the AI era.

At a practical level, WSP standardizes how a project:

* creates versioned “milestone bundles” (zips of papers, data, code, run outputs, decision records, etc.),
* hashes those bundles using a modern cryptographic hash (e.g., SHA‑256),
* timestamps those hashes through a public provenance identity, and
* publishes a public **Artifacts Index** that explains what each timestamp corresponds to and where to download the exact files (or why a bundle is hash-only).

This creates a Proof Trail that is hard to rewrite quietly. Over time, a project’s credibility becomes more **backloaded** than frontloaded: the work becomes harder to dismiss, not because of who produced it, but because its artifacts, commitments, corrections, and evolution are objectively visible.

WSP then adds a second layer: the Community Token.

WSP treats the token as a **social coordination tool**, not a substitute for evidence. The token exists to make it easy for people to:

* join a project community they believe is acting with integrity,
* support research directions they want explored,
* coordinate attention toward what should be audited, replicated, or tested next,
* and express “this deserves to be taken seriously” without requiring institutional endorsement.

The key idea is that credibility formation works best when participation is anchored to receipts. WSP’s intended legitimacy model is:

**evidence → public verification → community coordination → credibility**

WSP does not claim that tokens create truth. WSP claims that tokens can help communities coordinate around truth-seeking activity—when that activity is anchored to verifiable artifacts.

**1.3 How this white paper is organized**

This white paper has four parts:

**Part I: Vision and rationale**  
Explains the bigger picture: why WSP matters socially and institutionally, why it is designed for an AI-rich world, and why the token layer is central to the movement.

**Part II: Protocol specification**  
Defines the core terms, non-negotiable principles, bundle types (“packs”), verification rules, submission integrity workflow, and compliance tiers. This is where WSP becomes a real standard that projects can follow and outsiders can audit.

**Part III: Token community layer**  
Describes how WSP treats the token as a participation and coordination instrument, how receipts-first norms prevent “token-only credibility,” and how communities can incentivize audits, replication, and constructive critique.

**Part IV: Implementation and case studies**  
Shows how to adopt WSP with minimal overhead, how to publish an auditable Artifacts Index, and how real-world examples will be maintained over time. The white paper itself is intended to be released as a WSP artifact as a first demonstration of the protocol in action.

The appendices provide templates, schemas, and checklists so projects can implement WSP consistently without reinventing the workflow.

**2. The Vision**

Work Speaks Protocol is built around a simple conviction: **in the age of AI, credibility should be earned through auditable work—not granted through credentials, gatekeeping, or permission.** The protocol is meant to help create a world where serious independent projects can emerge, grow, and be evaluated in public, with a legitimacy path that is transparent, repeatable, and scalable.

This section describes the larger vision WSP is designed to serve: a future where independent research becomes mainstream, where the public can meaningfully participate in scientific progress, and where communities coordinate around *verification* rather than hype.

**2.1 An independent researcher movement: credibility without permission**

For most of modern history, doing “serious research” has been bottlenecked by access—access to institutions, compute, journals, funding, and networks. That reality created a strong coupling between **credentials** and **legitimacy**. Credentials are not meaningless, but they are not the same thing as truth, and they are not a scalable way to evaluate the flood of work that is coming.

AI changes the shape of the world.

As AI becomes a universal tool for reasoning assistance, simulation support, coding, literature synthesis, translation, and explanation, far more people will be capable of producing genuine intellectual output—often from outside academia. This is not a hypothetical. It is already happening. And it will accelerate.

That creates a new social question:

* If more people can do serious work, **how do they earn legitimacy** without institutional approval?

WSP’s answer is: **make legitimacy procedural.**

Instead of asking the public to trust an identity (“Who are you?”), WSP encourages projects to show a verifiable track record (“What have you committed to, what did you do, and can it be audited?”). Over time, credibility becomes something that **compounds** through:

* consistent version discipline,
* transparent corrections,
* reproducibility,
* and an additive record that cannot be quietly rewritten.

This is the “work speaks” thesis in practice: *not “believe me,” but “verify the receipts.”*

WSP is explicitly designed to support people who do not have institutional permission, and may never seek it. It is also designed for the many people who—because of automation and economic transitions—may find themselves looking for new forms of meaningful contribution. If large numbers of people are going to be displaced from traditional work structures, then society benefits from systems that can help people channel effort into rigorous, value-producing projects that are publicly auditable.

The intended end state is not an “anti-institution” rebellion. It is a **broader legitimacy pipeline**: one where institutions remain valuable, and independent work can still be evaluated on its merits with a clear, standardized credibility trail.

**2.2 Why the token layer is central, not optional**

WSP is often misunderstood if it is described only as “hashing and timestamps.” The proof trail is necessary, but it is not sufficient for the vision.

The token layer is included because it solves a different problem: **coordination**.

A world with millions of independent projects doesn’t just need receipts. It needs a way for people to:

* discover projects,
* join communities,
* coordinate audits and replications,
* and direct attention and support toward work that appears to be behaving with integrity and generating value.

The token is meant to function as a **public participation instrument**—a simple way for someone to say:

* “I want to be part of this project’s network,” and
* “I want this direction explored and tested.”

WSP’s token layer is not intended to replace evidence. It is intended to help communities coordinate around evidence-seeking activity. In other words:

* **The proof trail anchors reality.**
* **The token coordinates people.**

This matters because credibility formation is not only about correctness; it is also about whether a project can attract the auditing, replication, critique, and sustained attention required to *discover* correctness.

Without coordination, a project can be rigorous and still disappear.

Without receipts, a project can coordinate attention and still drift into hype.

WSP is built to prevent both failure modes by binding coordination to a publicly auditable record.

To keep the token layer aligned with the scientific ideal, WSP’s intended culture is “receipts-first”:

* major claims should reference specific artifacts,
* progress should be traceable through versioned bundles,
* independent audits and replications should be celebrated and rewarded,
* and correction should strengthen credibility rather than destroy it.

In this model, “public credibility voting” does not mean “token price equals truth.” It means the public has a structured way to participate in deciding what deserves attention, testing, and support—grounded in a proof trail that makes that participation accountable.

**2.3 AI as the missing medium between technical work and the public**

One reason WSP becomes viable now—not ten or twenty years ago—is that AI increasingly acts as a **translation layer** between specialized work and general understanding.

Historically, a large portion of the public was effectively locked out of meaningful engagement with technical research. Even if papers were publicly available, interpreting them required:

* advanced prerequisites,
* specialized vocabulary,
* and significant time investment.

AI changes that dynamic. Increasingly, a person can:

* ask AI to summarize a paper at their level,
* ask for explanations of equations or methods,
* compare two competing claims,
* identify what evidence would support or falsify a theory,
* and navigate complex material without years of formal training.

This doesn’t eliminate the value of expertise. It expands who can participate in the early stages of evaluation and coordination—especially when the underlying work is packaged in a way that is easy to verify.

WSP is designed to be compatible with this future in two ways:

1. **Machine-readable structure**  
   A standardized artifacts index, consistent bundle formats, and clear relationships between milestones make it easier for AI tools to assist with auditing, version comparison, and reproduction guidance.
2. **Human-legible integrity**  
   The protocol creates a public record that is understandable at a higher level even for non-specialists: what was committed, what changed, what was tested, what was corrected, and what was reproduced by third parties.

The long-term vision is a world where scientific progress is not an isolated conversation among a small technical class, while the rest of society remains largely disengaged. Instead, WSP is designed for a world where public engagement becomes more serious, more informed, and more anchored to evidence—because the tools exist to support that engagement and the protocol exists to organize it.

**3. Why This Is Needed Now**

Work Speaks Protocol is a response to a very specific moment: **capability is being democratized faster than legitimacy systems can adapt.** AI is expanding who can produce serious work, but the dominant credibility pathways—especially in science and research—still assume that legitimacy mostly flows through institutions, formal credentials, and a small number of gatekept channels.

That mismatch creates friction in three directions at once:

* high-quality independent work struggles to be seen as legitimate,
* the public lacks a reliable way to distinguish serious independent work from noise, and
* institutions face mounting pressure because their filtering role is increasingly visible, contested, and slow relative to the pace of change.

WSP is designed to meet this moment with a simple standard: **make credible work auditable, and make participation possible without permission.**

**3.1 The institutional bottleneck (without turning this into a rebellion)**

Institutions—universities, journals, conferences, and funding bodies—exist for good reasons. They concentrate expertise, enforce standards, fund long-running programs, and provide stable pathways for training and evaluation. WSP does not argue against these roles.

But institutional systems also have structural constraints that become more consequential as novelty and volume increase:

**1) Incentives favor safety over high-variance novelty.**  
Many institutional processes are optimized to reduce false positives and maintain stability. That naturally biases against work that is:

* outside prevailing frameworks,
* difficult to review with existing heuristics,
* or socially “risky” to endorse early.

This is not necessarily malice; it is often institutional self-preservation and risk management. But it can slow evaluation of genuinely valuable novelty.

**2) Review bandwidth is limited, and filters get tighter as volume rises.**  
Peer review and editorial processes do not scale linearly with the number of submissions. As volume grows, systems often respond by adding stronger triage, stricter scope, and faster rejections—especially for unfamiliar approaches. That can produce a “visibility problem” for independent work and for unconventional ideas.

**3) Legitimacy is still coupled to identity.**  
Even in systems that attempt double-blind review, credibility in practice often depends on institutional signals: affiliation, network proximity, past publication record, and alignment with accepted framing. This coupling is not always explicit, but it exists as a social reality.

**4) Institutions are not designed to be publicly auditable.**  
Most editorial decisions, review dynamics, and the reasons behind rejections are not visible to the public. This can be appropriate in many cases, but it also means the public cannot easily distinguish:

* scientifically grounded disagreement,
* “fit” decisions,
* and decisions shaped by convention.

WSP does not aim to replace institutions. It aims to add a parallel credibility pathway that remains compatible with them, while making independent work **legible and auditable** without requiring institutional endorsement.

**3.2 The legitimacy gap for independent work**

As independent research becomes more common, the public will face a familiar problem: **how to know what to take seriously** when there is no institutional stamp.

Today, independent work often suffers from two extremes:

* Some independent projects are dismissed by default as “not credible” because they lack affiliation.
* Other independent projects gain attention through charisma, narrative, or marketing, even when their work is not anchored to reproducible artifacts.

Both failures come from the same missing infrastructure: a standardized way for independent projects to build a **track record that can be audited by outsiders.**

WSP addresses this legitimacy gap by shifting the question from:

* “Who are you?”  
  to
* “What have you done, what did you commit to when, and can we verify it?”

A Proof Trail makes it possible to evaluate independent work on the basis of:

* version history and coherence over time,
* transparency around corrections and revisions,
* traceability between claims and concrete artifacts,
* reproducibility (when applicable), and
* third-party audits and replications.

This does not guarantee correctness. It does something more basic and more important for scaling legitimacy: it makes integrity **visible and checkable**, which is the foundation for serious evaluation.

**3.3 Why “publication acceptance” cannot remain the only legitimacy path**

Peer-reviewed publication remains valuable, but it cannot function as the only legitimacy mechanism in an AI-driven world for three reasons:

**1) Acceptance is not the same as truth, and rejection is not the same as error.**  
Publication decisions can reflect methodological quality, novelty, reviewer bandwidth, scope fit, and convention. They can also reflect the simple reality that novel work often takes time to be understood. A system that equates legitimacy with acceptance creates a bottleneck where many ideas are never tested publicly enough to be evaluated fairly.

**2) “Time-to-evaluation” matters more now.**  
In a world where technology advances quickly, the cost of waiting decades for recognition can be too high. The goal is not to bypass rigor. The goal is to **accelerate the path from claim → audit → test** while preserving accountability.

**3) The public needs a way to evaluate institutional behavior constructively.**  
As scientific literacy becomes more accessible through AI, more people will want visibility into:

* what is being evaluated,
* what is being rejected,
* and why.

WSP’s submission integrity model is designed to be policy-aware and non-hostile, while still providing a public record. If a project timestamps a submission package before submitting, then later records the outcome (with appropriate confidentiality handling), the public can eventually assess patterns—both for projects and for venues—based on a shared record.

This is not about turning peer review into a spectacle. It is about aligning incentives with the scientific ideal:

* projects become more accountable for the integrity of their declared work, and
* venues become more accountable—over time and at the institutional level—for the standards they appear to apply.

In short: **publication can remain a major pathway, but it cannot remain the only pathway.** WSP exists to add a legitimacy layer that is compatible with institutions, yet does not depend on them—so that credible independent work can be tested, audited, and supported in public.

**4. How WSP Works at a Glance**

Work Speaks Protocol is designed to be understood quickly at a conceptual level and implemented later with precision. This section gives the “one-screen” view: the layers of the protocol, the failure modes it avoids, the end-to-end workflow, and the boundaries of what it can and cannot prove.

**4.1 The three layers of Work Speaks Protocol**

WSP is best understood as three layers that reinforce each other. Each layer solves a different bottleneck, and the system works because none of the three is treated as sufficient on its own.

**Layer 1: The Proof Trail (receipts-first credibility)**

The Proof Trail is a public record of what a project **declared**, **when it declared it**, and **how those declarations connect over time**.

At a high level, the Proof Trail consists of:

* **artifact bundles** (typically a zip file containing a paper, dataset snapshot, code pointers, run logs, etc.),
* **a cryptographic hash** of the bundle (e.g., SHA‑256),
* **a public timestamp** anchoring that hash to a moment in time under the project’s provenance identity, and
* an **Artifacts Index** (a public mapping page/file) that explains every recorded hash:
  + what it represents,
  + where the bundle can be downloaded (or why it is not public), and
  + how it relates to other milestones.

This creates a discipline of “no silent rewrite”: the project cannot quietly swap out artifacts or retell its history without leaving visible evidence of version change.

**Layer 2: Submission Integrity and policy-aware accountability (commitments before outcomes)**

WSP extends the Proof Trail to cover the most credibility-sensitive moments—especially moments that are commonly disputed later, such as:

* “Was your method frozen before you saw the outcome?”
* “Did you change criteria after seeing results?”
* “What exactly did you submit to the journal, and when?”
* “What happened in the revision cycle?”

WSP answers these by encouraging **pre-commitment artifacts** (e.g., a submission package committed before submission) and **policy-aware outcome records** (e.g., decision summaries, redacted reports, or hash-only records when confidentiality rules apply).

This layer is not about attacking journals. It is about making both project behavior and publication outcomes more legible over time—while respecting real confidentiality constraints.

**Layer 3: The Community Token (participation and coordination)**

The token layer exists because credibility formation is not only about evidence existing—it is also about whether evidence is **audited, tested, replicated, and improved**.

The token layer provides a simple public mechanism to:

* form a community around a project,
* coordinate attention toward what should be verified next,
* fund or incentivize audits and replication efforts (directly or indirectly),
* and support work that appears to be behaving with integrity and building value over time.

WSP’s core cultural constraint is that community coordination must remain **anchored to receipts**. The token is not meant to replace evidence; it is meant to help coordinate the effort required to evaluate evidence at scale.

**4.2 Failure modes WSP is designed to avoid**

WSP is intentionally built around “anti-failure” design. The protocol is not just a set of tools; it is a set of constraints and norms aimed at preventing predictable credibility breakdowns.

**Failure mode 1: Token-only credibility (hype replaces verification)**

In many token ecosystems, narratives can detach from underlying reality. If a token becomes the primary source of legitimacy, communities can drift into persuasion and group identity rather than truth-seeking.

**WSP’s response:** the token layer is explicitly secondary to the Proof Trail. Major claims should map to concrete artifacts. Integrity is demonstrated through version discipline, reproducibility support, and independent audits—not through price action or social consensus.

**Failure mode 2: Receipts without coordination (rigorous work stays invisible)**

A project can do high-integrity work and still fail socially because no one notices it, no one replicates it, and no one invests attention into testing it.

**WSP’s response:** the community layer exists to coordinate attention, audits, and replication around verifiable milestones. WSP treats auditing and replication as first-class contributions, not afterthoughts.

**Failure mode 3: Publication acceptance becomes the only legitimacy path**

If the only legitimacy pathway is journal acceptance, then unconventional work is often filtered out before it is publicly tested. This slows evaluation of high-variance novelty, and it can preserve conventions even when evidence is moving.

**WSP’s response:** WSP provides a parallel legitimacy pathway based on public artifacts and auditable history. Journals remain valuable, but they are no longer the only route by which work can become credible and testable in public.

**Failure mode 4: Opaque gatekeeping (no legible record of what happened)**

When decisions and outcomes are not legible, distrust grows—either toward independent projects (“they’re hiding something”) or toward institutions (“they’re suppressing something”). Without records, disagreements become narratives.

**WSP’s response:** policy-aware transparency. WSP encourages projects to record pre-commitments (e.g., submissions before submission) and outcomes (e.g., decision summaries), using redaction or hash-only methods where confidentiality applies. The goal is not spectacle. The goal is legibility.

**4.3 The end-to-end lifecycle timeline (what a WSP project actually does)**

WSP uses a simple “milestone chain” model. A project creates a series of versioned bundles that represent meaningful steps in its work. Each bundle is hashed, timestamped, and mapped in the Artifacts Index so outsiders can verify integrity and follow the evolution.

A typical end-to-end WSP workflow looks like this:

**1) RulesPack (freeze definitions and evaluation rules)**  
Before outcome-revealing work, the project records what it intends to do and how it will evaluate results. This is where the project “locks in” key definitions, scoring rules, and criteria.

**2) DataPack (record data as received)**  
The project records the raw data or dataset snapshot used, along with source notes and acquisition context. If the data cannot be shared publicly (privacy/licensing), WSP supports a hash-only record with a public description of the constraint.

**3) RunPack (analysis / simulation / computation record)**  
The project records the run-level details: code reference(s), environment, configuration, logs, and outputs, plus reproduction instructions where feasible.

**4) ReleasePack (public release milestone)**  
The project publishes a coherent public release—paper/preprint, results summary, or major update—linked to the underlying packs that support it. This is the point where claims become cleanly referencable.

**5) SubmitPack (pre-submission commitment)**  
If the project submits to a journal or venue, it records the exact submission package *before submission* (public or hash-only, depending on policy strategy).

**6) DecisionPack (policy-aware outcome record)**  
The project records the decision outcome (and, if permitted, the decision letter/reviews). Where confidentiality applies, WSP supports a public summary plus hash-only storage of verbatim materials.

**7) RevisionPack(s) (revision and resubmission trail)**  
If revisions occur, the project records revised manuscripts and response trails as new artifacts rather than overwriting old versions.

**8) AuditPack(s) (independent verification and replication)**  
At any point, third parties can publish audits and replications as AuditPacks, referencing the exact artifacts they checked. These become part of the public credibility story.

This chain is not limited to academic research. It can be used for any serious project where progress is evidence-based and versioned. The central idea is that a project’s legitimacy becomes grounded in an auditable trail of commitments, outputs, and corrections—rather than in a single gatekeeper event.

**4.4 What WSP can prove and what it cannot prove**

WSP is powerful precisely because it is honest about its boundaries.

**What WSP can prove (strong guarantees, when used correctly)**

WSP can provide public evidence that:

* a specific bundle **existed by a certain time** (timestamped hash),
* the bundle a person downloads **matches** the committed bundle (hash match),
* the commitment is tied to a project’s declared provenance identity (provenance record + index mapping),
* a project’s declared history is **traceable** across versions (no silent rewrite),
* a submission package existed **before submission** (pre-submission commitment discipline).

These are credibility primitives. They do not establish truth, but they establish integrity and chronology.

**What WSP cannot prove (and does not claim to prove)**

WSP cannot guarantee that:

* the contents of a bundle are correct or true,
* a project did not do private experimentation before committing,
* a project disclosed every internal step (only what it declared),
* a market or community consensus equals scientific truth,
* institutions acted fairly in every case (only that records can be made more legible).

WSP is best understood as **credibility infrastructure**: it lowers the cost of verification and raises the cost of deception, while enabling public participation and coordination around work that can actually be audited.

**5. Definitions and Principles**

This section defines the core terms used throughout Work Speaks Protocol (WSP) and sets the non‑negotiable principles that make WSP credible in practice. The protocol is intentionally simple: it is a standard for **publicly auditable declared history** plus **coordination around verifiable work**.

Unless stated otherwise, the keywords **MUST**, **SHOULD**, and **MAY** are used in their plain, normative sense:

* **MUST**: required for compliance with the stated rule.
* **SHOULD**: strongly recommended; there may be valid reasons to deviate, but deviations should be rare and explained.
* **MAY**: optional.

**5.1 Core definitions**

**Project**  
A *Project* is any organized effort that produces versioned outputs and makes public claims about progress. WSP is designed for research projects, but applies to any work where credibility depends on traceable evolution (e.g., software, methodologies, frameworks, educational systems).

**Artifact**  
An *Artifact* is any concrete output relevant to a project’s claims, integrity, or reproducibility. Examples include manuscripts, datasets, code snapshots, run configurations, logs, figures, analysis outputs, submission packages, decision records, revision responses, and audit reports.

**Bundle**  
A *Bundle* is a package of one or more artifacts treated as a single immutable unit for verification. WSP typically uses a zip archive as the bundle file. The bundle file (its exact bytes) is what gets hashed and timestamped.

**Hash**  
A *Hash* is a cryptographic fingerprint of a file. WSP uses hashes to make integrity checkable: if the file changes, the hash changes. WSP’s default is **SHA‑256**, but the protocol is hash‑algorithm agnostic so long as the algorithm is widely accepted, deterministic, and publicly verifiable.

**Timestamp**  
A *Timestamp* is a publicly verifiable record that a given hash existed by a certain time and was declared by the project’s provenance identity. The timestamp mechanism may vary (e.g., blockchain transaction data, trusted timestamping services, or other public anchors), but the verification outcome must be objective: outsiders must be able to check that the timestamp record contains the exact hash and is associated with the project identity.

**Provenance identity**  
A *Provenance identity* is the public identifier used to anchor and authenticate the project’s Proof Trail. In WSP, this is typically a **Provenance Wallet** (a wallet address) used as the “official spine” of the record. The key requirement is not the technology; it is the public legibility of the identity and its continuity over time.

**Provenance Wallet**  
The *Provenance Wallet* is the project’s designated wallet address used to issue official timestamp records for WSP milestones. It functions as a public identity anchor, not as a general-purpose spending wallet. (WSP strongly recommends separating this wallet from treasury/admin wallets; see Section 11.)

**Proof Trail**  
A *Proof Trail* is the project’s public, auditable record of declared milestones. At minimum, it consists of:

1. bundles (or bundle descriptions),
2. their hashes,
3. timestamp records anchoring those hashes under the provenance identity, and
4. an Artifacts Index that maps timestamps to meaningful descriptions and downloadable files.

**Artifacts Index**  
The *Artifacts Index* is the mapping layer that makes the Proof Trail understandable and auditable. It is a public page and/or machine-readable file that explains every recorded hash: what it represents, where to download the corresponding bundle (or why it is unavailable), and how it relates to other milestones.

**Milestone**  
A *Milestone* is a declared event in the project’s evolution represented by a bundle and its timestamp record. Milestones form the project’s Proof Trail.

**Version**  
A *Version* is a label that distinguishes one milestone bundle from another and makes evolution comparable. Versioning is a core credibility mechanism: if content changes, a new version must be created and declared.

**Visibility classes**  
WSP recognizes three visibility classes for bundles:

* **PUBLIC**: the bundle is downloadable so outsiders can verify it directly.
* **REDACTED‑PUBLIC**: the bundle is downloadable but contains redactions, documented clearly.
* **HASH‑ONLY**: the bundle hash is declared and timestamped, but the bundle is not publicly distributed (typically due to privacy, licensing, or policy constraints). Hash‑only is allowed, but must be used transparently and sparingly, with a clear public explanation of why the bundle cannot be published.

**Pack**  
A *Pack* is a named bundle type representing a common milestone category (e.g., RulesPack, DataPack, RunPack, ReleasePack, SubmitPack). Packs standardize expectations so outsiders can interpret milestones consistently. (Pack taxonomy is specified in Section 6.)

**5.2 Non‑negotiable principles**

The principles below are the “constitution” of WSP. A project can adopt WSP tools and still fail if it violates these principles. Conversely, a project that follows these principles consistently can build durable credibility over time, even without institutional endorsement.

**Principle 1: No silent rewrite**

Once a bundle has been declared (hashed and timestamped), it **MUST NOT** be replaced in place.

* If something needs to change, the project **MUST** publish a new version as a new milestone.
* Older versions **MUST** remain accessible (PUBLIC) when feasible, or at least remain referenced in the Artifacts Index even if a host changes.

This is the core mechanism that makes a Proof Trail credible. WSP is explicitly designed to reward additive history and punish revisionist history.

**Principle 2: Explain every hash**

Every hash the project timestamps under its provenance identity **MUST** have a corresponding entry in the Artifacts Index that explains:

* what the hash represents (bundle type, version, purpose),
* where the corresponding bundle can be downloaded (or why it is hash‑only), and
* how it relates to other milestones.

A wallet history without a readable mapping layer is not a Proof Trail; it is noise. The Artifacts Index is not optional decoration—it is a required legibility layer.

**Principle 3: Version discipline is credibility**

A project **MUST** treat versioning as a credibility primitive, not a marketing decision.

* If substantive content changes, the version **MUST** change.
* Major pivots **SHOULD** be labeled and justified clearly.
* Corrections **SHOULD** be documented explicitly (what changed and why).

WSP assumes that honest work evolves. Version discipline ensures evolution is legible and accountable.

**Principle 4: Public by default, privacy by design**

WSP is “public by default” because public verification is the point. At the same time, WSP is “privacy by design” because real projects have legitimate constraints.

* Projects **SHOULD** publish milestones as PUBLIC whenever possible.
* Projects **MAY** use HASH‑ONLY milestones when necessary due to privacy, licensing, security, or policy constraints.
* When using HASH‑ONLY, projects **MUST** publish a clear public explanation of the constraint and what is being withheld.

Hash‑only is a tool for responsible disclosure—not a loophole for hiding inconvenient history.

**Principle 5: Interoperate, don’t reinvent**

WSP is not a replacement for repositories, DOIs, preprints, or code hosting. WSP is a protocol layer that ties them together.

* Projects **SHOULD** use existing stable infrastructure (e.g., DOI archives, code release platforms) for distribution and redundancy.
* WSP’s role is to provide the unified provenance spine and auditable mapping layer across those systems.

This principle keeps WSP practical and reduces friction for adoption.

**Principle 6: Token participation must remain anchored to receipts**

WSP includes the token layer to coordinate communities—not to replace evidence.

* Communities and project communications **SHOULD** treat major claims as meaningful only when they map to verifiable artifacts in the Proof Trail.
* Projects **SHOULD** encourage audits, replication, and falsification attempts as first‑class contributions.

WSP explicitly rejects “token price = truth.” The Proof Trail anchors integrity; the token coordinates participation.

**Principle 7: Accountability without hostility**

WSP supports accountability for both projects and institutions (including journals), but it **MUST** be practiced constructively.

* Projects **MUST NOT** encourage harassment, doxxing, or intimidation.
* Critique **SHOULD** focus on artifacts, methods, and reasoning—never on personal targeting.
* Where confidentiality policies apply (e.g., peer review), projects **MUST** handle disclosure responsibly.

WSP is intended to elevate standards, not degrade discourse.

**Principle 8: The scientific ideal is the reference point**

WSP is designed to help projects live up to the scientific ideal: clarity, falsifiability where applicable, reproducibility where feasible, transparent correction, and honest accounting of what was done.

The protocol does not assume institutions always fail or that outsiders are always right. It assumes something simpler:

* credibility should be earned through auditable behavior, and
* evaluation should become easier, faster, and more transparent as tools improve.

**5.3 Scope notes: what WSP does and does not claim**

To avoid confusion, WSP is explicit about what it is fundamentally designed to accomplish.

**What WSP is designed to accomplish**

WSP is designed to make it easier for outsiders to verify:

* that specific bundles existed by specific times,
* that public bundles match their declared hashes,
* that a project’s declared evolution is traceable (no silent rewrite),
* and that key commitments (e.g., frozen methods, submissions) were made before outcomes.

This creates a legitimacy pathway for independent work that does not depend on credentials.

**What WSP is not designed to guarantee**

WSP does not guarantee:

* that a claim is true,
* that a project did no private exploration before publishing milestones,
* that all steps were disclosed (only what was declared),
* that community consensus equals correctness, or
* that any institution behaved fairly in any specific case.

WSP is best described as **credibility infrastructure**: it lowers the cost of verification and raises the cost of deception, while enabling public participation and coordination around work that can be audited.

**6. Proof Trail Specification**

The Proof Trail is the credibility backbone of Work Speaks Protocol. It is the part of WSP that makes a project’s declared history **auditable**, **versioned**, and **hard to rewrite quietly**.

A WSP Proof Trail has four essential elements:

1. **Bundles** (versioned artifact packages)
2. **Hashes** (cryptographic fingerprints of those bundles)
3. **Timestamps** (public records anchoring those hashes to time under the project’s provenance identity)
4. **An Artifacts Index** (the mapping layer that explains every hash and links to the exact bundle bytes)

Without all four, outsiders cannot reliably verify what a project declared, when it declared it, and how the work evolved.

**6.1 Identity anchors (what must be publicly declared)**

A project claiming WSP compliance **MUST** publish a canonical set of identity anchors that tell the world where the official record lives and how to verify it.

At minimum, the project **MUST** publicly declare:

1. **Canonical Project Home**  
   A stable webpage that acts as the project’s official entry point.
2. **Artifacts Index (Proof Trail) URL**  
   A stable location where the project’s Artifacts Index is published (human-readable page and/or machine-readable files).
3. **Provenance Wallet (or provenance identity)**  
   The project’s official provenance identity used to issue timestamp records.

These declarations **MUST** be easy to find (e.g., directly on the canonical project home). They **SHOULD** be repeated inside each published bundle’s README so bundles remain legible even when shared independently.

**Recommended additional anchors (SHOULD):**

* A page listing **official wallets and their roles** (provenance vs treasury vs admin), to reduce impersonation risk
* A **key rotation / incident policy** page
* A simple **verification guide** for non-technical readers

WSP does not require a specific blockchain or timestamp mechanism. It requires that the identity anchors are stable and that verification is objective.

**6.2 Bundles and hashing**

**6.2.1 What counts as a bundle**

In WSP, the unit that gets committed is a **bundle file**, typically a zip archive. The exact bytes of that zip file are what outsiders verify.

A WSP bundle **MUST** be treated as immutable once timestamped:

* If you change anything substantive, you create a **new bundle**, publish it as a new milestone, and timestamp its new hash.

**6.2.2 Required bundle contents**

To make bundles legible and auditable, every PUBLIC or REDACTED‑PUBLIC bundle **MUST** contain:

* README.md — what this bundle is, what it supports, and where the canonical Proof Trail lives
* MANIFEST.json — machine-readable metadata describing bundle type, version, date, contents, and relationships to other milestones

Bundles **SHOULD** also include, when applicable:

* VERIFY.md — steps to reproduce key outputs (especially for runs/simulations)
* CHANGELOG.md — for version updates that supersede a prior bundle
* REDACTIONS.md — required if any redactions were applied

Hash‑only bundles (HASH‑ONLY) cannot be downloaded publicly by definition, but they **MUST** still have a corresponding Artifacts Index entry (Section 6.4) that describes what the bundle is and why it is not public.

**6.2.3 Hash algorithm**

WSP’s default hash algorithm is **SHA‑256**.

Projects **MAY** use an alternative widely accepted modern hash algorithm if they clearly state it in the Artifacts Index and verification instructions. The algorithm **MUST** be:

* deterministic,
* publicly verifiable, and
* widely available across common tools.

**6.2.4 Practical note: “same contents” vs “same bytes”**

Repackaging the same folder into a zip can produce different bytes (and therefore different hashes). Under WSP, **the bytes are the artifact**.

* If the zip bytes differ, it is a different bundle and must be treated as a new version.
* Projects **SHOULD** avoid “re‑zipping” after publication and instead preserve the exact released bundle file for mirroring.

**6.2.5 Self-hash recursion and outward-facing integrity stamps (Recommended)**

A WSP bundle **MUST NOT** attempt to contain its own cryptographic hash inside any file that is included in that same bundle. Embedding a bundle’s hash inside itself changes the bundle bytes, which changes the hash, indefinitely.

To handle this cleanly, WSP recommends a two-part pattern:

1. **Canonical bundle (the committed artifact):**  
   Create the final bundle (e.g., a ReleasePack ZIP or SubmitPack ZIP), compute its SHA-256 hash, and timestamp that hash under the project’s provenance identity. The canonical bundle does not contain its own hash.
2. **Outward-facing integrity stamp (the recipient checkpoint):**  
   Where a venue or platform allows (e.g., Zenodo description, journal cover letter, “comments to editor” fields, or an outward-facing PDF), the project SHOULD include an integrity stamp that references the canonical bundle’s SHA-256 and timestamp reference. This makes the outward-facing record self-describing and reduces ambiguity about what was committed at the time of submission/publication.

This pattern avoids recursion while strengthening the linkage between the public timestamped record and the outward-facing submission/publication.

**6.3 Timestamping the hash**

The timestamp is the mechanism that makes “this existed by time T” publicly checkable.

**6.3.1 Timestamp requirements**

A WSP timestamp record **MUST** satisfy all of the following:

1. **Contains the exact bundle hash** (or contains a verifiable commitment that unambiguously encodes that hash).
2. **Is publicly verifiable** by third parties without relying on private systems.
3. **Is attributable to the project’s provenance identity**, such that outsiders can confirm it was issued by the official provenance wallet/identity.

The timestamp mechanism is intentionally flexible. A project may timestamp by:

* publishing the hash in a public blockchain record associated with the provenance wallet,
* using a public timestamping service with verifiable proofs, or
* using another public, durable, third‑party verifiable anchoring method.

The important thing is not *how* you timestamp; it is that the timestamp record is objective and checkable.

**6.3.2 What must be recorded in the Artifacts Index**

For every milestone, the Artifacts Index **MUST** include enough information for a verifier to locate and verify the timestamp record, including:

* a **timestamp reference** (e.g., transaction ID, proof ID, or anchor identifier)
* the **timestamp method** (short label describing the mechanism)
* the **provenance identity** used (wallet/address or equivalent)

Projects **SHOULD** also record:

* timestamp date/time in UTC (best effort; derived from the public record)
* network/system used (if relevant)

**6.4 The Artifacts Index: the mapping layer that makes the Proof Trail usable**

A wallet history alone is not a usable Proof Trail. The Artifacts Index is the “translation layer” between raw timestamp records and human/auditor understanding.

**6.4.1 What the Artifacts Index is**

The Artifacts Index is a public mapping from:

**timestamped hash → meaning + location + relationships**

It is where the project fulfills the rule **Explain Every Hash**.

**6.4.2 Required properties**

A WSP-compliant project **MUST** maintain an Artifacts Index that is:

* **Publicly accessible**
* **Continuously updated** (every timestamped hash must be explained)
* **Non-revisionist** (old entries are preserved; updates are additive with versions)
* **Unambiguous** (each entry maps to exactly one bundle hash)

**6.4.3 Required fields (minimum)**

Each Artifacts Index entry **MUST** include at least:

* **Bundle type** (Pack type; e.g., RulesPack, RunPack)
* **Version** (project-defined version label)
* **Bundle hash** (e.g., SHA‑256)
* **Timestamp reference** (how to find the timestamp record)
* **Timestamp method label**
* **Visibility class** (PUBLIC / REDACTED‑PUBLIC / HASH‑ONLY)
* **Description** (plain-language purpose of the bundle)
* **Location**:
  + for PUBLIC / REDACTED‑PUBLIC: at least one download link to the exact bundle bytes
  + for HASH‑ONLY: a clear explanation of why it is not public and what it represents

Projects **SHOULD** also include:

* relationships to other milestones (e.g., “uses DataPack v0.2,” “supersedes RunPack v0.3”)
* mirror links (to reduce link rot)
* an optional short **Artifact ID** (a human-friendly reference label)

**6.4.4 Human-readable and machine-readable index**

For usability and future tooling, projects **SHOULD** publish both:

* a human-readable Artifacts page (webpage), and
* a machine-readable index file (e.g., wsp\_index.json and wsp\_index.csv).

The schema and templates for index files are provided in the appendices.

**6.4.5 Index anchoring (recommended)**

To reduce disputes about whether the index itself was edited retrospectively, projects **SHOULD** periodically:

* hash the machine-readable index file(s), and
* timestamp those hashes as a milestone.

This does not prevent edits; it makes edits auditable.

**6.5 Pack taxonomy (standard bundle types)**

WSP standardizes a set of pack types so outsiders can interpret milestones consistently. Each pack is simply a bundle type with a conventional purpose and expected contents.

All packs **MUST** include README.md and MANIFEST.json when PUBLIC or REDACTED‑PUBLIC. Packs that are HASH‑ONLY **MUST** still have a complete Artifacts Index entry.

**6.5.1 RulesPack**

**Purpose:** Freeze definitions, assumptions, evaluation criteria, scoring rules, or analysis plans **before** outcome‑revealing work.  
**Why it matters:** Reduces post-hoc storytelling by making commitments legible.

Typical contents:

* definitions / criteria / scoring
* scope and exclusions
* “frozen as-of” statement

**6.5.2 DataPack**

**Purpose:** Record data “as received” (raw dataset snapshot, sources, acquisition notes).  
**Why it matters:** Anchors what inputs were used.

Typical contents:

* dataset snapshot or pointers
* acquisition notes and licensing constraints
* integrity notes (optional per-file hashes)

**Note:** DataPacks may be HASH‑ONLY when data cannot be distributed.

**6.5.3 RunPack**

**Purpose:** Capture an analysis/simulation run: code reference, environment, config, logs, outputs, and reproduction guidance.  
**Why it matters:** Enables reproducibility and independent replication.

Typical contents:

* code pointer (commit hash/release tag)
* environment files (requirements, container config, etc.)
* run configs / seeds
* logs and outputs
* VERIFY.md reproduction steps (strongly recommended)

**6.5.4 ReleasePack**

**Purpose:** Public release milestone: preprint/paper, results summary, or major protocol/project release.  
**Why it matters:** This is where claims become cleanly referencable.

Typical contents:

* the released document(s)
* release notes
* references to supporting packs (Rules/Data/Run)

**6.5.5 SubmitPack**

**Purpose:** Record the exact submission package **before** submitting to a venue (journal, conference, review platform).  
**Why it matters:** Creates submission integrity: outsiders can see what existed prior to submission.

Typical contents:

* manuscript and supplementary materials
* reproducibility notes
* submission metadata (venue, date, category)

SubmitPacks may be PUBLIC or HASH‑ONLY depending on venue policy and project strategy.

**6.5.6 DecisionPack**

**Purpose:** Record the outcome of a submission (decision type, decision date, and—where allowed—decision letter/reviewer reports).  
**Why it matters:** Makes outcomes legible while respecting confidentiality constraints.

DecisionPacks may be PUBLIC, REDACTED‑PUBLIC, or HASH‑ONLY.

**6.5.7 RevisionPack**

**Purpose:** Record revisions, response-to-reviewers, and updated manuscripts as new versions.  
**Why it matters:** Preserves an accountable revision trail without overwriting history.

**6.5.8 AuditPack**

**Purpose:** Publish independent verification/replication attempts as first-class artifacts.  
**Why it matters:** Third-party audits are one of the strongest credibility signals.

AuditPacks should reference the exact artifact hashes they audited.

**6.5.9 Optional packs (recommended, not required)**

Projects **MAY** also define:

* **TemplatesPack** (distribution of templates/schemas)
* **IndexPack** (explicit packaged snapshots of index files)
* other project-specific pack types, provided they are clearly defined and used consistently.

**6.6 Interoperability (how WSP composes with existing infrastructure)**

WSP is designed to work with the best existing tools rather than compete with them.

Projects **SHOULD** use stable hosting where appropriate, such as:

* DOI repositories for published releases and papers
* public code hosts for versioned code releases
* preregistration platforms for frozen analysis plans (optional)
* mirror hosting to reduce link rot and improve durability

WSP’s unique role is to provide:

* a unified provenance spine (the provenance identity), and
* a public mapping layer (Artifacts Index) tying everything together across hosts.

Projects **SHOULD** embed their canonical identifiers in public artifacts (e.g., in the README and in papers), including:

* canonical project home URL, and
* Artifacts Index URL, and
* provenance identity reference

This makes the project harder to impersonate and makes verification easier.

**6.7 Minimum Proof Trail compliance (baseline)**

A project is considered **Proof Trail compliant at the baseline level** if it satisfies all of the following:

1. **Publishes identity anchors**  
   Canonical home + Artifacts Index URL + provenance identity.
2. **Publishes at least one PUBLIC bundle**  
   A ReleasePack is the simplest starting point.
3. **Hashes the bundle and timestamps the hash**  
   The timestamp record must be publicly verifiable and attributable to the provenance identity.
4. **Maintains the Artifacts Index with “Explain Every Hash” discipline**  
   Every timestamped hash is mapped to a meaningful entry with links or hash-only explanation.
5. **Follows “No silent rewrite”**  
   New versions are additive; old versions remain referenced and (when feasible) accessible.

This baseline does not require frozen rules, data recording, or submission integrity. Those are defined in later sections and in the compliance tiers. The baseline exists so a project can start immediately with credible, auditable releases—and then deepen rigor over time.

**7. Submission Integrity and Journal Accountability**

Work Speaks Protocol is not only about proving that a bundle existed at a certain time. It is also about making the most credibility‑sensitive moments in research and development **legible**—especially the moments that are commonly disputed later:

* *Were the rules frozen before the outcome was known?*
* *What exactly was submitted, and when?*
* *What happened in review and revision—without relying on rumors or selective retellings?*

WSP addresses these questions with a simple extension of the Proof Trail: **commit before you act, and record outcomes in a policy‑aware way.**

This section defines how WSP handles submissions and decisions without violating confidentiality norms, without turning review into spectacle, and without encouraging hostility toward institutions or individuals.

**7.1 Why submission integrity matters (what it solves)**

Peer review and editorial processes can be valuable. But when a project operates outside the mainstream credibility pipeline—or when work is controversial or novel—two predictable problems arise:

**7.1.1 Project-side integrity disputes (selective storytelling risk)**

Even honest projects can be accused of post-hoc manipulation:

* changing criteria after seeing outcomes,
* quietly revising manuscripts while claiming continuity,
* hiding failed attempts,
* or only publishing the “best” version of the story.

WSP cannot eliminate private experimentation, but it can **raise the cost of revisionist storytelling** by standardizing a visible habit:

**Pre-commit what matters, and preserve a traceable record of what changed.**

**7.1.2 Institution-side opacity (legibility problem)**

Most journals treat peer review as confidential; editorial outcomes are often opaque to outsiders. This is sometimes appropriate, but it means the public cannot easily distinguish:

* scientifically grounded rejection,
* “scope/fit” decisions,
* bandwidth triage,
* and rejection based on convention rather than method quality.

WSP does not claim to “prove” institutional unfairness. It creates the conditions for **constructive legibility over time**: when projects record what they submitted and record outcomes (within policy constraints), patterns become auditable at the institutional level.

**7.1.3 The core WSP stance**

WSP treats submission integrity as an **optional but high-value credibility layer**:

* Projects **MAY** submit to journals as usual.
* Projects **MAY** include WSP provenance references in submissions where appropriate.
* Projects **MUST** remain policy-aware and avoid hostile conduct.
* If a project claims “WSP Submission Integrity,” it **MUST** follow the requirements below.

**7.2 SubmitPack specification (pre-submission commitment)**

A **SubmitPack** is a bundle that commits to the exact submission package **before** it is submitted to a venue (journal, conference, review service, or other evaluation destination).

**7.2.1 What a SubmitPack is for**

A SubmitPack is meant to answer one question unambiguously:

**What exactly existed prior to submission, and when was it committed?**

It is not meant to guarantee acceptance, and it is not meant to weaponize review. It is meant to create a verifiable “before” record.

**7.2.2 Required behavior (normative)**

If a project publishes a SubmitPack milestone, it **MUST**:

1. **Bundle the submission package**  
   The bundle **MUST** correspond to what the project intends to submit (or did submit), including at least the manuscript and any required supplementary materials.
2. **Hash and timestamp before submission**  
   The bundle hash **MUST** be timestamped under the project’s provenance identity **before** the submission is transmitted to the venue.
3. **Index the milestone**  
   The Artifacts Index entry **MUST** include:
   * the bundle hash,
   * timestamp reference and method,
   * submission venue (named or coded),
   * intended submission date (UTC best-effort), and
   * visibility class (PUBLIC / REDACTED‑PUBLIC / HASH‑ONLY) with justification if not PUBLIC.
4. **Preserve history (no silent rewrite)**  
   If the submission package changes, the project **MUST** publish a new SubmitPack version. The old SubmitPack remains part of the record.

**7.2.3 Recommended contents (SHOULD)**

A SubmitPack **SHOULD** include:

* the exact manuscript file(s) submitted (PDF and/or source)
* supplementary information files (if submitted)
* a short submission.json metadata file including:
  + venue name (or venue code),
  + submission date/time (UTC best-effort),
  + submission category/type (article/letter/etc.),
  + a pointer to the related ReleasePack (if one exists), and
  + pointers to supporting packs (RulesPack/DataPack/RunPack) when applicable
* a README.md that explains:
  + what is included/excluded, and why
  + what the project claims this submission represents
  + where to verify the timestamp record

A SubmitPack **MAY** include:

* a cover letter (only if you are comfortable publishing it and policy allows),
* a “WSP Provenance Statement” inside the manuscript or as an appendix, referencing:
  + canonical project home,
  + Artifacts Index URL,
  + provenance identity.

**Note:** Some venues may not want extraneous identifiers in manuscripts. WSP does not require including provenance references inside a submitted paper. It recommends interoperability where feasible.

**7.2.4 Visibility guidance for SubmitPacks**

* **PUBLIC** is preferred when policy and strategy allow.
* **HASH‑ONLY** is acceptable when:
  + the venue has restrictive preprint rules,
  + the project wants to avoid any policy ambiguity, or
  + the submission contains sensitive or proprietary content.

If HASH‑ONLY is used, the project **MUST** still publish a public description of what the bundle is and why it is not public.

WSP also recommends an optional ‘Submission-Embedded Integrity Stamp (SEIS)’ (Section 7.2A) to make the recipient’s stored copy self-describing.

**7.2A Submission-Embedded Integrity Stamp (SEIS) (Recommended)**

Work Speaks Protocol recommends an optional but powerful upgrade to submission integrity: a **Submission-Embedded Integrity Stamp (SEIS)**.

SEIS means placing a short “integrity stamp” inside the submission materials themselves (cover letter and optionally the manuscript) so the **recipient’s stored copy** contains a direct reference back to the same committed submission record in the project’s Proof Trail.

This creates a two-sided checkpoint:

* The project’s public record proves what was committed and when.
* The journal’s stored copy points back to that same commitment record.

**Why SEIS matters**

Even when a project timestamps a SubmitPack before submitting, skeptics can still ask:

* “How do we know the journal received the same submission that you timestamped?”
* “Could you have committed one version publicly and submitted a different version privately?”

SEIS reduces that attack surface by making the submission itself **self-describing**: the submission materials carry a reference to the canonical committed package.

**Critical rule: avoid self-hash recursion (MUST NOT)**

A project **MUST NOT** attempt to embed a bundle’s own cryptographic hash inside any file that is included in that same bundle.

Reason: embedding the hash changes file bytes, which changes the bundle bytes, which changes the hash, indefinitely.

**Recommended canonical reference (SubmitPack ZIP)**

SEIS SHOULD reference the **canonical SubmitPack ZIP** (the complete submission package), not an individual manuscript file.

The canonical SubmitPack ZIP typically contains:

* manuscript PDF and/or source
* figures and supplementary files
* README + MANIFEST
* optional submission metadata

**Recommended workflow (recursion-safe)**

1. **Assemble the canonical submission package** as a SubmitPack ZIP (“SubmitPack-Canonical”).
2. Compute the **SHA-256** of the SubmitPack ZIP and timestamp it under the project’s provenance identity.
3. Publish a Proof Trail entry describing the SubmitPack (hash + timestamp reference + links or hash-only constraints).
4. Embed a SEIS block into submission materials that references the canonical SubmitPack record.

**Where to embed the stamp (SHOULD)**

Because some venues reformat manuscripts, WSP recommends placing the stamp in multiple locations when possible:

* **Cover letter** (recommended; low risk of being stripped)
* **Manuscript** (optional; e.g., a first-page footnote or a final “Provenance (WSP)” section)
* **Submission portal metadata fields** (recommended when available; e.g., “Comments to editor”)

**What the stamp should include (SHOULD)**

A SEIS block SHOULD include:

* Protocol name: “Work Speaks Protocol (WSP)”
* Artifact type: “SubmitPack”
* SubmitPack identifier (Artifact ID or pack name + version)
* Canonical package hash (SHA-256 of the SubmitPack ZIP)
* Timestamp reference (e.g., txid / proof ID)
* Provenance identity (provenance wallet/address or equivalent)
* Date of anchoring (UTC best-effort)
* Proof Trail entry location (URL or identifier)

Official identifiers reference (optional): URL to Identifiers Registry or embedded token contract/mint addresses (see Section 9.3).

**Suggested stamp text (copy/paste)**

Provenance & Integrity (Work Speaks Protocol)  
This submission corresponds to SubmitPack: <NAME>, version <VERSION>.  
Canonical package hash (SHA-256): <HASH>.  
Timestamp reference: <TIMESTAMP\_REF> under provenance identity <PROVENANCE\_ID> on <DATE\_UTC>.  
Official identifiers (optional): <IDENTIFIERS\_REF>.  
Proof Trail entry: <PROOF\_TRAIL\_ENTRY>.

**Disclosure limits (important)**

SEIS improves integrity and legibility, but it does not guarantee that:

* a venue will retain the stamp into the final typeset publication, or
* a portal will not convert uploads into a new internal PDF.

SEIS is primarily for **archival verification**: journals almost always retain the originally uploaded files internally, and portal metadata fields often persist even when manuscripts are reformatted.

**7.3 DecisionPack specification (policy-aware outcome record)**

A **DecisionPack** records the outcome of a submission: desk rejection, rejection after review, revise-and-resubmit, acceptance, or other editorial outcomes.

**7.3.1 What a DecisionPack is for**

A DecisionPack is meant to answer:

* **What happened, when, and under what constraints can we disclose details?**

WSP emphasizes **policy-aware transparency**: record outcomes in a way that preserves integrity without violating confidentiality.

**7.3.2 Required behavior (normative)**

If a project publishes a DecisionPack milestone, it **MUST**:

1. **Record the decision outcome**  
   The pack **MUST** include a decision type and decision date (UTC best-effort).
2. **Be explicit about visibility**  
   The Artifacts Index entry **MUST** label the DecisionPack as:
   * PUBLIC, or
   * REDACTED‑PUBLIC, or
   * HASH‑ONLY,  
     and explain why.
3. **Preserve history**  
   If additional decision information becomes publishable later (e.g., permission granted), the project **MUST** publish a new version rather than editing the old record in place.

**7.3.3 Recommended contents (SHOULD)**

A DecisionPack **SHOULD** include:

* decision.json metadata:
  + venue,
  + decision date,
  + decision type,
  + reference to the corresponding SubmitPack hash/version,
  + disclosure policy notes (e.g., “review text confidential; summary only”),
* a **public decision summary** written by the author(s) describing:
  + the nature of the decision (desk reject, etc.),
  + the stated reasons at a high level,
  + what the project plans to do next (revise, resubmit elsewhere, etc.).

A DecisionPack **MAY** include (only if policy/permission allows):

* the decision letter,
* reviewer reports,
* editorial correspondence.

**7.3.4 Handling reviewer text and confidentiality (critical)**

Many venues treat peer review materials as confidential. WSP therefore recommends a safe default:

* You can always publish **your own materials** (your manuscript versions, your responses, your summary of what occurred).
* You should publish **verbatim reviewer/editor text only if the venue’s policy allows it** or if you have permission.

When in doubt:

* do **not** publish verbatim review text; instead, use **HASH‑ONLY** for the verbatim materials and publish a public summary.

This maintains integrity (you can prove the record existed) while avoiding policy violations.

**7.4 RevisionPack(s) and resubmission trail**

A **RevisionPack** records changes made after a decision—especially after revise-and-resubmit requests—and preserves an accountable history of how the work evolved under critique.

**7.4.1 Required behavior (normative)**

If a project publishes a RevisionPack milestone, it **MUST**:

* publish revisions as **new versions** (no silent rewrite),
* link the revision to the corresponding SubmitPack and DecisionPack in the Artifacts Index, and
* maintain a clear mapping between “what changed” and “why it changed.”

**7.4.2 Recommended contents (SHOULD)**

A RevisionPack **SHOULD** include:

* revised manuscript (and optionally source)
* response-to-reviewers / response-to-editor (authored by the project)
* a concise change summary (“diff notes” or changelog)
* pointers to updated supporting packs if results changed (e.g., new RunPack)

**7.4.3 Resubmission**

Each resubmission (to the same venue or a different venue) **SHOULD** be recorded as a new SubmitPack version and linked to the relevant RevisionPack(s).

This creates a legible, additive trail:  
**Submit → Decision → Revision → Submit → Decision → …**

**7.5 Policy-aware transparency guardrails**

WSP’s goal is accountability through legibility—not public drama.

Projects adopting WSP **MUST** follow these guardrails:

1. **Respect confidentiality where it applies**  
   If a venue’s policies treat reviews as confidential, follow the DecisionPack rules above.
2. **No harassment, no doxxing, no personal targeting**  
   Critique should target artifacts, reasoning, and institutional behavior patterns—not individuals.  
   WSP does not endorse identifying reviewers or directing hostility toward editorial staff.
3. **Be precise about what is known vs inferred**  
   When publishing a decision summary, distinguish:
   * what the venue explicitly stated,
   * what the project is interpreting, and
   * what remains uncertain.
4. **Avoid “acceptance = truth” framing**  
   Acceptance is not proof of correctness; rejection is not proof of error.  
   WSP treats publication outcomes as one input into a broader evidence-based evaluation loop.
5. **Prefer minimal necessary disclosure**  
   Publish the materials required for integrity and auditability; avoid publishing extraneous private communications unless clearly permitted and relevant.

These guardrails are essential for WSP to remain credible and constructive. They ensure that submission integrity strengthens the scientific ideal rather than undermining it.

**8. Compliance Levels (Bronze / Silver / Gold)**

WSP is designed to be adoptable without pretending that every project can—or should—operate at maximum rigor from day one. At the same time, WSP is designed to prevent the most common failure of “credibility systems”: **inflated claims**.

Compliance tiers exist to make one idea enforceable:

**Only claim what an outsider can audit.**

In WSP, “compliance” does not mean “truth,” “correctness,” or “scientific validity.” It means a project is following a transparent, verifiable discipline for how it records and evolves its work.

**8.1 Why tiers exist (honest claims + fast auditing)**

Compliance tiers serve three purposes:

**1) Honest self-description**  
Projects vary widely. Some are early-stage. Some cannot publish data. Some are purely theoretical. Some are building software. A tiered system allows a project to make an accurate claim like:

* “We follow Bronze now; we aim for Silver by Q2.”  
  instead of implying full rigor without the receipts.

**2) Reduced audit cost**  
Auditors and new readers should be able to scan a project’s declared tier and know what to expect:

* What artifact types should exist?
* What integrity commitments are being made?
* Where are the boundaries?

**3) Incentives for better behavior over time**  
The purpose of tiers is not to create hierarchy for its own sake. The purpose is to reward projects that:

* preserve history,
* publish reproducible materials,
* accept critique,
* and make commitments visible before outcomes.

A project MUST meet the Bronze requirements before claiming Silver, and MUST meet Bronze + Silver before claiming Gold.

**Compliance summary (informal quick reference)**

* **Bronze:** Timeline integrity (Proof Trail foundation).
* **Silver:** Rules + inputs integrity (pre-commitments and data/inputs recorded).
* **Gold:** End-to-end evidence chain (reproducibility + submission integrity where applicable + independent audits encouraged).

**8.2 Bronze — Timeline integrity (Proof Trail foundation)**

**Bronze compliance** is the minimum level at which a project can truthfully say it is operating under WSP in a meaningful way. Bronze corresponds to the baseline Proof Trail requirements defined in Section 6.7.

**8.2.1 What Bronze means (the promise)**

A Bronze-compliant project provides outsiders enough structure to verify:

* the project has a canonical identity,
* milestones are real bundles (not vague claims),
* those bundles were committed by time-stamped hashes,
* the mapping from hashes to bundles is legible, and
* history is not being quietly rewritten.

Bronze does **not** require preregistration, frozen rules, public datasets, reproducible runs, or journal submission trails. It creates the credibility floor: *an auditable declared history exists.*

**8.2.2 Bronze requirements (normative)**

To claim Bronze compliance, a project **MUST**:

1. **Publish identity anchors**
   * canonical project home
   * Artifacts Index URL
   * provenance identity (typically a provenance wallet)
2. **Publish at least one PUBLIC bundle**  
   A ReleasePack is the simplest starting point.
3. **Hash + timestamp bundles under the provenance identity**  
   The timestamp record must be publicly verifiable and attributable to the project’s provenance identity.
4. **Maintain an Artifacts Index that explains every hash**  
   Every timestamped hash from the provenance identity must have a corresponding index entry with:
   * bundle type, version, hash
   * timestamp reference/method
   * download link(s) (or hash-only explanation)
5. **Follow “No silent rewrite”**  
   Older versions remain referenced, and updates occur as new versions.

Bronze projects **SHOULD** also publish machine-readable index files and mirror key bundles to reduce link rot, but those are not required for Bronze.

**8.3 Silver — Rules + inputs integrity (pre-commitment discipline)**

Silver compliance is where WSP begins to meaningfully constrain after-the-fact narrative shaping. Silver introduces two disciplines:

* **Freeze the rules before outcomes** (so “what counts as success” is not revised later), and
* **Record the inputs** used to generate results (so “what you relied on” is not ambiguous).

**8.3.1 What Silver adds (the promise)**

A Silver-compliant project provides outsiders enough structure to evaluate:

* whether the project defined key evaluation criteria before outcome-revealing work, and
* whether the project recorded the key inputs used (datasets, corpora, external assets, or other concrete inputs).

Silver reduces the credibility gap between “a nice story” and “a constrained, testable process.”

**8.3.2 Silver requirements (normative)**

To claim Silver compliance, a project **MUST** meet all Bronze requirements and **MUST** additionally:

1. **Publish at least one RulesPack for any major results claim**  
   The RulesPack **MUST** be timestamped **before** the milestone where outcomes are revealed (typically before the associated RunPack or ReleasePack that presents the result).

A RulesPack **MUST** clearly state:

* + the definitions/assumptions that matter to evaluation,
  + the evaluation criteria (what would count as support/failure where applicable), and
  + the scope (what is and is not being claimed).

1. **Publish at least one DataPack (or input equivalent) when results depend on external inputs**  
   The DataPack **MUST** represent the relevant “inputs as received” (or the best feasible snapshot/pointer), and it **MUST** be timestamped at or near acquisition (best effort).
   * If inputs cannot be published due to privacy/licensing, a HASH‑ONLY DataPack is allowed, but the index entry **MUST** explain the constraint clearly.
2. **Link the chain in the Artifacts Index**  
   For each major result, the project **MUST** ensure the index shows relationships such as:
   * RunPack → uses DataPack
   * ReleasePack → supported by RunPack
   * ReleasePack → governed by RulesPack

Silver projects **SHOULD** publish simple reproducibility guidance when possible, but full run reproducibility is a Gold-level expectation.

**8.4 Gold — End-to-end evidence chain (reproducibility + submission integrity + audits)**

Gold compliance is the strongest WSP tier. It is designed for projects that want maximum credibility through:

* run-level reproducibility where feasible,
* clear pre-commitment and outcome trails for submissions when the project submits to venues, and
* a culture that treats independent auditing and replication as first-class contributions.

Gold is not “prestige.” Gold is **maximum auditability**.

**8.4.1 What Gold adds (the promise)**

A Gold-compliant project makes it materially easier for outsiders to:

* reproduce or at least meaningfully inspect the steps that generated key results,
* see exactly what was committed and when (especially before submissions), and
* publish third-party audits against a stable, legible target.

Gold is the tier designed to hold up under the strongest skepticism.

**8.4.2 Gold requirements (normative)**

To claim Gold compliance, a project **MUST** meet all Silver requirements and **MUST** additionally:

1. **Publish RunPacks for major result milestones (where computation/analysis is involved)**  
   For each major results claim that depends on computation, analysis, or simulation, the project **MUST** publish at least one RunPack that includes:
   * an unambiguous code reference (commit hash or release tag),
   * environment or dependency information (as feasible),
   * configuration/parameters (and seeds if relevant),
   * logs and outputs (as feasible), and
   * reproduction instructions (VERIFY.md) when reasonably possible.

If full reproduction is not feasible (e.g., compute constraints), the project **MUST** still provide the best feasible evidence chain and clearly state limitations.

1. **Submission integrity trail when submissions are part of the project’s process**  
   If the project submits work to journals/venues and intends to claim WSP submission integrity, it **MUST** follow Section 7:
   * SubmitPack committed before submission (PUBLIC or HASH‑ONLY)
   * DecisionPack recorded policy‑aware
   * RevisionPack(s) recorded as new versions where applicable

If a project does not submit to venues, it should simply not claim submission integrity; Gold can still apply to its end-to-end evidence chain in other dimensions.

1. **Enable and respect independent auditing**  
   Gold projects **MUST** maintain a stable target for audits by:
   * preserving access to referenced PUBLIC bundles (or clearly marking hash-only constraints), and
   * keeping the index relationships coherent.

Gold projects **SHOULD** actively encourage AuditPacks and replications and treat them as first-class milestones in the Artifacts Index.

**8.4.3 Strongly recommended Gold practices (SHOULD)**

Gold projects **SHOULD** also adopt:

* **Index anchoring** (periodically hash + timestamp the index file itself)
* **Wallet role separation** (provenance wallet separated from treasury/admin)
* **An incident/key-rotation policy** (published and referenced)
* **A credibility scorecard** that is independent of token price and emphasizes audits, replication, and correction behavior

These practices strengthen the operational trustworthiness of the Proof Trail and reduce the impact of inevitable operational risks.

**8.5 How to publish a compliance declaration**

WSP compliance is meaningful only if it is legible and falsifiable. Projects should treat compliance claims as public commitments that outsiders can check.

**8.5.1 Required properties of a compliance declaration**

A compliance declaration **MUST** be:

* **public** (available on the canonical project home or linked prominently),
* **dated**, and
* **specific** (it must name the tier and the WSP spec version being used).

A compliance declaration **MUST** include:

1. **Project name and canonical links**
   * canonical project home
   * Artifacts Index URL
2. **Provenance identity**
   * provenance wallet (or equivalent identity)
3. **Claimed tier**
   * Bronze / Silver / Gold
4. **WSP spec version**
   * e.g., “WSP White Paper v1.0”
5. **Exceptions and limitations (if any)**
   * e.g., “DataPack is HASH‑ONLY due to licensing,”
   * “Full reproducibility not feasible; partial run evidence provided.”

A compliance declaration **SHOULD** also include:

* pointers to the milestone entries that demonstrate compliance (e.g., Artifact IDs or links to specific index rows), and
* a short “how to verify” summary.

**8.5.2 Example compliance declaration (fictional example)**

Below is an example of what a clear compliance declaration looks like. (Values here are illustrative.)

**WSP Compliance Declaration — ExampleLab**

* **Date:** 2025‑12‑20 (UTC)
* **WSP Spec:** Work Speaks Protocol White Paper v1.0
* **Claimed Tier:** Silver
* **Canonical Home:** [https://examplelab.org](https://examplelab.org/)
* **Artifacts Index:** <https://examplelab.org/wsp/artifacts>
* **Provenance Wallet:** 0xEXAMPLEPROVENANCEADDRESS

**Evidence (milestone links):**

* ReleasePack v0.1 — “Initial public release” (PUBLIC)
* RulesPack v0.1 — “Frozen evaluation criteria for Experiment A” (PUBLIC)
* DataPack v0.1 — “Dataset snapshot as received” (HASH‑ONLY due to license; public description provided)
* RunPack v0.1 — “Experiment A run outputs” (PUBLIC; reproducibility partial due to compute)

**Limitations:**

* Full dataset redistribution is not permitted; the DataPack is hash-only.
* Full reproduction requires specialized hardware; partial reproduction steps are provided.

This format makes the claim auditable and makes limitations explicit—exactly the behavior WSP is designed to reward.

**9. The Token Layer: What it is and what it is not**

Work Speaks Protocol includes a token layer because credibility is not only a technical problem. It is also a **coordination problem**.

In a world where independent projects can produce serious work outside institutional pipelines, the missing piece is often not intelligence or effort—it is the ability to:

* form a durable community around the work,
* coordinate what should be tested, audited, or replicated next,
* attract sustained attention without marketing hype,
* and support long-range projects without relying entirely on grants or gatekeepers.

The token is designed to be a **simple participation and coordination instrument** that sits on top of the Proof Trail. The Proof Trail anchors integrity. The token helps coordinate people around integrity.

WSP explicitly rejects the idea that a token’s market price is “proof” of scientific truth. The protocol treats the token as a way for the public to participate in the process of **credibility formation**—grounded in receipts—rather than a substitute for evidence.

**9.1 Token as participation and coordination mechanism**

In WSP, the token’s primary purpose is social and organizational:

* **Participation:** a simple way for people to say, “I’m part of this project’s community.”
* **Coordination:** a way to focus collective attention on what is worth reading, verifying, testing, and replicating.
* **Support:** a mechanism for people to support projects they believe are acting with integrity and producing value.

WSP views the token as a public “membership-like” primitive that can emerge around a project’s Proof Trail. In practice, token participation can support:

* funding audits and replication (directly or indirectly),
* incentivizing structured critique and verification,
* coordinating contributors (technical and non-technical),
* making long-horizon research socially sustainable.

This is not intended to replace traditional funding or peer review. It is intended to open additional pathways that scale with the new reality: **more capable independent work exists than institutions can absorb.**

**9.1.1 Public “voting” and signal—what it really means**

When WSP describes the token as a way for the public to “vote” on projects, it means something specific:

* The public can express *belief that a project is worth attention and support*.
* Communities can coordinate around *what deserves auditing, replication, or deeper study*.
* The signal is a social-economic commitment to the *process*, not a claim of correctness.

In other words, WSP’s “vote” is closer to:

* “this deserves to be tested and taken seriously,”  
  than:
* “this is proven true.”

That distinction is essential. WSP is designed to make it easier for communities to discover value early—especially in high-variance, novel work—while keeping legitimacy anchored to verifiable artifacts.

**9.1.2 Why a token at all (why not just donations or subscriptions?)**

Donations and subscriptions can work for some projects, but they are often:

* fragmented,
* platform-dependent,
* non-portable across ecosystems,
* and weak at coordinating decentralized participation.

A token is a simple, portable primitive that can unify:

* identity (“this is the project token”),
* membership (“I’m part of the network”), and
* coordination (“we support audits, replication, and progress milestones”).

WSP does not claim a token is morally superior. It claims a token is **operationally useful** for public participation in a world where independent work will increasingly matter.

**9.2 Receipts-first norms: how WSP avoids token-only credibility**

WSP includes the token layer while explicitly designing against a common failure mode: communities that become disconnected from evidence.

The protocol’s guiding cultural rule is:

**Receipts first.**

This means:

* A community should treat major claims as meaningful only when they are anchored to **specific artifacts** in the Proof Trail.
* Progress should be evaluated through **versioned milestones**, not narrative momentum.
* Critique should reference **bundle IDs/hashes**, not vibes.
* Auditing and replication should be rewarded and celebrated, not treated as hostility.

**9.2.1 What WSP encourages communities to reward**

WSP aims to shift incentives away from persuasion and toward integrity. Communities operating under WSP should reward:

* **Consistency and continuity** (no silent rewrite, clear evolution)
* **Prediction discipline** where applicable (commitments before outcomes)
* **Reproducibility** (or best‑effort reproduction guidance)
* **Corrections and transparent updates** (integrity under pressure)
* **Third-party audits and replications** (AuditPacks)

This is why WSP treats AuditPacks as first-class artifacts: the strongest credibility is not “I said so,” but “independent people tried to verify it.”

**9.2.2 What WSP discourages (explicitly)**

WSP discourages:

* **Price-as-truth framing** (“the token is up, therefore the theory is correct”)
* **Narrative-only credibility** (marketing substituting for artifacts)
* **Gating verification behind token ownership** (see below)
* **Harassment and personal targeting** (critique must focus on work)

WSP’s intent is a public culture of rational engagement—where it is easy to ask:  
“Where is the bundle that supports this claim?”

**9.2.3 Credibility Scorecard: separating evidence from market mood (recommended)**

To prevent communities from equating token dynamics with scientific validity, WSP strongly recommends that projects publish a **Credibility Scorecard** that is independent of market price.

A Credibility Scorecard can include objective indicators such as:

* current compliance tier (Bronze/Silver/Gold),
* number of PUBLIC milestone bundles,
* number of independent AuditPacks and replications,
* correction behavior (how errors are handled),
* prediction track record (when applicable),
* reproducibility status of key results (full/partial/limited),
* transparency constraints (what is hash-only and why).

This helps the community focus on the things WSP is actually designed to reward: integrity, auditability, and progress.

**9.3 Project identity linking: token ↔ project ↔ wallet ↔ artifacts**

If a token is going to coordinate credible participation, the project must make it hard to impersonate, fragment, or confuse identity. WSP therefore treats identity linking as part of the protocol’s integrity layer.

A WSP project should make the following linkage unambiguous:

1. **The Project** (canonical home)
2. **The Proof Trail** (Artifacts Index + machine-readable index)
3. **The Provenance Identity** (provenance wallet or equivalent identity)
4. **The Token** (official identifier and, where applicable, contract address)

**9.3.1 Required identity publication (recommended best practice)**

A project that uses the token layer **SHOULD** publish an “Official Identifiers” page that includes:

* canonical project home URL
* Artifacts Index URL
* provenance wallet address (and any role-separated wallets, clearly labeled)
* official token name
* official token ticker/symbol (if used)
* official token contract address(es) (if applicable)
* the chain/network(s) where the token exists (if applicable)
* official community channels (and warnings about impersonators)

This is not about branding. It is about preventing identity confusion that destroys credibility.

**9.3.1A Genesis Identity Declaration (Recommended)**

Work Speaks Protocol recommends a simple best practice for making project identity linkage difficult to spoof and easy to audit: a **Genesis Identity Declaration**.

A Genesis Identity Declaration is an early, public record made under the project’s provenance identity that declares the project’s canonical identifiers in a single place. It creates a “root-of-trust checkpoint” that ties together:

* the provenance identity used for timestamps,
* the official token identifiers (if a token layer exists), and
* the canonical location of the project’s Proof Trail (Artifacts Index).

This helps prevent identity confusion and impersonation, especially in token ecosystems where fake addresses and lookalike pages are common.

**What it is (in practice)**

A Genesis Identity Declaration is a WSP milestone recorded under the provenance identity that includes, at minimum:

* Protocol name: “Work Speaks Protocol (WSP)”
* Project identifier (project name or short ID)
* Canonical home (project’s official site)
* Artifacts Index location (Proof Trail URL or identifier)
* Provenance identity (wallet/address used for timestamps)
* Official token identifiers (if applicable):
  + token name/symbol
  + network/chain
  + contract address / mint address

**When to do it (SHOULD)**

Projects SHOULD publish a Genesis Identity Declaration:

* at project launch, or
* at token launch, or
* at the first public release milestone,

and SHOULD update it only via additive milestones (no silent rewrite). If official identifiers change (e.g., wallet migration, token contract migration), publish a new identity declaration milestone referencing the change.

**Why this matters**

WSP credibility depends on identity continuity:

* A Proof Trail is only useful if outsiders know which wallet and which index are official.
* A token is only safe to interact with if outsiders can verify the official contract/mint addresses.

A Genesis Identity Declaration provides a durable “first anchor” tying these together in a way that is publicly auditable.

**Recommended structured payload (readable encoding)**

When recording identity declarations (and other milestones), WSP recommends using a consistent, human-readable payload format (whether in an on-chain memo field, a signed statement, or a published log). Common implementations include a self-transfer transaction carrying a UTF-8 memo (EVM-style) or a memo-program transaction (Solana-style).

Example payload format (illustrative):

protocol=WSP | project=PROJECT\_ID | canonical=CANONICAL\_HOME | artifacts=ARTIFACTS\_INDEX | provenance=PROVENANCE\_ID | token\_evm=CONTRACT\_ADDR | token\_spl=MINT\_ADDR

Projects may shorten fields, but should keep the structure consistent across milestones.

**Important note**

A Genesis Identity Declaration strengthens identity linkage and reduces impersonation risk, but it does not replace the Artifacts Index. The Artifacts Index remains the canonical mapping layer for all milestone bundles and proofs.

**9.3.2 Embedding WSP identifiers in public artifacts (recommended)**

WSP recommends embedding canonical identifiers in:

* the project website header/footer,
* README files inside bundles,
* and (where appropriate) published papers or releases.

At minimum, bundles should point to:

* canonical home, and
* Artifacts Index URL.

This ensures that even if a bundle is mirrored, reposted, or shared independently, outsiders can still trace it back to the canonical record.

**9.3.3 Separating roles: provenance vs treasury vs operations**

WSP treats the provenance identity as special: it is the “signature” of the Proof Trail.

Projects should assume that any wallet used for frequent operations is more exposed. Therefore, projects using a token layer **SHOULD**:

* use a **dedicated provenance wallet** for timestamping milestones,
* keep it protected and minimally used,
* and use separate wallets for treasury, operations, and deployment.

This reduces the chance that an operational mistake compromises the credibility spine of the project.

(Operational security guidance is expanded in Section 11.)

**9.4 Disclosures and boundaries (risk + integrity)**

Because tokens attract attention, they also attract misinterpretation. WSP therefore includes explicit boundaries to keep the token layer aligned with integrity.

**9.4.1 What the token does not represent**

Unless a project explicitly states otherwise, a WSP project token is not intended to represent:

* ownership of the project or its outputs,
* equity, dividends, or profit entitlement,
* guarantees of returns,
* or proof that claims are correct.

WSP does not dictate a particular token model. But it does require that the project **disclose clearly what the token is for**, and what it is not for, so the community understands the intended role.

**9.4.2 “Do not gate verification” rule (strong recommendation)**

A core integrity mistake in many communities is to place essential verification behind paywalls or membership gates.

WSP therefore strongly recommends:

* **PUBLIC artifacts and verification steps should remain publicly accessible.**
* If token gating is used at all, it should apply to optional community experiences (e.g., private discussion spaces), not to the proof trail required to audit major claims.

If people cannot verify claims without holding a token, the protocol’s credibility function collapses and the community drifts toward belief-based dynamics.

**9.4.3 Market behavior is not a credibility metric (explicit)**

WSP encourages projects and communities to treat token price as:

* a measure of attention and participation,
* not a measure of truth.

A healthy WSP community should be able to say:

* “The token is doing well, but we still demand audits.”
* “The token is doing poorly, but the receipts remain strong.”
* “We changed our view because the evidence changed, not because the chart changed.”

This discipline is part of what distinguishes WSP from purely speculative token cultures.

**9.4.4 Handling fraud, impersonation, and forks**

Tokens attract impersonators. WSP therefore encourages projects to:

* publish a canonical “Official Identifiers” page (Section 9.3.1),
* anchor changes (like new contract addresses or migrations) in the Proof Trail as milestone bundles,
* and maintain a public “wallet history” record for continuity.

If a fork or impersonation occurs, the project should respond with:

* a clear public statement,
* a proof-trail anchored milestone referencing official addresses, and
* a reminder that only the canonical Artifacts Index + provenance identity defines the official record.

**9.4.5 Minimal disclosure expectations (recommended)**

A WSP project using the token layer should publish, at minimum:

* an explanation of token purpose (participation/coordination/support),
* a receipts-first statement (claims map to artifacts),
* an official identifiers page (wallets + token identity),
* and a community conduct standard (no harassment, no doxxing).

These disclosures protect the project, protect participants, and preserve the protocol’s credibility.

**10. Incentives, Governance Primitives, and Community Design**

A Proof Trail makes integrity legible—but it does not automatically produce attention, replication, or progress. In practice, the projects that thrive over time are the ones that can coordinate people around the work: testing it, criticizing it constructively, reproducing it, improving it, and explaining it.

The token layer is WSP’s coordination primitive. This section outlines practical ways a WSP community can use that primitive to reward verification and build credibility without drifting into hype or coercion.

WSP does not require any specific governance model (DAO, on-chain voting, etc.). It provides **minimal governance primitives** and a **receipts-first culture** so communities can scale responsibly if and when they choose.

**10.1 Incentivizing audits and replication**

The highest-value contributions to credibility are often not flashy. They are the work of careful verification:

* checking that a bundle matches its hash,
* re-running an analysis under the stated configuration,
* reproducing a figure or table,
* stress-testing assumptions,
* attempting falsification, and
* documenting what worked and what didn’t.

WSP treats this work as first-class contribution, and encourages communities to **reward verification explicitly**.

**10.1.1 Audit bounties (recommended)**

A WSP project community **SHOULD** consider publishing structured “Audit Bounties” tied to specific milestones.

A good audit bounty is:

* **artifact-specific** (points to exact bundle hash/version),
* **scope-bounded** (“reproduce Figure 3,” not “validate the whole theory”),
* **deliverable-based** (requires a published AuditPack), and
* **evaluation-transparent** (states who determines whether the deliverable is met).

A typical audit bounty specification includes:

* **Target:** Artifact ID / bundle hash / version (e.g., RunPack v0.4)
* **Objective:** What to verify or reproduce (one paragraph)
* **Required deliverable:** an **AuditPack** containing:
  + steps taken,
  + environment notes,
  + outputs produced,
  + comparison notes,
  + discrepancies (if any),
  + conclusion (“reproduced / partially reproduced / failed to reproduce”)
* **Acceptance criteria:** what counts as “completed”
* **Disclosure rules:** what must be public vs what may be hash-only
* **Reward mechanism:** defined by the project/community (WSP is agnostic)

The key is not the reward structure. The key is that verification work becomes **visible, comparable, and archived** as part of the Proof Trail.

**10.1.2 Replication challenges (recommended)**

Replication is often best coordinated as a “challenge” rather than an open-ended request.

A replication challenge is a public call for attempts to reproduce a specific result, with clear rules:

* the target artifact(s) to use,
* what constitutes a successful replication (tolerances, metrics),
* what constitutes a partial replication, and
* what evidence must be published.

Replication challenges work especially well when results are sensitive to:

* computational environments,
* randomness / seeds,
* hardware constraints, or
* dataset versions.

Publishing multiple independent AuditPacks against the same milestone is one of the strongest credibility signals a project can accumulate.

**10.1.3 Funding the “boring work” (a cultural rule)**

Most communities overweight novelty and underweight validation. WSP is designed to reverse that.

Projects and communities operating under WSP **SHOULD** treat these contributions as core:

* reproducibility improvements (better VERIFY.md, cleaner configs),
* dataset packaging and documentation,
* automated verification tools (hash checkers, index validators),
* and independent audit writeups.

If a community only rewards “new claims,” it will drift toward narrative competition. If it rewards verification, it will drift toward the scientific ideal.

**10.2 Reputation through work: auditors and replicators as first-class actors**

A central purpose of WSP is to make credibility **portable** and **behavior-based**. This applies not only to projects, but also to the people and groups who audit them.

In a mature WSP ecosystem, “auditors” and “replicators” become a parallel legitimacy layer:

* Their reputation is built on published AuditPacks,
* Their work is verifiable, and
* Their credibility compounds over time.

**10.2.1 AuditPacks as a reputation primitive**

An AuditPack is not just “a blog post about whether something seems right.” It is a structured artifact that can be:

* checked,
* compared,
* cited,
* and built upon.

WSP communities **SHOULD** treat AuditPacks as the default output format for serious verification.

At minimum, an AuditPack **SHOULD** contain:

* scope (what was checked, what was not checked)
* exact artifacts audited (hashes/versions)
* verification steps
* environment details (when relevant)
* outputs produced
* differences observed
* conclusions and remaining uncertainty

Auditors **SHOULD** be precise about what they did and did not verify. The point is legibility, not rhetorical dominance.

**10.2.2 Healthy adversarial collaboration**

WSP encourages an attitude that is rare in many communities:

**Adversarial collaboration is a feature.**

A strong project should want its work stress-tested, and a strong auditor should want to produce a report that the project can learn from—even if the outcome is negative.

WSP communities should normalize statements like:

* “This did not reproduce, and here is the exact step where it diverged.”
* “We found an error; the project fixed it and published a correction.”
* “We disagree, but we’re auditing the same artifacts.”

The goal is not social victory. The goal is to reduce uncertainty.

**10.2.3 Curators and explainers matter too**

Not everyone can run simulations or reproduce code. But a healthy WSP ecosystem also needs:

* **curators** who organize evidence trails,
* **explainers** who translate artifacts for broader audiences, and
* **reviewers** who compare versions and identify what changed.

WSP treats these roles as valuable when they remain anchored to receipts (bundle hashes, index entries, reproducibility artifacts), not personality.

**10.3 Governance: lightweight by default, scalable if needed**

WSP does not mandate a governance system. Governance is a tool, not the core.

The core is the Proof Trail. Governance should serve the Proof Trail—never replace it.

**10.3.1 Start simple: “governance by publication”**

A project can run WSP credibly with **no on-chain governance** and no complex voting.

The minimum viable governance model is:

* publish what you’re doing,
* record major decisions as artifacts,
* keep wallet roles clear,
* and invite audits.

In other words: **governance by receipts**.

Examples of decisions that communities **SHOULD** record as milestones (publicly or hash-only as needed):

* changes to official wallet addresses or wallet roles
* changes to where the canonical Artifacts Index is hosted
* major changes to token identity (migrations, contract upgrades, re-issuance)
* changes to compliance claims (e.g., Bronze → Silver)
* major changes to project scope (“pivot” decisions)

Recording these decisions as bundles preserves continuity and reduces ambiguity.

**10.3.2 If you scale governance, keep it constrained**

If a project/community chooses to add formal governance mechanisms, WSP recommends a conservative posture:

* keep governance scope narrow at first,
* prefer slow, transparent processes,
* and ensure governance actions do not create incentives to hide or distort the Proof Trail.

Common governance domains that can be useful (when done carefully):

* allocating funds toward audits/replication challenges
* choosing which milestones to prioritize next
* approving operational changes (e.g., new mirror hosts, new tooling)
* community moderation and code-of-conduct enforcement

WSP strongly recommends separating:

* **scientific evaluation** (evidence-based, artifact-linked)  
  from
* **resource allocation** (what the community funds or focuses on)

Governance should not become “voting on truth.” It should become “voting on what to test, replicate, and build.”

**10.3.3 Operational safety (recommended constraints)**

Any community managing resources—whether through a token treasury or otherwise—**SHOULD** adopt operational safeguards such as:

* role-separated wallets (provenance vs treasury vs operations)
* multi-signature controls for treasury actions
* time delays for sensitive changes (to reduce rushed decisions)
* a published incident response policy
* public, auditable records for major administrative actions (recorded as bundles)

These are not ideological choices; they are practical protections against predictable failure.

**10.4 Anti-manipulation posture (realistic, not utopian)**

WSP assumes manipulation exists. Every coordination mechanism—institutions, media, grants, markets, tokens—can be gamed.

WSP’s approach is not “ban manipulation.” It is:

* anchor legitimacy to receipts,
* reward verification,
* and keep decision-making legible.

**10.4.1 What WSP can mitigate**

WSP can reduce the impact of manipulation by making it harder to manufacture credibility without real artifacts.

A project cannot credibly claim progress under WSP without producing:

* verifiable bundles,
* a coherent index,
* and a traceable update history.

A community cannot credibly sustain high-integrity status if it:

* discourages audits,
* hides artifacts,
* or replaces evidence with narrative enforcement.

**10.4.2 What WSP cannot eliminate**

WSP cannot eliminate:

* speculative behavior around tokens,
* social herding,
* hype cycles,
* or attempts to influence perception.

This is why WSP insists on an explicit separation:

* **Token participation is not proof.**
* **Proof Trail receipts are the audit surface.**

**10.4.3 Recommended norms that keep communities healthy**

WSP communities **SHOULD** adopt norms such as:

* “If you make a major claim, link the artifact.”
* “If you criticize a claim, reference the bundle/version you’re criticizing.”
* “Audits and replications are celebrated, even when negative.”
* “Corrections strengthen credibility.”
* “No harassment, no doxxing, no personal targeting.”

And projects **SHOULD** publish a Credibility Scorecard (independent of token price) so newcomers can orient around:

* compliance tier,
* public artifacts count,
* audit/replication count,
* correction history,
* reproducibility status,
* and disclosure constraints (what is hash-only and why).

**10.4.4 A simple guiding test**

A WSP community can sanity-check itself with one question:

**If the token disappeared tomorrow, would the credibility of the work remain legible through the Proof Trail?**

If the answer is “yes,” the community is anchored to evidence.

If the answer is “no,” the community has drifted toward belief-based legitimacy—and WSP is not being practiced as intended.

**11. Implementation Guide**

**How to actually run Work Speaks Protocol in the real world**

This section translates WSP from concept into operational practice. The goal is not to force a single software stack or a single blockchain choice. The goal is to make WSP **easy to adopt**, **easy to verify**, and **hard to misuse**.

WSP works best when you treat it like a discipline, not a one-time announcement:

* you publish a canonical home,
* you maintain a legible Artifacts Index,
* you release bundles as immutable milestones,
* you timestamp their hashes under a stable provenance identity, and
* you build credibility through continuity.

**11.1 Minimal viable adoption**

**Start small, ship receipts, then deepen rigor**

WSP is intentionally designed so a project can become meaningfully compliant quickly, without waiting for perfect tooling, journal outcomes, or a full case study.

**11.1.1 Minimal viable WSP (Bronze in practice)**

A new project can adopt WSP at Bronze level with a single clean release cycle:

1. **Choose your provenance identity**
   * Create a dedicated provenance identity (typically a wallet address).
   * Treat this identity as the “official signature” of your Proof Trail.
2. **Publish a canonical project home**  
   Your canonical home is the public entry point that states:
   * what the project is,
   * where the Proof Trail (Artifacts Index) is, and
   * what the official provenance identity is.
3. **Create an Artifacts Index (start with one entry)**  
   Even a one-entry index is enough to begin—so long as it is accurate and publicly accessible.
4. **Publish your first ReleasePack**  
   Create a bundle (zip) that contains:
   * the released document(s) or deliverable(s),
   * README.md and MANIFEST.json,
   * any supporting materials you want public.
5. **Hash the bundle (SHA‑256) and timestamp the hash**
   * Compute the SHA‑256 of the zip file.
   * Create a public timestamp record under the provenance identity.
6. **Update the Artifacts Index (“explain every hash”)**  
   Add an index entry with:
   * hash, timestamp reference, method label, and download link(s),
   * visibility class (PUBLIC),
   * a short description of what it is and why it matters.
7. **Do a public verification pass**  
   Pretend you are an outsider:
   * download the bundle from the link,
   * recompute the hash,
   * confirm it matches the index,
   * confirm the timestamp record contains that hash and is attributable to the provenance identity.

If this works, you are operational. Everything else is an upgrade path.

**11.1.2 “Silver upgrade” path (when you start making major claims)**

Silver is the point where you stop relying on narrative to justify evaluation and start relying on **pre-commitment discipline**.

Upgrade to Silver when you are producing major results claims by adding:

* **RulesPacks** before outcome-revealing work  
  Freeze definitions and evaluation criteria before running key analyses.
* **DataPacks / Input packs** at acquisition time (best effort)  
  Record data “as received,” or use hash-only with a public constraint explanation.

Silver is achieved by making the “inputs and rules” legible—not by writing longer explanations.

**11.1.3 “Gold upgrade” path (when you want maximum auditability)**

Gold is about making the process reproducible and the integrity disputes boring.

Upgrade toward Gold by adding:

* **RunPacks** for major computational results  
  Include code pointers, environment notes, configs, logs, outputs, and reproduction guidance where feasible.
* **Submission Integrity** only if you actually submit to venues and want that accountability layer  
  Use SubmitPack → DecisionPack → RevisionPack trails policy-aware.
* **Invite independent audits**  
  Encourage AuditPacks as first-class artifacts.

Gold is achieved by building a complete evidence chain—then letting third parties test it.

**11.2 Canonical infrastructure stack**

**A simple architecture that composes with existing tools**

WSP works best when you separate roles:

1. **Canonical Home** (identity + explanations)
2. **Artifact Hosting** (where files live)
3. **Provenance Identity** (where hashes are timestamped)
4. **Artifacts Index** (the mapping layer tying it all together)

A recommended “canonical stack” looks like this:

* **Canonical Home:** your project website (stable, easy to find)
* **Artifact Hosting:** one or more stable hosts (e.g., DOI archive for releases, repository releases for code, project site mirrors)
* **Provenance Identity:** a dedicated provenance wallet/identity for timestamp records
* **Artifacts Index:** published on the canonical home (human-readable + machine-readable)

**11.2.1 Redundancy is part of credibility**

Links rot. Platforms change. Projects relocate. WSP therefore recommends that projects:

* keep at least **two mirrors** for major public bundles (where feasible), and
* keep the Artifacts Index updated when mirrors change.

WSP credibility depends on outsiders being able to retrieve exact bundle bytes. If access disappears, verification becomes harder, and the project’s credibility suffers (even if the timestamp remains valid).

**11.2.2 Keep the timestamp method flexible, but verification objective**

WSP is chain-agnostic. What matters is that an outsider can objectively verify:

* the timestamp record contains the committed hash, and
* the record is attributable to the project’s provenance identity.

Choose a timestamp method you can maintain and that your audience can verify without specialized access.

**11.3 Website requirements**

**What pages to publish so outsiders can actually verify you**

A WSP project’s website is not a marketing layer; it is a **verification interface**.

At minimum, a WSP project SHOULD publish three pages:

**11.3.1 Protocol page**

A short page describing:

* that the project follows Work Speaks Protocol,
* the project’s claimed compliance tier (Bronze/Silver/Gold),
* the official provenance identity, and
* a link to the Artifacts Index and verification guide.

This page should be the “front door” for credibility.

**11.3.2 Artifacts / Proof Trail page**

This is the most important page in the entire system.

It should contain:

* the Artifacts Index (human-readable),
* a link to machine-readable index files (recommended), and
* clear instructions for “how to verify a milestone.”

Outsiders should be able to answer in under a minute:

* What did you publish?
* Where is it?
* How do I verify it?

**11.3.3 Token / Community page (if you use the token layer)**

If your project uses a token, publish a page that explains:

* what the token is intended to represent (participation/coordination/support),
* what it is *not* intended to represent (truth, guaranteed returns, etc.),
* official identifiers (token identity, official links), and
* receipts-first norms (claims must map to artifacts).

This page is critical to prevent drift into token-only credibility.

**11.3.4 Strongly recommended supporting pages**

To reduce confusion and improve operational trustworthiness, projects SHOULD also publish:

* **Verification guide:** a non-technical “how to verify” walkthrough
* **Official identifiers page:** official wallets (with roles) and official community channels
* **Security / continuity policy:** key rotation and incident response policy
* **Case Studies page (optional in v1):** a place for living updates that don’t belong inside a static PDF

These pages don’t need to be long. They need to be clear.

**11.4 Human-readable and machine-readable index**

**Make the Proof Trail legible to both people and tools**

The Artifacts Index is where WSP becomes usable. Without it, outsiders can’t easily interpret a timestamp history.

**11.4.1 Human-readable index (minimum requirement)**

A human-readable index should allow scanning and filtering by:

* date
* bundle type (RulesPack, RunPack, ReleasePack, etc.)
* version
* visibility class (PUBLIC / REDACTED‑PUBLIC / HASH‑ONLY)

Each entry should include:

* description (1–3 sentences)
* bundle hash (SHA‑256)
* timestamp reference + method label
* download link(s) (or hash-only explanation)
* relationships to other milestones where relevant

**11.4.2 Machine-readable index (recommended)**

Machine-readable files (for example, JSON and CSV) make it easier to build tools that:

* validate index integrity,
* automatically verify hashes,
* diff versions,
* and power AI-assisted auditing.

WSP strongly recommends publishing:

* an index JSON file, and
* an index CSV file,  
  and updating them in lockstep with the human-readable page.

**11.4.3 Index update discipline (“explain every hash” operationalized)**

A simple operational rule keeps the system coherent:

**Every time you timestamp a hash, you update the index before you move on.**

This prevents a common failure mode where the wallet history becomes filled with unexplained hashes and the mapping layer falls behind.

**11.4.4 Index anchoring (recommended)**

To make the index itself auditable over time, projects SHOULD periodically:

* hash the machine-readable index file(s), and
* timestamp those hashes as milestones.

This does not prevent edits; it makes edits traceable and discourages quiet retroactive rewriting of the map.

**11.5 Security and operational continuity**

**Protect the provenance identity and preserve continuity under failure**

WSP credibility depends on continuity. Operational failures are not theoretical: wallets get compromised, keys get lost, links rot, and maintainers disappear. WSP encourages projects to treat these risks as part of the protocol, not an afterthought.

**11.5.1 Provenance wallet discipline (strong recommendation)**

Projects SHOULD treat the provenance wallet like a signing key, not a daily-use wallet.

Recommended posture:

* hardware wallet protection (where feasible)
* minimal signing frequency (only milestone timestamps)
* avoid using the provenance wallet for treasury operations
* publish wallet role separation clearly

**11.5.2 Role separation (recommended)**

A mature project SHOULD separate:

* **Provenance identity** (timestamping)
* **Treasury** (funds management)
* **Operations** (day-to-day actions, deployments, admin work)

This reduces the blast radius of operational errors.

**11.5.3 Key rotation and incident policy (recommended)**

Projects SHOULD publish a simple policy answering:

* What happens if the provenance wallet is compromised?
* How do we migrate to a new provenance identity while preserving continuity?
* How do we communicate the change publicly?
* How do we prevent impersonation during the transition?

A key rotation policy does not reduce credibility. It increases credibility, because it shows the project expects real-world failures and has a coherent response.

**11.5.4 Continuity under maintainer loss**

WSP encourages projects to consider:

* redundant access to critical accounts (where appropriate),
* shared documentation for release processes,
* archived copies of major bundles,
* and a public “continuity plan” for what happens if the original maintainer disappears.

This matters because credibility is not just about honesty—it is also about survivability.

**11.5.5 Operational checklist (recommended)**

Before publishing any milestone, a project SHOULD confirm:

* the bundle is final and preserved as exact bytes
* hash computed and recorded correctly
* timestamp record is publicly verifiable
* Artifacts Index entry is complete and accurate
* download link(s) work
* relationships to other milestones are correct
* old versions remain available and referenced

This checklist is not bureaucracy. It is the difference between a proof trail that can be audited and a proof trail that becomes confusing.

**12. Case Studies**

**Proof by demonstration**

WSP is designed to be more than a theory about credibility. It is designed to be **practiced**, audited, and improved in public. This section documents how WSP is applied in real releases and outlines how longer-running case studies are maintained over time.

A key principle of WSP case studies is that a static PDF is always a **snapshot**. The canonical, continuously updated source of truth for case study status is the project’s **Artifacts Index (Proof Trail)**.

**12.1 Case Study 0 — This white paper as a WSP artifact**

The first and simplest demonstration of WSP is that the protocol document itself is released under the protocol.

**12.1.1 What this case study demonstrates**

Case Study 0 demonstrates the minimum WSP loop:

* A public bundle exists (the ReleasePack for this white paper).
* The bundle hash is published and timestamped under the project’s provenance identity.
* The Artifacts Index maps the hash to an explanation and to the exact downloadable bytes.
* Any outsider can verify the integrity of the release.

This proves **integrity and timeline**, not correctness of the ideas inside the document.

**12.1.2 What is released (ReleasePack contents)**

A typical WSP release of the white paper SHOULD include:

* the white paper PDF,
* the source document (e.g., DOCX/Markdown),
* any templates or schemas referenced in the appendices (optional but recommended),
* README.md with canonical links and verification instructions, and
* MANIFEST.json describing the release bundle, its version, and its relationships.

**12.1.3 How to verify Case Study 0 (public checklist)**

An outsider can verify the release without trusting the author by following these steps:

1. **Locate the ReleasePack entry**  
   Go to the project’s Artifacts Index and find the entry for the WSP white paper ReleasePack (the entry should include the bundle version, SHA‑256 hash, timestamp reference, and download mirrors).
2. **Download the exact bundle file**  
   Download the ReleasePack zip from the index link. If multiple mirrors exist, you may use any mirror—but you MUST verify the bytes match the hash.
3. **Compute the SHA‑256 hash of the zip file**  
   Use any standard tool. Examples:
   * macOS/Linux: shasum -a 256 <bundle.zip> or sha256sum <bundle.zip>
   * Windows (PowerShell): Get-FileHash <bundle.zip> -Algorithm SHA256
4. **Compare your computed hash to the Artifacts Index**
   * If it matches: you have verified the bundle bytes.
   * If it does not match: treat the bundle as unverified and report the discrepancy.
5. **Verify the timestamp record**  
   Using the timestamp reference in the Artifacts Index:
   * confirm the record contains the same bundle hash, and
   * confirm the record is attributable to the project’s provenance identity.
6. **Optional integrity checks**
   * Open the bundle and confirm it contains README.md and MANIFEST.json.
   * Confirm the README points back to the canonical project home and Artifacts Index.
   * Confirm the manifest’s metadata aligns with the Artifacts Index entry (type/version/date/relationships).

**12.1.4 What this verification does and does not establish**

If the verification passes, an outsider has established:

* the release bundle existed by the timestamp time,
* the bundle bytes match the declared hash, and
* the hash was declared under the project’s provenance identity.

This does **not** establish that the protocol is “correct,” that the token model is “good,” or that any claims in the paper are true. It establishes that the project is practicing **transparent, auditable release discipline**, which is the credibility foundation WSP is built on.

**12.2 Case Study 0B — Submission integrity demonstration**

*(Optional in v1; expanded over time)*

WSP’s submission integrity layer is valuable, but it must be practiced in a policy-aware way. Case Study 0B demonstrates the submission pipeline using a controlled submission scenario (for example, submitting a document for formal review, review service feedback, or a journal—depending on what is feasible and appropriate).

**12.2.1 What this case study demonstrates**

This case study demonstrates:

* how to commit a SubmitPack **before** submission,
* how to record the outcome as a DecisionPack, and
* how to preserve the revision chain without violating confidentiality.

It also demonstrates that WSP can create accountability **without requiring journal cooperation**.

**12.2.2 The minimal submission integrity chain**

A minimal WSP submission integrity chain is:

1. **SubmitPack** (timestamped before submission)
2. **DecisionPack** (recording the decision outcome, policy-aware)
3. **RevisionPack** (if revisions occur)
4. **SubmitPack vNext** (if resubmitted)

Each step is additive, versioned, and indexed.

**12.2.3 Confidentiality-safe disclosure rules**

Peer review is often confidential. Therefore, Case Study 0B is conducted under these constraints:

* The project MAY publish the submission bundle publicly if permitted.
* If uncertain, the project SHOULD default to **HASH‑ONLY** for the full submission package while still publishing:
  + the submission metadata, and
  + a public summary of what was submitted.

For decisions:

* The project SHOULD publish a public decision summary (authored by the project).
* Verbatim reviewer reports or decision letters MUST be published only if policy/permission allows.
* If not allowed, the project SHOULD store the verbatim materials as **HASH‑ONLY** (timestamped) and publish a public summary.

This preserves integrity without creating policy violations.

**12.2.4 What success looks like**

Success for this case study is not “acceptance.” Success is:

* a verifiable pre-submission commitment,
* a legible outcome record, and
* a coherent revision chain if applicable.

**12.3 Case Study 1 — Absolute Relativity implementation and end-to-end pipeline**

WSP is intended to support high-variance novelty without requiring prior institutional endorsement. Case Study 1 demonstrates WSP in a real independent research program context, where credibility must be earned through traceable work over time.

This case study is accompanied by a public essay that explains the implementation plan and rationale (“WSP – Initial AR Plan”), released as a WSP artifact alongside the protocol materials.

**12.3.1 What this case study demonstrates**

Case Study 1 demonstrates the core promise of WSP:

* An independent research project can build public credibility through a Proof Trail,
* the public can coordinate around the work through a token community layer, and
* major claims can be anchored to reproducible artifacts and version discipline.

It also demonstrates WSP’s stance toward institutions:

* WSP is not anti-academic and not anti-journal,
* but WSP does not require journals to confer legitimacy in order for work to be auditable and testable.

**12.3.2 “Gold chain” structure (the target standard)**

The target structure for an end-to-end “Gold” record is:

* **RulesPack** (definitions / evaluation criteria frozen before outcome-revealing steps)
* **DataPack** (inputs as received, or hash-only with clear constraints)
* **RunPack** (code pointer + environment + config + logs + outputs; reproduce where feasible)
* **ReleasePack** (public articulation of claims linked to supporting packs)
* **SubmitPack** (pre-submission commitment, when submitting)
* **DecisionPack** (policy-aware outcome record)
* **RevisionPack(s)** (revision trail, no silent rewrite)
* **AuditPack(s)** (independent verification/replication reports)

This chain is designed so that outsiders can audit integrity even when they disagree with conclusions.

**12.3.3 How the token layer integrates in the case study**

In Case Study 1, the token layer is treated as:

* a participation primitive (“I’m part of this community”),
* a coordination layer (audits/replications prioritized and rewarded), and
* a support mechanism for sustained research and infrastructure development.

Crucially, the token layer is explicitly constrained by receipts-first norms:

* major claims must map to artifacts,
* updates must be versioned,
* and credibility is earned through continuity and independent audits—not through narrative momentum.

**12.3.4 What “progress” means in this case study**

For WSP, progress is defined operationally as:

* increased completeness and coherence of the Proof Trail,
* improved reproducibility and verification tooling,
* more independent audits and replications over time, and
* honest correction behavior when errors are found.

Journal outcomes and institutional responses are part of the record when relevant, but they are not treated as the sole legitimacy mechanism.

**12.4 Living case-study updates: the website is the canonical record**

Because WSP is a versioned protocol practiced over time, case studies evolve. A static PDF cannot capture all ongoing milestones, audits, revisions, and mirrors.

Therefore:

* The canonical, continuously updated record of case study status MUST be maintained on the project’s Artifacts Index.
* This white paper SHOULD be treated as a snapshot of the protocol at the time of release.
* Case study pages SHOULD link directly to specific Artifacts Index entries (Artifact IDs or hash entries) so readers can verify.

When meaningful changes occur (completed submission trails, significant audit outcomes, major tooling releases), the project SHOULD publish a new protocol release (e.g., v2) and record it as a new ReleasePack, preserving the old versions without silent rewrite.

**13. Roadmap and Ecosystem Vision**

Work Speaks Protocol is not intended to be a one-off framework used by a single project. The long-term vision is an ecosystem: many independent projects adopting a shared, auditable discipline—and a public that can participate in credibility formation through verification, replication, and constructive critique.

This section describes how WSP can evolve from “one project practicing receipts-first integrity” into a broader decentralized network of serious projects, along with the tooling and integrations that would make WSP easy to use at scale.

**13.1 From “one project” to a decentralized independent research network**

WSP becomes most powerful when it becomes a pattern that many projects can follow. The ecosystem vision is:

* **Projects become legible** because their work is published as versioned milestones with an auditable Proof Trail.
* **Communities become constructive** because participation is anchored to receipts and rewarded through verification work.
* **Auditors and replicators gain reputation** because their audits are also published as verifiable artifacts (AuditPacks).
* **Institutions remain valuable**, but independent legitimacy pathways exist in parallel, reducing single-point gatekeeping.

**13.1.1 The “WSP project pattern” (standardized project anatomy)**

A mature WSP project tends to converge on a common anatomy:

1. **Canonical Home**  
   A stable public entry point stating official identifiers and the project’s credibility posture.
2. **Proof Trail**
   * An Artifacts Index (human-readable + machine-readable)
   * Versioned bundles with hashes and timestamps
3. **Provenance Identity**  
   A stable identity anchoring timestamp records and preventing ambiguity.
4. **Receipts-first culture**  
   The norm that claims map to artifacts, and corrections are additive and visible.
5. **Optional token community layer**  
   A participation/coordination mechanism that amplifies verification rather than replacing it.

The key is not any one piece—it is the coherent combination. When many projects converge on this pattern, the ecosystem becomes searchable, comparable, and auditable.

**13.1.2 Credibility compounding as a network effect**

WSP’s ecosystem vision depends on a network effect: credibility compounds not only for each project, but for the ecosystem as a whole.

As more projects adopt WSP:

* new readers learn a single verification method that works across projects,
* auditors can reuse tooling and practices,
* norms become enforceable socially (“link the artifact”), and
* cross-project credibility becomes easier to compare without relying on institutional signals.

Over time, the ecosystem develops a shared language of legitimacy:

* “This project is Bronze/Silver/Gold.”
* “This claim maps to a RulesPack + RunPack + independent AuditPack.”
* “This project’s correction behavior is strong.”
* “This project has a long, coherent Proof Trail with minimal drift.”

This is the opposite of a hype-based attention economy. It is an evidence-indexed attention economy.

**13.1.3 Inter-project composability**

A decentralized ecosystem becomes stronger when projects can build on each other transparently. WSP supports this by encouraging:

* clear dependency references between packs (“this RunPack uses that DataPack”),
* stable artifact identifiers, and
* machine-readable indexes that can be ingested by tools.

In a mature WSP ecosystem, “citation” becomes more concrete:

* not only “paper cites paper,”  
  but also
* “result cites run,”
* “claim cites dataset snapshot,”
* “analysis cites configuration.”

This supports higher-integrity collaboration across independent teams.

**13.1.4 A note on pluralism and novelty**

WSP is designed to reduce the cost of taking novelty seriously without lowering standards.

It does this by shifting the default posture from:

* “Ignore anything outside the mainstream,”  
  to
* “If it’s interesting, demand receipts and audits.”

That is a constructive pluralism: many ideas can exist, but they must be anchored to verifiable work to earn sustained attention.

**13.2 Tooling roadmap**

**What needs to exist for WSP to be easy at scale**

WSP can be practiced today with basic tools (zip, SHA‑256 hashing, a timestamp method, and a website). But for WSP to scale to large numbers of projects and auditors, the ecosystem benefits from standardized, open tooling.

Below is a practical roadmap of tooling categories that would make WSP adoption frictionless.

**13.2.1 WSP CLI (bundle creation, hashing, manifests, and signing)**

A lightweight command-line tool could standardize:

* bundle packaging with deterministic conventions,
* manifest generation (MANIFEST.json),
* optional per-file checksums inside bundles,
* and simple verification reports.

Core CLI commands (conceptual examples):

* wsp bundle create
* wsp bundle hash
* wsp index add-entry
* wsp verify bundle
* wsp verify index

The goal is not to centralize control. The goal is to standardize mechanics so outsiders can verify without guesswork.

**13.2.2 Index validators and “Explain Every Hash” enforcement**

The most common operational failure is a wallet full of hashes that are not explained, or an index full of broken links.

Tooling should make it easy to run checks like:

* “Every timestamped hash in the provenance wallet has an index entry.”
* “Every PUBLIC entry has a working download link.”
* “Downloaded bundle hash matches the index hash.”
* “Relationships reference valid artifact IDs/hashes.”

This could exist as:

* a standalone validator,
* a GitHub Actions workflow,
* or a website build step.

The goal is to make WSP compliance easy to maintain, not easy to claim.

**13.2.3 Audit tooling (reproducibility scaffolding)**

Auditors often spend time reconstructing environments. Tooling can reduce that cost by encouraging projects to publish:

* container specs (where feasible),
* dependency files,
* standardized run metadata,
* and “expected outputs” checks.

Audit tooling should support:

* “run pack reproduction mode” (best effort),
* “diff outputs” comparisons,
* and “audit report templates” that publish directly as AuditPacks.

**13.2.4 AI-assisted verification (helpful, not authoritative)**

AI tools can make WSP easier to use, as long as they remain subordinate to receipts:

* auto-summarize what a bundle contains,
* compare versions and highlight meaningful diffs,
* generate “what changed and why” drafts from manifests and changelogs,
* guide non-technical users through verification steps,
* and help auditors identify likely failure points in reproduction.

In the long term, this can reduce the barrier to public participation while keeping the core integrity constraint intact: **AI explanations must point back to verifiable artifacts.**

**13.2.5 Public discovery and indexing (ecosystem directory)**

A decentralized ecosystem still benefits from discovery layers—ideally open, plural, and non-authoritative.

A WSP directory could list projects and their declared anchors:

* canonical home URL,
* provenance identity,
* index URLs,
* compliance tier claim,
* and recent milestones.

Importantly, discovery should not become a centralized gatekeeper. It should be an index that can be mirrored and forked—consistent with WSP’s ethos of interoperability and survivability.

**13.3 Integrations and future improvements**

**Where WSP can connect to existing ecosystems (and where it can go next)**

WSP is intentionally designed to compose with existing infrastructure. The strongest version of WSP is not one that replaces everything, but one that stitches together a coherent, auditable history across many systems.

**13.3.1 Repository and DOI integrations**

Projects can already host artifacts on stable platforms. WSP becomes stronger when releases are mirrored across:

* DOI repositories (long-term reference stability),
* code hosting releases (developer workflows),
* and project websites (canonical mapping layer).

A mature workflow is:

* publish to stable host → hash bundle → timestamp hash → index entry points to stable host(s).

This reduces disputes about “what version is real” and makes public verification simple.

**13.3.2 Data repository corroboration (future)**

A long-term improvement is the ability to corroborate “data as received” more strongly.

Today, a project can timestamp that it *has* a dataset snapshot by time T. In the future, data repositories could optionally provide:

* signed receipts for downloads,
* versioned dataset fingerprints,
* and stable dataset snapshot identifiers.

That would strengthen the “inputs integrity” story and reduce ambiguity about dataset versions—especially for high-impact results.

WSP does not require this, but it is a natural integration path.

**13.3.3 Open review and review-service integrations**

Peer review confidentiality varies widely across venues. WSP’s policy-aware model can integrate with:

* open peer review journals,
* transparent peer review platforms,
* and review services that allow publishing reports.

Where policies allow, DecisionPacks can include:

* verbatim decision letters and reviews,
* response-to-reviewers,
* and revision trails.

Where policies do not allow, WSP can still preserve integrity through hash-only storage plus public summaries.

**13.3.4 Standardized “WSP metadata blocks” in papers (optional)**

Over time, WSP could converge on a minimal, optional metadata block that papers include, similar to a “data availability statement,” but for provenance:

* canonical project home,
* Artifacts Index URL,
* and optionally a reference to the relevant ReleasePack / RunPack.

This is not required and should never violate venue policies, but it can make research objects easier to verify and connect.

**13.3.5 Protocol evolution: stable core, iterative improvement**

WSP should evolve carefully. The credibility of the protocol depends on stability and clarity.

A mature evolution posture is:

* keep the core principles stable (no silent rewrite, explain every hash, receipts-first),
* version the protocol itself as ReleasePacks,
* document changes in changelogs,
* and treat improvements as additive rather than disruptive.

Future improvements should focus on reducing friction and increasing auditability—not on increasing complexity.

**14. FAQ / Objections**

This section addresses common questions and objections about Work Speaks Protocol (WSP). The goal is to make the protocol easier to interpret, harder to misrepresent, and clearer about what it can—and cannot—guarantee.

**14.1 “Isn’t this just Zenodo / OSF / a DOI / GitHub releases?”**

WSP is designed to **interoperate** with existing infrastructure, not replace it.

Platforms like Zenodo, OSF, and GitHub are excellent at hosting and distributing artifacts. DOI systems are excellent at providing stable references. What they do *not* provide by default is a unified, protocolized, project-level **Proof Trail** that:

* timestamps the exact bytes of a released bundle under a stable provenance identity,
* maps each timestamped hash to a legible description and relationship graph, and
* preserves an additive history where “what changed and when” is visible across all project outputs.

WSP is best understood as a **layer above distribution platforms**:

* distribution platforms host files,
* WSP ties those files into a single coherent, auditable chain of declared work.

In practice, a healthy WSP project will often use Zenodo/DOIs and GitHub releases as mirrors—then use WSP to create the master record that makes those releases easy to verify and compare.

**14.2 “Couldn’t someone just timestamp fake hashes and pretend it proves something?”**

Yes—anyone can timestamp arbitrary strings. That is why WSP’s credibility does not come from timestamping alone.

In WSP, a timestamped hash matters only when it is paired with:

* a publicly readable **Artifacts Index** that explains what the hash is,
* a downloadable **PUBLIC bundle** whose bytes match the hash (or an explicitly justified hash-only record), and
* coherent version discipline and relationships that hold up under scrutiny over time.

A wallet full of unexplained hashes is not a Proof Trail. It is noise.

WSP’s credibility emerges from *behavior over time*:

* Does the project consistently publish bundles?
* Are they verifiable?
* Are claims mapped to artifacts?
* Are corrections additive rather than silent rewrites?
* Do independent auditors produce AuditPacks that confirm or challenge results?

If a project only timestamps hashes but cannot produce the corresponding bundles or a coherent mapping layer, it will not accumulate credibility under WSP.

**14.3 “How do I verify a WSP project if I’m not technical?”**

WSP verification is intentionally designed to be mechanical.

At the simplest level, you can verify a single milestone in minutes:

1. Go to the project’s **Artifacts Index**.
2. Pick a **PUBLIC** entry (e.g., a ReleasePack).
3. Download the exact bundle zip.
4. Compute the SHA‑256 hash of the zip file using a standard tool.
5. Confirm the hash matches the index.
6. Use the timestamp reference to confirm the hash exists in the public timestamp record tied to the project’s provenance identity.

You do not need to understand the science to verify the integrity of a milestone. You only need to verify that:

* the bundle is real,
* the hash matches, and
* the timestamp is attributable to the project’s declared provenance identity.

Higher-level evaluation (whether results are valid) can then be approached incrementally—with help from auditors, replication attempts, and AI-assisted explanation—without relying on credentials alone.

**14.4 “Doesn’t public participation (via tokens) create manipulation or speculation problems?”**

Public participation can create manipulation and speculation pressures—this is true in every coordination system, including institutions, media, and markets.

WSP’s approach is not to claim that manipulation disappears. It is to make credibility **harder to manufacture without receipts** and to encourage norms that reward verification rather than persuasion.

WSP reduces token-driven distortion by insisting on:

* **Receipts-first norms:** major claims map to artifacts, not hype.
* **Audit incentives:** replication and audits are first-class contributions.
* **A credibility scorecard separate from price:** compliance tier, audit count, correction behavior, reproducibility status, and transparency constraints are all more meaningful than market mood.

WSP also strongly recommends that projects **do not gate verification** behind token ownership. If people must hold a token to inspect evidence, the system drifts toward belief-based legitimacy and away from the scientific ideal.

Finally, WSP’s stance is explicit:

* A token is a participation and coordination tool.
* A Proof Trail is the audit surface.
* Price is not proof.

**14.5 “What about peer review confidentiality? Am I allowed to publish decision letters or reviewer reports?”**

Many journals and venues treat peer review content as confidential. WSP is designed to be **policy-aware**, not reckless.

WSP’s safe default is:

* You may always publish your **own** materials (your manuscript versions, your responses, your public summary of events).
* You should publish **verbatim reviewer/editor text only if the venue’s policy allows it** or if you have permission.

When in doubt, WSP recommends:

* publishing a **public decision summary** (authored by the project), and
* storing verbatim review materials as **HASH‑ONLY** (timestamped) if you want an integrity record without publishing the content.

This preserves the accountability benefits of a Proof Trail while reducing policy and ethical risk.

**14.6 “Is WSP anti-academia or anti-journal?”**

No.

WSP treats institutions as important—and also recognizes that:

* institutional bandwidth is limited,
* novelty is hard to evaluate quickly, and
* legitimacy pathways need to scale as AI expands who can do serious work.

WSP does not require institutional endorsement to build credibility, but it is compatible with institutional processes. In many cases, WSP can strengthen institutional trust by making project behavior more legible and by producing cleaner artifacts for review and replication.

The intention is not to replace peer review. The intention is to reduce the global cost of verification and increase transparency around how work evolves.

**14.7 “Does WSP prove a theory is correct?”**

No. WSP is not a truth oracle.

WSP can prove (when practiced correctly):

* a bundle existed by a certain time,
* the bundle you downloaded matches the committed hash,
* the timestamp record is attributable to the project’s provenance identity, and
* the project’s declared history is additive and traceable.

WSP cannot prove:

* correctness of conclusions,
* that no private exploration occurred before publication,
* or that community consensus equals truth.

What WSP *does* do is make it easier for others to test, audit, replicate, and challenge claims—because the artifacts and history are legible.

**14.8 “What stops a project from rewriting history by just starting over?”**

Nothing stops a project from starting a new project identity. But restarting has a real credibility cost in WSP:

* it breaks continuity,
* it discards the compounding Proof Trail, and
* it forces the project to rebuild trust from zero.

This is one of WSP’s core incentives: credibility compounds when you preserve history, including errors and corrections. A project that routinely “rebrands” instead of versioning transparently signals low integrity and will struggle to accumulate long-term credibility under the protocol.

**15. Conclusion / Call to Action**

Work Speaks Protocol (WSP) is a simple idea expressed as a disciplined practice:

* **Publish verifiable artifacts.**
* **Commit to them publicly with timestamps under a stable provenance identity.**
* **Maintain a legible map of what each commitment means.**
* **Let communities coordinate around verification—not around persuasion.**

In the age of AI, output is easy. Claims are cheap. What becomes scarce—and therefore valuable—is **integrity that can be verified**, and **coordination that stays anchored to evidence**.

WSP exists to make those two things easier:

1. **The Proof Trail** makes declared work auditable over time (no silent rewrite, explain every hash).
2. **Submission integrity** makes key credibility moments legible, policy-aware, and historically traceable.
3. **The token layer** makes serious public participation possible—so attention, replication, critique, and support can scale beyond institutional bottlenecks, without collapsing into hype.

WSP does not promise truth. It promises something more fundamental: a way to build legitimacy through **auditable behavior**, so that truth-seeking projects—especially independent and novel ones—can be evaluated in public without requiring prior permission.

**15.1 If you want to adopt WSP: start with one honest milestone**

You do not need a DAO, a token, a journal submission, or a complex toolchain to begin. You need one verifiable release and a coherent mapping layer.

A minimal “start today” path is:

1. **Publish a canonical project home**  
   Make it the entry point that declares official identifiers.
2. **Create an Artifacts Index**  
   Even a one-entry index is enough if it is legible and accurate.
3. **Choose a provenance identity**  
   Use a dedicated provenance wallet/identity for timestamping milestones.
4. **Publish your first ReleasePack (PUBLIC)**  
   Bundle your artifact(s) into a zip with README.md and MANIFEST.json.
5. **Hash the bundle and timestamp the hash**  
   Then add the entry to the Artifacts Index with download link(s) and timestamp reference.
6. **Verify it like an outsider would**  
   If someone can’t verify the milestone in minutes, refine the packaging and index until they can.

That is Bronze-level WSP in practice. Once you can do that consistently, you can deepen rigor toward Silver and Gold as your work demands.

**15.2 If you want to contribute to a WSP ecosystem: become an auditor, replicator, or curator**

A WSP ecosystem becomes valuable when verification becomes normal.

If you want to contribute, you can:

* publish **AuditPacks** that reproduce results or verify integrity,
* create **replication challenges** and produce structured reports,
* build and share **verification tooling** (index validators, hash checkers, diff tools),
* write **curation summaries** that link claims to artifacts (receipts-first explanation),
* translate complex work into understandable forms—while always pointing back to the evidence trail.

WSP’s credibility grows fastest when independent verification is rewarded socially and structurally.

**15.3 A note to institutions: WSP is compatible with the scientific ideal**

WSP is not an anti-institution movement. It is an attempt to strengthen the scientific ideal under new conditions.

Institutions are still essential for many reasons: training, funding, infrastructure, and deep review culture. But the world is changing:

* more people can do serious work outside institutions,
* verification costs must scale, and
* legitimacy pathways must become more legible.

WSP offers institutions an opportunity, not a threat:

* a way to receive better-organized artifacts,
* a way to reduce disputes about version history and pre-commitment integrity, and
* a way to build public trust through clearer, evidence-linked records—where policy allows.

WSP also respects real constraints: confidentiality and ethics matter. The protocol’s submission integrity layer is designed to be policy-aware, not disruptive.

**15.4 The core invitation**

WSP is an open protocol and a public norm proposal:

**Make credibility auditable. Make participation possible. Let the work speak.**

If you believe the future will include many serious independent projects—and that society benefits when scientific engagement becomes more public, more evidence-linked, and more accountable—then the next step is straightforward:

* adopt the Proof Trail discipline,
* publish one clean milestone,
* invite verification,
* and build credibility through continuity.

WSP will evolve through practice: through real releases, audits, replications, corrections, and community design that stays anchored to receipts.

**Appendix A — Glossary**

This glossary defines key terms used throughout the Work Speaks Protocol (WSP) white paper. Where helpful, entries include brief notes on how the term is used in WSP.

**Normative keywords**

**MUST / SHOULD / MAY**  
Used as normative keywords indicating requirement level:

* **MUST**: required for compliance with the stated rule.
* **SHOULD**: strongly recommended; deviations should be rare and explained.
* **MAY**: optional.

**A**

**Artifact**  
Any concrete output relevant to a project’s claims, integrity, reproducibility, or evaluation. Examples include manuscripts, datasets, code snapshots, run configurations, logs, figures, analysis outputs, submission packages, decision records, revision responses, and audit reports.

**Artifacts Index**  
The public mapping layer that makes the Proof Trail usable. It maps each timestamped hash to:

* what it represents (bundle type/version/purpose),
* where to download the exact bundle bytes (if PUBLIC), or why it is HASH‑ONLY, and
* how it relates to other milestones (e.g., “this RunPack uses that DataPack”).

**Artifact ID**  
A human-friendly identifier for a specific milestone entry (optional but recommended). An Artifact ID improves communication and citation (e.g., “ARP‑RUN‑0021”) without replacing hashes.

**Audit**  
A verification activity performed by a third party (or an independent internal party) to check integrity, reproduction, replication, or consistency of a project’s artifacts.

**AuditPack**  
A bundle type used to publish an audit or replication attempt as a first-class WSP artifact. An AuditPack should reference the exact artifacts (hashes/versions) that were audited and include steps, environment notes, outputs, discrepancies, and conclusions.

**B**

**Bundle**  
A packaged file—typically a zip archive—treated as a single immutable unit for verification. WSP hashes the *exact bytes* of the bundle file and timestamps that hash. If bundle bytes change, it is a new bundle and must be versioned as a new milestone.

**Bundle Hash**  
The cryptographic hash (e.g., SHA‑256) computed from the exact bundle file bytes. Used to verify that a downloaded bundle matches the committed milestone.

**C**

**Canonical Home**  
The project’s official public entry point (usually a website page) that declares identity anchors: where the Proof Trail lives, what the official provenance identity is, and how to verify the project.

**Case Study**  
A concrete demonstration of WSP in use, maintained through the Proof Trail. Case studies evolve over time; a PDF is a snapshot, while the Artifacts Index is the canonical live record.

**Compliance Declaration**  
A public statement by a project declaring:

* which WSP tier it claims (Bronze/Silver/Gold),
* which WSP spec version it follows,
* its canonical home and Artifacts Index URLs, and
* its provenance identity, plus any constraints/exceptions (e.g., HASH‑ONLY data).

**Compliance Tier**  
A defined level of WSP adoption:

* **Bronze:** timeline integrity via Proof Trail discipline,
* **Silver:** adds rules + inputs integrity (RulesPack + DataPack discipline),
* **Gold:** adds end-to-end evidence chain (RunPacks, submission integrity where applicable, and audit-first culture).

**Credibility Scorecard**  
A project-published, evidence-based summary of credibility signals that is intentionally separate from token price. Typical fields include compliance tier, number of public milestones, independent audits, correction behavior, reproducibility status, and disclosure constraints.

**D**

**DataPack**  
A bundle type used to record data or inputs “as received,” including dataset snapshot(s), acquisition notes, and constraints. DataPacks may be PUBLIC or HASH‑ONLY depending on licensing/privacy.

**DecisionPack**  
A bundle type used to record the outcome of a submission (desk rejection, rejection after review, revise-and-resubmit, acceptance, etc.) in a policy-aware way. May be PUBLIC, REDACTED‑PUBLIC, or HASH‑ONLY.

**H**

**HASH‑ONLY**  
A visibility class indicating that a milestone hash is declared and timestamped, but the corresponding bundle is not publicly distributed—typically due to privacy, licensing, security, or policy constraints. HASH‑ONLY is permitted in WSP, but requires clear explanation in the Artifacts Index.

**Hash**  
A cryptographic fingerprint of a file. If the file changes, the hash changes. WSP uses hashes to enable public integrity checks.

**I**

**Identity Anchors**  
The minimal public identifiers a WSP project must publish so outsiders can verify it:

* canonical home,
* Artifacts Index location(s), and
* provenance identity (typically a provenance wallet).

**M**

**Manifest (MANIFEST.json)**  
A machine-readable file included in PUBLIC/REDACTED‑PUBLIC bundles that describes bundle type, version, date, contents, and relationships (e.g., “this ReleasePack is supported by RunPack v0.4”). Used to support tooling and reduce ambiguity.

**Milestone**  
A declared event in the project’s evolution represented by a bundle and its timestamped hash, mapped in the Artifacts Index. Milestones form the project’s Proof Trail.

**P**

**Pack**  
A standardized bundle type representing a milestone category with conventional expectations (e.g., RulesPack, RunPack, ReleasePack). Packs help outsiders interpret what a bundle is supposed to contain and what role it plays in the evidence chain.

**Proof Trail**  
The project’s public, auditable record of declared milestones. At minimum it consists of:

* bundles (or bundle descriptions),
* bundle hashes,
* timestamp records anchoring those hashes under the project’s provenance identity, and
* an Artifacts Index explaining each hash and linking to bundle bytes (or explaining hash-only constraints).

**Provenance Identity**  
The public identifier used to authenticate and anchor the Proof Trail. In WSP this is typically a wallet address (the Provenance Wallet). The specific technology may vary, but the identity must be publicly verifiable and stable over time.

**Provenance Wallet**  
The designated wallet address used to issue official timestamp records for WSP milestones. This wallet functions as the “signature spine” of the Proof Trail and should be protected and used minimally.

**PUBLIC**  
A visibility class indicating the bundle is publicly downloadable so outsiders can re-hash and verify it directly.

**R**

**REDACTED‑PUBLIC**  
A visibility class indicating the bundle is publicly downloadable but contains documented redactions (with a clear REDACTIONS.md or equivalent explanation).

**ReleasePack**  
A bundle type used for a coherent public release milestone: a white paper, preprint, paper release, major update, or other referencable deliverable. ReleasePacks should reference the supporting evidence chain (Rules/Data/Run packs where applicable).

**RevisionPack**  
A bundle type used to record revisions, response-to-reviewers, and updated manuscripts as additive versions—without overwriting old versions.

**RulesPack**  
A bundle type used to freeze definitions, assumptions, evaluation criteria, scoring rules, or analysis plans before outcome-revealing steps. RulesPacks support pre-commitment discipline.

**RunPack**  
A bundle type used to capture run-level details of analysis/simulation/computation: code pointers, environment/dependencies, configuration, logs, outputs, and reproduction guidance where feasible.

**S**

**SHA‑256**  
A widely used cryptographic hash algorithm. WSP’s default hash algorithm is SHA‑256, though WSP is hash-algorithm agnostic if the algorithm is modern, deterministic, and publicly verifiable.

**Submission Integrity**  
A WSP practice that commits the exact submission package (SubmitPack) before submission and records outcomes (DecisionPack) in a policy-aware way, preserving revision trails (RevisionPacks) without silent rewrite.

**SubmitPack**  
A bundle type used to record the exact submission package before it is submitted to a venue. SubmitPacks support a verifiable “before” record and reduce retrospective disputes.

**T**

**Timestamp**  
A publicly verifiable record that a given hash existed by a certain time and was declared under a project’s provenance identity. The method can vary; the requirement is objective third-party verifiability.

**Timestamp Method**  
A short label describing how timestamping is implemented (e.g., “onchain\_tx,” “public\_anchor”). The method label helps readers understand how to verify the timestamp record.

**Token Layer / Community Token**  
The participation and coordination layer of WSP. The token is intended to support community formation, attention coordination, audit incentives, and project support—while remaining anchored to the Proof Trail. The token is not treated as proof of truth.

**V**

**Verification**  
The process of checking that:

* a downloaded bundle’s hash matches the Artifacts Index hash, and
* the timestamp record contains that hash and is attributable to the project’s provenance identity.

**Version**  
A label distinguishing one milestone bundle from another. In WSP, version discipline is a credibility primitive: if substantive content changes, the version must change, and the old version remains referenced.

**W**

**Work Speaks Protocol (WSP)**  
A standard for building public credibility and community coordination around serious projects by combining:

* an auditable Proof Trail (bundles + hashes + timestamps + mapping), and
* a token-enabled participation layer constrained by receipts-first norms.

**Receipts-first**  
A cultural and operational norm in WSP: claims and discussions should point to verifiable artifacts. Coordination is encouraged, but it must remain anchored to public receipts.

**Appendix B — Bundle Templates**

**Pack structures, required files, and MANIFEST.json schema**

This appendix provides **recommended, copy-ready bundle structures** (“packs”) so projects can publish WSP artifacts consistently. These templates are designed to make milestones:

* easy to verify (hash → timestamp → index → download),
* easy to interpret (standard pack types),
* easy to audit (clear inputs, rules, and run evidence),
* and easy to evolve (versioned, additive history).

WSP does not require a specific folder layout. However, **standard layouts reduce audit friction** and help third parties verify work faster.

**Template notation:**  
This appendix uses angle‑bracket variables like <PROJECT\_ID> or <VERSION> to indicate fields a project would substitute when creating its own bundles. These are **template variables**, not unfinished placeholders in this white paper.

**B.1 General bundle conventions (applies to every Pack)**

**B.1.1 One Pack per bundle (recommended)**

WSP strongly recommends **one pack type per zip bundle** (e.g., one RulesPack per bundle, one RunPack per bundle). This keeps milestones legible and makes index relationships clean.

If you must combine multiple pack types (early-stage convenience), treat it explicitly as a composite bundle and document it clearly in the README.md and MANIFEST.json. (Composite bundles are discouraged for mature projects.)

**B.1.2 Canonical top-level files (required for PUBLIC / REDACTED‑PUBLIC bundles)**

Every PUBLIC or REDACTED‑PUBLIC bundle **MUST** include:

* README.md
* MANIFEST.json

Every bundle **SHOULD** include when applicable:

* VERIFY.md (especially for RunPacks and AuditPacks)
* CHANGELOG.md (if this version supersedes an earlier one)
* REDACTIONS.md (required if any redactions exist)
* LICENSES/ or LICENSES.md (recommended if redistributing third-party materials)

Hash-only (HASH‑ONLY) bundles are not downloadable, but their corresponding Artifacts Index entry must still describe what the bundle is and why it is not public (Section 6.4).

**Integrity Note (Recommended)**  
Projects MAY include a short INTEGRITY\_NOTE.md inside PUBLIC release bundles (ReleasePacks) and other canonical bundles to clarify a common point that can confuse auditors:

* a bundle cannot contain its own hash without recursion, and
* the canonical bundle’s hash and timestamp are recorded externally in the Proof Trail (Artifacts Index).

Recommended file:

INTEGRITY\_NOTE.md (optional, but helpful)

Suggested content (copy/paste):

**Integrity Note (Work Speaks Protocol)**  
This ZIP is the canonical archived bundle committed under WSP. A bundle cannot include its own SHA-256 hash inside itself because adding it would change the bytes and therefore change the hash.  
The canonical SHA-256 hash and timestamp reference for this bundle are recorded in the project’s Proof Trail / Artifacts Index entry for this release.

This note improves clarity without introducing recursion.

**B.2 README.md template (recommended minimum content)**

A strong README.md makes audits dramatically easier. Recommended structure:

**README.md**

* **Title**: <PROJECT\_NAME> — <PACK\_TYPE> <VERSION>
* **What this is** (1–3 sentences)
* **Why it matters** (1–3 bullets; what claim or milestone it supports)
* **How to verify**
  + Link to canonical home
  + Link to Artifacts Index entry (or index page)
  + State: “Verify by hashing this zip and matching the index hash; then check the timestamp reference.”
* **Relationships**
  + “Uses: …”
  + “Supports: …”
  + “Supersedes: …” (if applicable)
* **Contents overview**
  + brief list of key folders/files
* **Constraints (if any)**
  + redactions, licensing, privacy limitations
* **Contact / issue reporting** (optional)

**B.3 MANIFEST.json schema (bundle-level metadata)**

The manifest is intended to be machine-readable and stable across pack types.

**B.3.1 Manifest design rule**

Because the **bundle hash is computed over the entire zip bytes**, the manifest **MUST NOT** attempt to embed the final bundle hash inside itself (that creates circularity). Instead, the manifest contains metadata and (optionally) per-file hashes.

**B.3.2 Recommended MANIFEST.json (v1 schema)**

Below is a recommended baseline schema. Projects may extend it, but should keep the core fields stable.

{

"wsp\_spec": {

"name": "Work Speaks Protocol",

"version": "1.0"

},

"project": {

"name": "<PROJECT\_NAME>",

"project\_id": "<PROJECT\_ID>",

"canonical\_home": "<CANONICAL\_HOME\_URL>",

"artifacts\_index": "<ARTIFACTS\_INDEX\_URL>",

"provenance\_identity": "<PROVENANCE\_WALLET\_OR\_ID>"

},

"bundle": {

"pack\_type": "<PACK\_TYPE>",

"version": "<VERSION>",

"created\_utc": "YYYY-MM-DDTHH:MM:SSZ",

"visibility": "PUBLIC",

"title": "<SHORT\_TITLE>",

"description": "<SHORT\_DESCRIPTION>",

"hash\_algorithm": "SHA-256"

},

"relationships": {

"uses": [

{ "artifact\_ref": "<ARTIFACT\_ID\_OR\_HASH>", "note": "optional note" }

],

"supports": [

{ "artifact\_ref": "<ARTIFACT\_ID\_OR\_HASH>", "note": "optional note" }

],

"supersedes": [

{ "artifact\_ref": "<ARTIFACT\_ID\_OR\_HASH>", "note": "optional note" }

]

},

"contents": [

{

"path": "README.md",

"role": "bundle\_readme",

"sha256": "<OPTIONAL\_PER\_FILE\_SHA256>",

"media\_type": "text/markdown"

}

],

"disclosures": {

"redactions": false,

"redactions\_note": "",

"hash\_only\_reason": "",

"licensing\_notes": ""

}

}

**B.3.3 Notes on key fields**

* project.provenance\_identity: the official identity used for timestamps (often a wallet address).
* bundle.visibility allowed values:
  + PUBLIC
  + REDACTED-PUBLIC
  + HASH-ONLY *(typically used only in the Artifacts Index entry; downloadable bundles are PUBLIC/REDACTED-PUBLIC)*
* relationships.\*.artifact\_ref: can be an Artifact ID (preferred) or a hash reference, as long as it’s unambiguous in the project’s index system.
* contents[].sha256 is optional but recommended for high‑rigor packs, especially RunPacks and AuditPacks.

**B.4 Pack templates (recommended folder structures)**

Each template below lists:

* **Purpose**
* **Minimum required files** (PUBLIC/REDACTED‑PUBLIC)
* **Recommended structure**
* **Recommended metadata files**

These are designed to be “audit-friendly defaults.”

**B.4.1 RulesPack — Template**

**Purpose:** Freeze definitions, evaluation criteria, scoring rules, scope, exclusions, and (where applicable) analysis plans **before** outcome-revealing work.

**Minimum required files:**

* README.md
* MANIFEST.json
* RULES.md *(or equivalent; must contain the frozen rules)*

**Recommended structure:**

<PROJECT\_ID>\_RulesPack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

├── RULES.md

├── SCOPE.md

├── DEFINITIONS.md

├── CRITERIA.md

├── EXCLUSIONS.md

├── CHANGELOG.md (recommended if this supersedes prior rules)

└── APPENDICES/ (optional)

└── examples.md

**RulesPack content guidelines:**

* State what is being “frozen” (definitions, criteria, scoring).
* State what would count as support/failure (where applicable).
* State scope boundaries clearly.
* If the project later changes rules, publish a new RulesPack version and justify the change.

**B.4.2 DataPack — Template**

**Purpose:** Record data/inputs “as received” (dataset snapshots, sources, acquisition notes, constraints).

**Minimum required files:**

* README.md
* MANIFEST.json
* DATA\_SOURCES.md *(source provenance and acquisition notes)*

**Recommended structure:**

<PROJECT\_ID>\_DataPack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

├── DATA\_SOURCES.md

├── LICENSES.md (recommended)

├── INTEGRITY.md (optional: per-file hashes summary)

├── data/

│ ├── raw/ (preferred)

│ ├── processed/ (optional)

│ └── notes/ (optional)

└── metadata/

├── data\_dictionary.csv (optional)

└── dataset\_manifest.csv (optional: files, sizes, checksums)

**DataPack notes:**

* If data cannot be redistributed, use HASH‑ONLY for the DataPack milestone and publish a public description in the Artifacts Index entry (what it is, why it cannot be public, how others can legally obtain it).
* If data is huge, you may include pointers plus per-file hashes rather than raw data bytes, but be explicit about what is and isn’t included.

**B.4.3 RunPack — Template**

**Purpose:** Capture a computation/analysis/simulation run in a way that makes reproduction or inspection feasible.

**Minimum required files:**

* README.md
* MANIFEST.json
* VERIFY.md *(strongly recommended; effectively required for serious RunPacks)*

**Recommended structure:**

<PROJECT\_ID>\_RunPack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

├── VERIFY.md

├── code/

│ ├── CODE\_REFERENCE.md (commit hash/tag + repo URL)

│ └── patches/ (optional)

├── env/

│ ├── requirements.txt (or environment.yml)

│ ├── container/ (optional)

│ │ └── Dockerfile

│ └── hardware\_notes.md (optional)

├── config/

│ ├── run\_config.json

│ └── seeds.txt (optional)

├── runs/

│ ├── run\_001/

│ │ ├── run\_metadata.json

│ │ ├── logs/

│ │ └── outputs/

│ └── run\_002/ (optional)

└── outputs\_summary/

├── figures/

├── tables/

└── summary.md

**Recommended run\_metadata.json (per run):**

{

"run\_id": "run\_001",

"date\_utc": "YYYY-MM-DDTHH:MM:SSZ",

"code\_reference": {

"repo": "<REPO\_URL>",

"commit": "<COMMIT\_HASH\_OR\_TAG>"

},

"inputs": [

{ "artifact\_ref": "<DATA\_PACK\_ID\_OR\_HASH>", "note": "input dataset reference" }

],

"config\_files": [

"config/run\_config.json"

],

"randomness": {

"seed": 12345,

"notes": "fixed seed for reproducibility"

},

"environment": {

"method": "pip",

"notes": "see env/requirements.txt"

},

"outputs": [

"runs/run\_001/outputs/"

]

}

**VERIFY.md recommended sections:**

* prerequisites (hardware/OS)
* environment setup steps
* how to run (exact commands)
* expected outputs and checks
* known limitations (“full reproduction requires GPU X,” etc.)

**B.4.4 ReleasePack — Template**

**Purpose:** Publish a coherent public release milestone (paper/preprint, major update, protocol release).

**Minimum required files:**

* README.md
* MANIFEST.json
* the release artifact(s) themselves (e.g., PDF)

**Recommended structure:**

<PROJECT\_ID>\_ReleasePack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

├── release/

│ ├── <TITLE>.pdf

│ ├── <TITLE>.docx (optional but recommended)

│ └── assets/ (optional: figures, diagrams)

├── RELEASE\_NOTES.md

├── CITATION.cff (optional but recommended)

└── links/

├── DOI\_OR\_ARCHIVE\_LINKS.md (optional)

└── mirrors.md (optional)

**ReleasePack notes:**

* ReleasePacks should link to supporting evidence packs where applicable (Rules/Data/Run).
* If a ReleasePack supersedes an earlier release, include a short change summary and keep old versions accessible.

**B.4.5 SubmitPack — Template**

**Purpose:** Commit the exact submission package **before submitting** to a journal/conference/venue (PUBLIC or HASH‑ONLY depending on policy/strategy).

**Minimum required files (if PUBLIC/REDACTED‑PUBLIC):**

* README.md
* MANIFEST.json
* submission/submission.json
* the submitted manuscript (and any required supplements)

**Recommended structure:**

<PROJECT\_ID>\_SubmitPack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

├── submission/

│ ├── submission.json

│ ├── manuscript.pdf

│ ├── manuscript\_source/ (optional)

│ ├── supplements/ (optional)

│ ├── cover\_letter.pdf (optional)

│ └── venue\_policy\_notes.md (optional)

└── provenance/

└── wsp\_provenance\_statement.md (optional)

**Recommended submission.json:**

{

"venue": "<VENUE\_NAME\_OR\_CODE>",

"submission\_date\_utc": "YYYY-MM-DDTHH:MM:SSZ",

"submission\_type": "journal\_article",

"related\_release": "<RELEASE\_PACK\_ID\_OR\_HASH>",

"supports\_packs": [

"<RULES\_PACK\_ID\_OR\_HASH>",

"<DATA\_PACK\_ID\_OR\_HASH>",

"<RUN\_PACK\_ID\_OR\_HASH>"

],

"visibility\_strategy": {

"pack\_visibility": "HASH-ONLY",

"reason": "venue policy uncertainty; preserving pre-submission commitment without disclosure risk"

}

}

**B.4.6 DecisionPack — Template**

**Purpose:** Record the outcome of a submission (policy-aware). May be PUBLIC, REDACTED‑PUBLIC, or HASH‑ONLY.

**Minimum required files (if PUBLIC/REDACTED‑PUBLIC):**

* README.md
* MANIFEST.json
* decision/decision.json
* decision/decision\_summary.md *(public summary written by the project)*

**Recommended structure:**

<PROJECT\_ID>\_DecisionPack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

└── decision/

├── decision.json

├── decision\_summary.md

├── letters/ (optional)

│ ├── decision\_letter\_redacted.pdf

│ └── reviewer\_comments\_redacted.pdf

└── disclosure\_notes.md (recommended)

**Recommended decision.json:**

{

"venue": "<VENUE\_NAME\_OR\_CODE>",

"decision\_date\_utc": "YYYY-MM-DDTHH:MM:SSZ",

"decision\_type": "desk\_reject",

"related\_submitpack": "<SUBMIT\_PACK\_ID\_OR\_HASH>",

"review\_materials": {

"included\_verbatim": false,

"reason": "venue confidentiality policy; public summary provided; verbatim stored hash-only"

}

}

**Policy-safe note:**  
If reviewer text is confidential, do **not** publish it verbatim. You may store it as HASH‑ONLY (timestamped) and publish a public summary.

**B.4.7 RevisionPack — Template**

**Purpose:** Publish revised manuscripts, response-to-reviewers, and change summaries as additive versions.

**Minimum required files:**

* README.md
* MANIFEST.json
* revised manuscript
* response document (authored by the project)

**Recommended structure:**

<PROJECT\_ID>\_RevisionPack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

├── revision/

│ ├── revised\_manuscript.pdf

│ ├── revised\_manuscript\_source/ (optional)

│ ├── response\_to\_reviewers.pdf (or .md)

│ ├── change\_summary.md

│ └── diffs/ (optional)

│ └── manuscript\_diff.pdf

**RevisionPack notes:**

* Always link to the specific SubmitPack and DecisionPack being responded to.
* If revisions change results, publish new RunPacks and reference them.

**B.4.8 AuditPack — Template**

**Purpose:** Publish an independent audit/replication attempt as a first-class artifact.

**Minimum required files:**

* README.md
* MANIFEST.json
* VERIFY.md (or equivalent audit procedure)
* explicit list of audited artifacts (hashes/versions)

**Recommended structure:**

<PROJECT\_ID>\_AuditPack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

├── VERIFY.md

├── audit/

│ ├── audit\_scope.md

│ ├── audited\_artifacts.json

│ ├── procedure.md

│ ├── environment.md

│ ├── outputs/

│ ├── discrepancies.md

│ └── conclusions.md

└── evidence/

├── logs/

└── reproduction\_outputs/

**Recommended audited\_artifacts.json:**

{

"audited\_artifacts": [

{ "artifact\_ref": "<RUN\_PACK\_ID\_OR\_HASH>", "note": "reproduced key figure outputs" },

{ "artifact\_ref": "<DATA\_PACK\_ID\_OR\_HASH>", "note": "verified dataset fingerprint match" }

],

"audit\_date\_utc": "YYYY-MM-DDTHH:MM:SSZ",

"auditor\_identity": "<AUDITOR\_NAME\_OR\_HANDLE>",

"scope\_notes": "What was checked and what was not checked."

}

AuditPacks are strongest when they are precise, bounded, and reproducible.

**B.4.9 TemplatesPack (optional)**

**Purpose:** Distribute reusable templates and schemas used by a project (index schema, manifest schema, verification scripts, etc.).

**Recommended structure:**

<PROJECT\_ID>\_TemplatesPack\_<VERSION>.zip

├── README.md

├── MANIFEST.json

├── schemas/

│ ├── manifest\_schema.json

│ └── index\_schema.json

├── templates/

│ ├── README\_template.md

│ ├── VERIFY\_template.md

│ └── submission\_template.json

└── tools/

├── index\_validator.py (optional)

└── hash\_check.sh (optional)

**B.5 Redaction and disclosure templates**

**B.5.1 REDACTIONS.md (required when REDACTED‑PUBLIC)**

If you publish a REDACTED‑PUBLIC bundle, include a REDACTIONS.md explaining:

* what was removed or obscured,
* why,
* how the redacted material relates to the original, and
* whether a hash-only version of the unredacted materials exists.

Recommended structure:

* Summary of redactions
* Redaction rationale (privacy/licensing/security)
* Redacted files list
* Notes on impact (what verification is still possible)
* Reference to related HASH‑ONLY milestone (if applicable)

**B.5.2 Hash-only disclosure (required in Artifacts Index entry)**

If a milestone is HASH‑ONLY, the Artifacts Index entry should include:

* a short description of what the bundle contains,
* why it cannot be published,
* how others could legally obtain equivalent inputs (if applicable), and
* whether a redacted public bundle exists.

**B.6 Minimal “starter bundles” (copy-ready suggestions)**

If a new project wants the fastest, cleanest start:

**Starter ReleasePack (Bronze):**

* README.md
* MANIFEST.json
* release/<YOUR\_DOC>.pdf
* RELEASE\_NOTES.md

**Starter RulesPack (Silver step):**

* README.md
* MANIFEST.json
* RULES.md
* SCOPE.md

**Starter RunPack (Gold step):**

* README.md
* MANIFEST.json
* VERIFY.md
* CODE\_REFERENCE.md
* requirements.txt
* run\_config.json
* logs/ + outputs/

These bundles alone can establish a clear proof trail and allow meaningful third-party engagement.

**Appendix C — Naming and Versioning Conventions**

**Stable identifiers, clear filenames, and credibility-preserving version discipline**

This appendix defines **recommended conventions** for naming WSP bundles (“packs”), assigning artifact identifiers, and managing versions over time. These conventions are not about aesthetics. They are about auditability:

* outsiders should be able to tell what a bundle is *at a glance*,
* versions should be unambiguous and never silently reused, and
* the “story of change” should be legible without requiring insider knowledge.

**Template notation:**  
This appendix uses angle‑bracket variables like <PROJECT\_ID> and <VERSION> to indicate values a project substitutes for its own implementation.

**C.1 Design goals**

WSP naming and versioning conventions aim to satisfy four goals:

1. **Uniqueness:** no two bundles should be confusingly similar or reuse the same identifier for different bytes.
2. **Legibility:** a filename should communicate pack type and version without opening it.
3. **Stability:** long-term references should survive website migrations, mirror moves, and team changes.
4. **Audit efficiency:** an auditor should be able to map claims to bundles quickly and compare versions reliably.

These goals are aligned with WSP’s core principles:

* **No silent rewrite**
* **Explain every hash**
* **Version discipline is credibility**

**C.2 Versioning rules (the credibility contract)**

A WSP “version” is a public commitment. Versioning is not marketing. It is integrity.

**C.2.1 The single most important rule**

If the bundle bytes change in any substantive way, the version **MUST** change.

* Repackaging the same content into a new zip can change bytes. If the bytes change, it’s a new bundle.
* Do not overwrite older versions. Publish a new bundle and link it via the Artifacts Index (supersedes relationship).

**C.2.2 “Never reuse a version label”**

Once a version label has been published as a milestone, it **MUST NOT** be reused for different bytes.

* If a mistake is found, publish a new version (typically a patch version) with a correction note.

**C.2.3 Recommended versioning schemes (pick one and be consistent)**

WSP does not mandate a single versioning scheme, but projects should choose a scheme that matches their work type.

**Option A: Semantic Versioning-like (vMAJOR.MINOR.PATCH) — recommended for protocols, software, and stable docs**

* **MAJOR:** breaking change in meaning, structure, or compatibility (e.g., index schema change that breaks tools).
* **MINOR:** additive changes (new sections, new fields that don’t break consumers).
* **PATCH:** corrections, clarifications, typo fixes, small non-breaking updates.

**Option B: Calendar Versioning (YYYY.MM.DD or YYYYMMDD) — recommended for frequent research milestones**

* Best when releases are driven by time, not interface compatibility.
* Still requires strict “no silent rewrite” and clear changelogs.

**Option C: Sequential (v1, v2, v3) — acceptable for early-stage projects**

* Use only if release frequency is low and changes are easy to interpret.
* Strongly recommended to add a CHANGELOG.md to reduce ambiguity.

**Rule:** Whatever scheme you choose, publish it on your Protocol page and apply it consistently.

**C.2.4 Patch discipline (corrections should strengthen credibility)**

When correcting an error:

* publish a new version (patch or sequential increment),
* include a CHANGELOG.md or correction note explaining what changed and why, and
* preserve the old version as part of the record.

WSP treats visible correction as a credibility-positive behavior.

**C.3 Filename conventions for packs (recommended default)**

A filename should be:

* machine-friendly (no spaces; stable separators),
* human-readable,
* and unambiguous about pack type and version.

**C.3.1 Recommended pack filename format**

**Recommended default:**

<PROJECT\_ID>\_<PACKTYPE>\_<VERSION>.zip

Examples:

* ARP\_ReleasePack\_v1.0.0.zip
* ARP\_RulesPack\_2025.12.21.zip
* ARP\_RunPack\_v0.3.1.zip
* ARP\_SubmitPack\_v1.zip

**C.3.2 Allowed characters (recommended)**

To reduce cross-platform issues, filenames **SHOULD** use only:

* A–Z, a–z, 0–9, underscore \_, hyphen -, and dot .

Avoid:

* spaces,
* parentheses,
* emojis,
* special characters that break URLs.

**C.3.3 Pack type naming (canonical)**

Use consistent pack type names. WSP canonical pack types include:

* RulesPack
* DataPack
* RunPack
* ReleasePack
* SubmitPack
* DecisionPack
* RevisionPack
* AuditPack
* TemplatesPack *(optional)*

Projects may add custom pack types, but if you do:

* define them on your Protocol page, and
* use the exact name consistently.

**C.3.4 “Latest” pointers (allowed, but not as a substitute for versioning)**

Projects MAY provide convenience pointers like:

* latest\_release.zip
* a “Latest” page on the website

…but the Proof Trail **MUST** remain versioned and additive. “Latest” pointers are UI conveniences only.

**C.4 Artifact ID conventions (recommended)**

Hashes are the cryptographic ground truth, but humans communicate better with stable IDs. WSP therefore recommends assigning an **Artifact ID** to each milestone entry in the Artifacts Index.

**C.4.1 Artifact ID format (recommended)**

A practical, readable format:

<PROJECT\_ID>-<PACK>-<SEQUENCE>

Examples:

* ARP-RELEASE-0001
* ARP-RULES-0003
* ARP-RUN-0021
* ARP-AUDIT-0007

Where:

* <PROJECT\_ID> is a short stable project code
* <PACK> is a short pack code (uppercase recommended)
* <SEQUENCE> is zero-padded for sorting consistency

**C.4.2 Pack code mapping (recommended)**

Suggested pack code mapping:

* RULES
* DATA
* RUN
* RELEASE
* SUBMIT
* DECISION
* REVISION
* AUDIT
* TEMPLATES

**C.4.3 Rules for Artifact IDs**

* An Artifact ID **SHOULD** never be reassigned to a different bundle hash.
* If an entry must be deprecated or corrected, publish a new entry with a new Artifact ID and link it via supersedes / deprecated\_by.

Artifact IDs exist to reduce confusion, not to replace hashes.

**C.5 Relationship declarations (how to make evolution legible)**

WSP bundles should not exist as isolated blobs. The Proof Trail is strongest when relationships are explicit.

**C.5.1 Recommended relationship fields**

In MANIFEST.json and in the Artifacts Index, projects SHOULD declare relationships such as:

* uses — what inputs or prior artifacts this depends on
* supports — what claims/releases this artifact supports
* supersedes — what artifact/version it replaces
* related — other relevant links (e.g., “same dataset family”)

**C.5.2 Supersession and deprecation rules**

When an artifact is replaced:

* the new version SHOULD include supersedes: <prior\_artifact\_ref>
* the prior artifact SHOULD remain referenced and accessible (PUBLIC if possible)

If an artifact is found to be materially wrong or compromised, do not delete it. Mark it clearly as superseded or deprecated, and explain why.

**C.6 Index file naming conventions (recommended)**

WSP strongly recommends a machine-readable index alongside the human-readable Artifacts page.

**C.6.1 Recommended canonical index filenames**

Publish these at stable URLs:

* wsp\_index.json
* wsp\_index.csv

**C.6.2 Recommended snapshot naming (optional but helpful)**

If you publish periodic snapshots, use deterministic names such as:

* wsp\_index\_2025-12-21.json
* wsp\_index\_2025-12-21.csv

If you timestamp index snapshots as milestones (recommended for high-rigor projects), snapshots make it easy to audit index evolution.

**C.7 Changelog and correction conventions**

Version numbers without explanations create confusion. A changelog turns version discipline into credibility.

**C.7.1 When to include CHANGELOG.md**

A bundle SHOULD include CHANGELOG.md when:

* it supersedes a previous bundle, or
* it contains corrections, or
* it changes claims materially.

Minimal changelog format:

* **What changed** (bullet list)
* **Why it changed** (one paragraph)
* **Impact on prior claims** (what is still valid, what is revised)
* **Links/refs** to superseded artifacts

**C.7.2 Correction notes (recommended)**

When an error is discovered, the new version SHOULD include a short “Correction Note” that:

* identifies the affected prior artifact(s) by ID/hash,
* states the error clearly,
* states the fix clearly,
* and states whether conclusions changed.

WSP treats transparent correction as a strength.

**C.8 Multi-venue submissions and parallel tracks**

Projects may submit to multiple venues over time. Naming should make this legible.

**C.8.1 SubmitPacks and venue codes (recommended)**

If you anticipate multiple submissions, include a short venue code in metadata (not necessarily in the filename):

* In submission.json: "venue": "<VENUE\_NAME\_OR\_CODE>"

You MAY also reflect the venue in the filename if it improves clarity, e.g.:

* ARP\_SubmitPack\_v1\_JOURNALX.zip

…but keep it consistent and avoid leaking confidential details if policy constraints exist.

**C.8.2 Resubmission and revision trail rules**

A resubmission SHOULD be recorded as:

* a new SubmitPack version, linked to the relevant RevisionPack and prior DecisionPack.

This creates a clean additive trail:  
Submit → Decision → Revision → Submit → …

**C.9 Minimal checklist (before publishing any milestone)**

Before publishing any bundle as a WSP milestone, projects SHOULD confirm:

* ✅ Bundle bytes are final and preserved
* ✅ Filename follows convention and includes version
* ✅ README.md and MANIFEST.json are present (PUBLIC/REDACTED‑PUBLIC)
* ✅ Version label is new (not reused)
* ✅ If superseding: CHANGELOG.md included and supersedes relationship declared
* ✅ Hash computed correctly (e.g., SHA‑256)
* ✅ Timestamp record created and verifiable under the provenance identity
* ✅ Artifacts Index entry is complete (“explain every hash”)
* ✅ Download links work (PUBLIC/REDACTED‑PUBLIC)
* ✅ Prior versions remain referenced and (when feasible) accessible

This checklist is short on purpose. WSP is meant to be practical and repeatable.

**Appendix D — Artifacts Index Schema**

**Machine-readable Proof Trail mapping (wsp\_index.json and wsp\_index.csv)**

The **Artifacts Index** is the mapping layer that turns a Proof Trail into something verifiable and usable. A provenance wallet (or any timestamp log) can prove that *some* hashes were recorded by time **T**, but without a readable map, outsiders cannot tell:

* what each hash represents,
* where to download the exact bundle bytes (if public),
* how milestones relate across versions, or
* what changed over time.

This appendix defines a recommended, machine-readable schema for publishing the Artifacts Index as:

* wsp\_index.json (canonical, expressive, tool-friendly)
* wsp\_index.csv (easy for humans/spreadsheets and lightweight tooling)

Projects MAY implement only one, but WSP strongly recommends publishing **both**, generated from the same canonical source of truth.

**Template notation:**  
This appendix uses angle‑bracket variables like <PROJECT\_ID> and <PROVENANCE\_ID> as template variables that each project substitutes for its own implementation.

**D.1 Index design goals (why this schema exists)**

This schema is designed to satisfy five goals:

1. **Explain every hash**  
   Every timestamped milestone must be mapped to a clear description and retrieval method.
2. **Support fast verification**  
   A verifier should be able to locate a PUBLIC bundle, hash it, match it to the index, and then verify the timestamp reference.
3. **Make relationships legible**  
   Milestones should form a coherent graph: rules → data → runs → releases → submissions → decisions → revisions → audits.
4. **Be toolable**  
   The index should be easy to validate automatically (missing fields, broken links, supersession rules, etc.).
5. **Survive platform change**  
   The index should remain stable even as mirrors move or hosting platforms change.

**D.2 Canonical files and publishing requirements**

**D.2.1 Required public endpoints (recommended best practice)**

Projects should publish these at stable URLs on the canonical home domain:

* .../wsp\_index.json
* .../wsp\_index.csv

The human-readable Artifacts page may live anywhere, but these two files SHOULD be stable and easy to locate.

**D.2.2 Index update discipline (normative)**

A project following WSP **MUST** ensure:

* every timestamped milestone hash is represented by an index entry, and
* entries are **additive** over time.

Edits should be treated with caution. If a correction is needed, the project SHOULD publish:

* a new entry that supersedes the old entry, and
* clear notes explaining the correction,  
  rather than silently rewriting historical records.

**D.2.3 Index anchoring (strongly recommended)**

Projects SHOULD periodically:

* hash the current wsp\_index.json (and optionally wsp\_index.csv), and
* timestamp that hash as a milestone (often as an IndexPack or ReleasePack).

This makes index evolution auditable and discourages quiet retroactive rewrites.

**D.3 wsp\_index.json top-level structure (recommended)**

A recommended wsp\_index.json structure:

{

"wsp\_spec": { "name": "Work Speaks Protocol", "version": "1.0" },

"index": {

"format": "wsp\_index",

"schema\_version": "1.0",

"generated\_utc": "YYYY-MM-DDTHH:MM:SSZ",

"canonical\_home": "<CANONICAL\_HOME\_URL>",

"artifacts\_index\_url": "<ARTIFACTS\_PAGE\_URL>",

"provenance\_identity": "<PROVENANCE\_WALLET\_OR\_ID>",

"hash\_algorithm\_default": "SHA-256"

},

"project": {

"name": "<PROJECT\_NAME>",

"project\_id": "<PROJECT\_ID>",

"contact": "<OPTIONAL\_CONTACT>",

"official\_links": {

"protocol\_page": "<URL>",

"verify\_guide": "<URL>",

"wallets\_page": "<URL>"

}

},

"entries": [

{ /\* Artifact entry objects \*/ }

]

}

**Notes**

* schema\_version refers to the index schema (this appendix), not the WSP spec version.
* generated\_utc is when the index file itself was generated.
* provenance\_identity is the official identity that issued the milestone timestamps.

**D.4 Artifact entry object (required fields + recommended fields)**

Each element in entries[] describes a single milestone bundle (or hash-only milestone).

**D.4.1 Required fields (minimum viable)**

Each entry **MUST** include the following fields:

* artifact\_id *(recommended but treated as required for practical use)*
* pack\_type
* version
* title
* description
* created\_utc *(best effort; when the bundle was created/released)*
* bundle object:
  + filename
  + hash\_algorithm
  + hash *(e.g., SHA-256 hex)*
* visibility *(PUBLIC / REDACTED-PUBLIC / HASH-ONLY)*
* timestamp object:
  + method *(short label)*
  + reference *(tx id / proof id / anchor id)*
  + attribution *(how to confirm it’s tied to provenance identity; may be implicit)*
* provenance\_identity *(explicit per entry; allows audits even if project header changes)*
* retrieval object:
  + if PUBLIC/REDACTED-PUBLIC: at least one URL in mirrors[]
  + if HASH-ONLY: hash\_only\_reason must be present and non-empty

**D.4.2 Recommended fields (highly useful for audits and tools)**

Entries SHOULD also include:

* artifact\_type *(optional alias for pack\_type; not necessary if pack\_type is canonical)*
* relationships (uses, supports, supersedes, related)
* tags (e.g., ["simulation","preprint","submission"])
* size\_bytes (bundle file size; helps detect download corruption)
* content\_summary (brief list of key files)
* notes (free-text clarifications)
* disclosures (redaction notes, licensing notes, privacy constraints)
* mirrors with typed roles (primary/mirror/doi/etc.)
* integrity object (optional per-file hashes, manifest checksum, etc.)

**D.5 Canonical entry schema (recommended)**

Below is a recommended entry schema that satisfies WSP requirements and supports tooling:

{

"artifact\_id": "<PROJECT\_ID>-<PACK>-<SEQUENCE>",

"pack\_type": "RunPack",

"version": "v0.3.1",

"title": "Experiment A — Run outputs (seeded)",

"description": "RunPack capturing code reference, env, configs, logs, and outputs for Experiment A.",

"created\_utc": "YYYY-MM-DDTHH:MM:SSZ",

"provenance\_identity": "<PROVENANCE\_WALLET\_OR\_ID>",

"visibility": "PUBLIC",

"bundle": {

"filename": "<PROJECT\_ID>\_RunPack\_v0.3.1.zip",

"hash\_algorithm": "SHA-256",

"hash": "<SHA256\_HEX>",

"size\_bytes": 12345678

},

"timestamp": {

"method": "<TIMESTAMP\_METHOD\_LABEL>",

"reference": "<TIMESTAMP\_REFERENCE\_ID>",

"verification\_hint": "Publicly verifiable record containing the bundle hash, attributable to the provenance identity."

},

"retrieval": {

"mirrors": [

{

"url": "<PRIMARY\_DOWNLOAD\_URL>",

"role": "primary",

"notes": ""

},

{

"url": "<MIRROR\_DOWNLOAD\_URL>",

"role": "mirror",

"notes": ""

}

],

"hash\_only\_reason": ""

},

"relationships": {

"uses": [

{ "artifact\_ref": "<PROJECT\_ID>-DATA-0002", "note": "Dataset snapshot as received" },

{ "artifact\_ref": "<PROJECT\_ID>-RULES-0001", "note": "Frozen evaluation criteria" }

],

"supports": [

{ "artifact\_ref": "<PROJECT\_ID>-RELEASE-0003", "note": "Supports the v0.3 release claims" }

],

"supersedes": [

{ "artifact\_ref": "<PROJECT\_ID>-RUN-0019", "note": "Updated config and fixed logging bug" }

],

"related": []

},

"content\_summary": [

"README.md",

"MANIFEST.json",

"VERIFY.md",

"env/requirements.txt",

"config/run\_config.json",

"runs/run\_001/logs/",

"runs/run\_001/outputs/"

],

"disclosures": {

"redactions": false,

"redactions\_note": "",

"licensing\_notes": "",

"privacy\_notes": ""

},

"notes": "Full reproduction requires GPU class X; partial verification possible on CPU (see VERIFY.md).",

"tags": ["run", "reproducibility", "experiment-a"]

}

**D.6 HASH‑ONLY entry requirements (policy-aware transparency)**

HASH‑ONLY is allowed in WSP, but it must be used transparently.

A HASH‑ONLY entry MUST:

* include the bundle hash and timestamp reference, and
* include a clear hash\_only\_reason explaining why the bundle is not public.

Example:

{

"artifact\_id": "<PROJECT\_ID>-DATA-0004",

"pack\_type": "DataPack",

"version": "v1.0",

"title": "Dataset snapshot (restricted)",

"description": "Restricted dataset snapshot as received; redistribution not permitted by license.",

"created\_utc": "YYYY-MM-DDTHH:MM:SSZ",

"provenance\_identity": "<PROVENANCE\_WALLET\_OR\_ID>",

"visibility": "HASH-ONLY",

"bundle": {

"filename": "<PROJECT\_ID>\_DataPack\_v1.0.zip",

"hash\_algorithm": "SHA-256",

"hash": "<SHA256\_HEX>",

"size\_bytes": 0

},

"timestamp": {

"method": "<TIMESTAMP\_METHOD\_LABEL>",

"reference": "<TIMESTAMP\_REFERENCE\_ID>",

"verification\_hint": "Public timestamp record contains the bundle hash."

},

"retrieval": {

"mirrors": [],

"hash\_only\_reason": "Redistribution prohibited by dataset license. See DATA\_SOURCES.md summary on canonical site for acquisition details and instructions to obtain legally."

},

"relationships": {

"uses": [],

"supports": [{ "artifact\_ref": "<PROJECT\_ID>-RUN-0021", "note": "Inputs used for Experiment B run" }],

"supersedes": [],

"related": []

},

"disclosures": {

"redactions": false,

"redactions\_note": "",

"licensing\_notes": "Restricted license; no redistribution.",

"privacy\_notes": ""

},

"tags": ["data", "restricted"]

}

**Rule of thumb:**  
HASH‑ONLY preserves integrity, but PUBLIC bundles maximize verifiability. Projects should prefer PUBLIC where feasible.

**D.7 wsp\_index.csv schema (columns and mapping)**

CSV is a practical companion format. It should be consistent and easy to parse.

**D.7.1 Recommended CSV columns**

A recommended column set (in this order):

1. artifact\_id
2. pack\_type
3. version
4. title
5. description
6. created\_utc
7. visibility
8. bundle\_filename
9. hash\_algorithm
10. bundle\_hash
11. size\_bytes
12. provenance\_identity
13. timestamp\_method
14. timestamp\_reference
15. primary\_url
16. mirror\_urls *(semicolon-separated)*
17. hash\_only\_reason
18. uses *(semicolon-separated artifact\_refs)*
19. supports *(semicolon-separated artifact\_refs)*
20. supersedes *(semicolon-separated artifact\_refs)*
21. tags *(semicolon-separated)*
22. notes

**D.7.2 Example CSV row**

ARP-RUN-0021,RunPack,v0.3.1,"Experiment A — Run outputs (seeded)","RunPack capturing env/config/logs/outputs for Experiment A.",2025-12-21T00:00:00Z,PUBLIC,ARP\_RunPack\_v0.3.1.zip,SHA-256,<SHA256\_HEX>,12345678,0xPROVENANCE,"onchain\_tx","<TXID>","https://example.org/bundles/ARP\_RunPack\_v0.3.1.zip","https://mirror1.example.org/ARP\_RunPack\_v0.3.1.zip;https://mirror2.example.org/ARP\_RunPack\_v0.3.1.zip","",ARP-DATA-0002;ARP-RULES-0001,ARP-RELEASE-0003,ARP-RUN-0019,run;reproducibility;experiment-a,"Full reproduction requires GPU class X."

*(CSV will not capture nested structure as cleanly as JSON; that’s expected. JSON remains the canonical format for complex relationships and disclosures.)*

**D.8 Validation rules (what tooling SHOULD check)**

To make WSP reliable at scale, index tooling SHOULD validate:

1. **Completeness**

* required fields present for every entry
* visibility is valid
* PUBLIC entries have at least one mirror URL
* HASH‑ONLY entries have non-empty hash\_only\_reason

1. **Uniqueness**

* artifact\_id unique
* (pack\_type, version) uniqueness is strongly recommended
* bundle\_hash should not appear under multiple different artifact IDs unless explicitly documented (rare)

1. **Relationship integrity**

* referenced artifact IDs exist (where the project uses IDs)
* supersedes forms an acyclic chain (no loops)

1. **Retrieval integrity (PUBLIC/REDACTED‑PUBLIC)**

* downloadable bundle exists
* downloaded hash matches bundle\_hash
* (optional) size\_bytes matches

1. **Timestamp verifiability**

* timestamp reference resolves to a public record
* the record contains the bundle hash
* the record is attributable to the provenance identity (as defined by the project’s verification method)

**D.9 Minimal guidance for projects adopting this schema**

If you want the simplest, cleanest adoption:

* Make wsp\_index.json canonical.
* Auto-generate wsp\_index.csv and your human-readable Artifacts page from it.
* Treat the index as an append-only ledger:
  + new milestones add new entries,
  + corrections supersede old entries,
  + nothing is silently rewritten.

That is how the Proof Trail stays legible as the project grows.

**Appendix E — Verification Guide**

**Beginner verification, auditor verification, and how to report discrepancies**

This appendix provides practical, step-by-step guidance for verifying a Work Speaks Protocol (WSP) Proof Trail. The goal is to make verification **mechanical**—so you do not need to trust the author’s identity, credentials, or narrative in order to verify the integrity of a milestone.

WSP verification answers two core questions:

1. **Integrity:** “Do the bytes I downloaded match the committed hash?”
2. **Timeline + attribution:** “Was that hash publicly recorded by time *T* under the project’s declared provenance identity?”

This appendix is intentionally **timestamp-method agnostic**. Projects may timestamp using an on-chain record, a public timestamping service, or another durable public anchor. The verification logic remains the same: the public record must contain the hash and be attributable to the provenance identity.

**E.1 What you need (minimal)**

To verify a WSP milestone you need:

* the project’s **Artifacts Index** (human-readable page or machine-readable wsp\_index.json/csv), and
* access to at least one **PUBLIC** bundle download link, and
* a way to compute a **SHA‑256** hash on your machine, and
* access to the project’s **timestamp reference** record (e.g., explorer page / proof page) so you can confirm the hash exists in a public timestamp record.

You do **not** need to run any code, open any notebooks, or understand the underlying research to verify integrity.

**E.2 Beginner verification (PUBLIC milestone)**

**Verify one milestone end-to-end in minutes**

This is the simplest, most important verification flow. It applies to any **PUBLIC** ReleasePack, RulesPack, RunPack, etc.

**Step 1 — Locate the milestone entry in the Artifacts Index**

Go to the project’s Artifacts Index and locate an entry marked:

* **Visibility:** PUBLIC

Record (copy/paste) these fields from the entry:

* **Bundle hash** (SHA‑256)
* **Hash algorithm** (should be SHA‑256 unless stated otherwise)
* **Bundle filename** (optional but helpful)
* **Download URL** (primary and/or mirror)
* **Timestamp method label**
* **Timestamp reference** (e.g., transaction ID / proof ID / anchor ID)
* **Provenance identity** (e.g., provenance wallet address)

**Step 2 — Download the bundle file**

Download the zip bundle from the index link.

Best practices:

* Download directly from the URLs listed in the Artifacts Index (not re-uploaded copies from third parties).
* If the index lists size\_bytes, confirm the downloaded file size matches.

**Step 3 — Compute the SHA‑256 hash of the downloaded zip**

Compute the hash of the **zip file itself** (not its extracted folder).

**macOS / Linux (Terminal):**

* shasum -a 256 <bundle.zip>  
  or
* sha256sum <bundle.zip>

**Windows (PowerShell):**

* Get-FileHash <bundle.zip> -Algorithm SHA256

This produces a hex hash string.

**Step 4 — Compare your computed hash to the Artifacts Index hash**

* If your computed hash **matches** the index hash: you have verified the bundle bytes.
* If it **does not match**: treat the bundle as unverified. Do not assume malice; it may be mirror drift, partial download, or an index error. Proceed to E.6 (Reporting discrepancies).

**Step 5 — Verify the timestamp record**

Using the Artifacts Index “timestamp reference,” locate the public timestamp record and confirm:

* the record contains the exact bundle hash, and
* the record is anchored to the project’s provenance identity (see Step 6).

What “contains the hash” means depends on the timestamp method:

* for on-chain records: the hash may appear in transaction input data, message fields, logs, or other verifiable fields,
* for timestamping services: the hash should appear in the proof payload or verifiable receipt.

If you cannot find the hash in the record, treat the timestamp as unverified and report it (E.6).

**Step 6 — Confirm attribution to the provenance identity**

WSP requires that the timestamp is attributable to the project’s declared provenance identity.

How you verify attribution depends on the timestamp method:

**Common case: on-chain transaction**

* Confirm the “from” address (or signing identity) matches the project’s declared provenance wallet.
* Confirm the transaction exists and is final (included in a block / finalized).
* Confirm the block time (or network time) is consistent with the milestone date.

**Timestamp service case**

* Confirm the receipt/proof is verifiable via the service’s published verification method.
* Confirm the proof is linked to the project’s identity in whatever way the project defines (for example, a signed statement, a verified account, or a public anchor record tied to the provenance identity).

If the timestamp record contains the hash but attribution is ambiguous, the milestone is not fully WSP-verifiable. Report it as an issue.

**Step 7 — Record your verification result (recommended)**

A basic verification note should include:

* project name + canonical home
* artifact ID (if provided)
* bundle hash (SHA‑256)
* timestamp reference
* date/time you verified (UTC best-effort)
* outcome: “hash verified / timestamp verified / attribution verified” (yes/no)

You can share this publicly (e.g., as a comment, forum post) or as a formal AuditPack (see E.5).

**E.3 How to verify timestamp records (method-agnostic rules)**

WSP does not require one timestamp mechanism. It requires that timestamp verification is **objective**.

A timestamp record is WSP-valid if a verifier can confirm:

1. **Inclusion:** the record contains the exact bundle hash (or a deterministic commitment that unambiguously encodes it).
2. **Public verifiability:** the record is accessible and can be verified without private access.
3. **Attribution:** the record is attributable to the project’s provenance identity.
4. **Time:** the record includes or implies a reliable timestamp (e.g., block time, service timestamp, anchor time).

**E.3.1 Common timestamp patterns and what to check**

**Pattern A — On-chain transaction record**

**Verify:**

* transaction exists and is included/finalized,
* the transaction data contains the bundle hash,
* the signing/from identity matches the provenance wallet,
* record time is consistent.

**Common pitfalls:**

* the hash is encoded differently than expected (uppercase/lowercase, prefixed, truncated),
* the record contains a different hash (e.g., per-file hash instead of bundle hash),
* a non-provenance wallet was used.

**Pattern B — Public timestamping service receipt**

**Verify:**

* the receipt is authentic and verifiable via the service’s procedure,
* the receipt contains the bundle hash,
* the receipt is tied to the project’s provenance identity (as defined by the project),
* timestamp is stable.

**Common pitfalls:**

* screenshots or PDFs of receipts without verifiable proofs,
* receipts that prove “something was timestamped” without showing what,
* identity ambiguity.

**Pattern C — Signed statement anchored publicly**

Some projects may publish a signed statement that includes the hash and is anchored through a public channel.

**Verify:**

* the signature verifies against the project’s published public key/provenance identity,
* the statement is anchored publicly at time T (e.g., archived, timestamped, or otherwise durable),
* the statement contains the exact bundle hash.

**Common pitfalls:**

* unsigned statements,
* signatures that don’t match published identity keys,
* anchoring that isn’t durable or independently verifiable.

**E.4 Auditor verification (deeper checks)**

Auditor-level verification goes beyond “the hash matches” and evaluates whether the Proof Trail is coherent, complete, and usable for replication or critique.

**E.4.1 Proof Trail completeness check (“Explain Every Hash”)**

A WSP project’s Proof Trail is only as credible as its mapping discipline.

An auditor SHOULD check:

* Every timestamped milestone hash claimed as official is explained in the Artifacts Index.
* Every Artifacts Index entry has:
  + a bundle hash,
  + a timestamp reference,
  + a provenance identity,
  + and a retrieval method (PUBLIC URL(s) or HASH‑ONLY reason).
* PUBLIC links work; mirrors are consistent.
* No “mystery hashes” exist in the official provenance identity record without index explanations.

If the project publishes wsp\_index.json, auditors should prefer validating against it (tool-friendly).

**E.4.2 Bundle internal integrity checks (PUBLIC / REDACTED‑PUBLIC bundles)**

After verifying the bundle hash, an auditor SHOULD inspect inside the zip for expected structure:

* README.md exists and points to canonical home + Artifacts Index.
* MANIFEST.json exists and is internally coherent:
  + correct pack type and version,
  + correct canonical links,
  + correct relationship references (if included).
* If redacted:
  + REDACTIONS.md exists and explains what is removed and why.

Optional but strong checks:

* If the manifest includes per-file SHA‑256 values, recompute those and confirm they match.
* Confirm that no unexpected executable artifacts are included unless the pack requires it (e.g., RunPack tools).

**E.4.3 Relationship graph sanity check**

A mature Proof Trail is a graph, not a pile.

Auditors SHOULD check:

* Relationship references resolve (artifact IDs/hashes exist in the index).
* Supersession chains are coherent (new versions supersede older ones; no loops).
* Major claims in ReleasePacks reference the supporting chain (Rules/Data/Run where applicable).

**E.4.4 RunPack reproducibility / inspection (when applicable)**

If auditing a RunPack, an auditor SHOULD:

* confirm code references are unambiguous (repo + commit hash/tag),
* confirm environment/dependency files exist (requirements.txt, environment.yml, container spec),
* confirm run configs are present and referenced,
* confirm logs and outputs exist and match described results,
* attempt reproduction if feasible, or attempt partial reproduction (e.g., regenerate a figure).

If full reproduction is not feasible:

* document exactly what was attempted,
* document constraints (hardware, licensing, compute),
* and publish an AuditPack reporting the result.

**E.4.5 Submission integrity verification (when present)**

If a project claims submission integrity:

* verify that SubmitPack is timestamped **before** submission (best effort based on timestamps),
* verify that DecisionPack is recorded and policy-aware,
* verify that revision chains are additive (RevisionPack versions, no silent overwrite).

Auditors should be careful with confidentiality: verify the integrity trail without demanding policy-violating disclosures.

**E.5 Publishing verification as an AuditPack (recommended output format)**

WSP’s strongest ecosystem behavior is third parties publishing audits as artifacts.

If you perform any non-trivial verification, you SHOULD consider publishing an **AuditPack** so others can review what you did.

**E.5.1 Minimal AuditPack contents (recommended)**

A minimal, high-value AuditPack includes:

* README.md (what you audited and why)
* MANIFEST.json
* audit\_scope.md (what you did and did not check)
* audited\_artifacts.json (exact artifact IDs/hashes audited)
* procedure.md (steps taken)
* environment.md (tools/OS, and dependencies if you ran code)
* outputs/ (any outputs produced)
* discrepancies.md (if any)
* conclusions.md (clear statement: reproduced / partially reproduced / did not reproduce / integrity failure)

**E.5.2 Auditor standards (culture)**

An AuditPack should be:

* specific,
* bounded,
* evidence-forward,
* and precise about uncertainty.

Audits are not “final judgments.” They are contributions to a shared evidence process.

**E.6 Reporting discrepancies (responsible disclosure and corrective workflow)**

Not every mismatch is fraud. The most common issues are mundane:

* partial downloads,
* mirror drift,
* index typos,
* wrong file hashed,
* outdated links,
* inconsistent timestamp encoding.

When you find a discrepancy, report it clearly and constructively.

**E.6.1 Discrepancy severity categories (practical)**

**Category 1 — Retrieval issues**

* broken link, mirror missing, file renamed without index update

**Category 2 — Mapping issues**

* index entry missing required fields,
* “mystery hashes” in provenance identity record,
* ambiguous pack type/version labeling

**Category 3 — Integrity issues**

* downloaded bundle hash does not match index hash (from a listed official URL),
* per-file hashes don’t match manifest (if provided)

**Category 4 — Timestamp / attribution issues**

* timestamp record does not contain the hash,
* timestamp record exists but is not attributable to the declared provenance identity,
* timestamp reference is invalid or unverifiable

Category 3–4 issues are serious because they undermine the Proof Trail.

**E.6.2 What to include in a good discrepancy report**

A good report includes:

* project canonical home URL
* artifact ID (or pack type + version)
* the index-listed hash and the hash you computed
* the download URL used
* your OS and hash command used
* the timestamp reference and what you observed
* screenshots are optional; copy-pasteable values are better

**E.6.3 Preferred remediation behavior (what a WSP project should do)**

A WSP project should respond to discrepancies by:

* acknowledging the report,
* preserving additive history (no deleting or rewriting),
* publishing a corrected version or corrected index entry (superseding if needed),
* and documenting what changed and why.

Transparent correction is credibility-positive in WSP.

**E.7 Security notes (verify safely)**

Verification usually requires only hashing files—no code execution. But some bundles (especially RunPacks and tool bundles) may contain code.

Recommended safety posture:

* Do not run downloaded code on a machine you care about.
* Use a sandbox, container, or isolated environment for reproduction attempts.
* Treat external links inside bundles as untrusted until verified.
* Prefer hashing and inspection first; execution only when needed for audit scope.

WSP improves integrity verification, but it does not eliminate general software safety risk.

**Appendix F — Checklists and Quick Reference**

**Practical checklists for projects, auditors, and communities**

This appendix provides copy-ready checklists that operationalize Work Speaks Protocol (WSP). The purpose is simple: **reduce mistakes and reduce audit friction**.

These checklists are intentionally short, mechanical, and aligned with the core principles:

* **No silent rewrite**
* **Explain every hash**
* **Version discipline is credibility**
* **Receipts first**

**F.1 Bronze / Silver / Gold compliance checklists**

**F.1.1 Bronze checklist — Timeline integrity (Proof Trail foundation)**

A project MAY claim **Bronze** only if all items below are true:

**Identity + anchors**

* ☐ Canonical project home is public and easy to find
* ☐ Artifacts Index URL is public and linked prominently
* ☐ Provenance identity (e.g., provenance wallet) is publicly declared
* ☐ (Recommended) “Official Identifiers” page exists (wallet roles + official links)

**Proof Trail mechanics**

* ☐ At least one **PUBLIC** bundle exists (typically a ReleasePack)
* ☐ Each PUBLIC bundle includes README.md and MANIFEST.json
* ☐ Bundle hash (SHA‑256 by default) is computed from the **zip file bytes**
* ☐ Hash is **timestamped** via a public verifiable record attributable to the provenance identity
* ☐ Artifacts Index entry exists for every timestamped milestone hash
* ☐ Each entry includes: pack type, version, hash, timestamp reference/method, and download link(s)

**History discipline**

* ☐ No older public versions have been overwritten or deleted from the record
* ☐ Corrections are published as new versions (not silent edits)

**F.1.2 Silver checklist — Rules + inputs integrity (pre-commit discipline)**

A project MAY claim **Silver** only if all Bronze items are true and:

**Pre-commit rules**

* ☐ For any major results claim, a **RulesPack** exists
* ☐ RulesPack is timestamped **before** outcome-revealing runs/releases
* ☐ RulesPack clearly defines: scope, criteria, assumptions, and exclusions

**Inputs integrity**

* ☐ For results dependent on external inputs, a **DataPack (or input equivalent)** exists
* ☐ DataPack is timestamped at or near acquisition (best effort)
* ☐ If DataPack is HASH‑ONLY, the Artifacts Index explains the constraint clearly

**Index coherence**

* ☐ The index links the chain: Rules → Data → Run (if applicable) → Release
* ☐ Supporting artifacts are referenced by Artifact ID and/or hash unambiguously

**F.1.3 Gold checklist — End-to-end evidence chain (maximum auditability)**

A project MAY claim **Gold** only if Bronze + Silver items are true and:

**Run-level evidence (when computation/analysis is involved)**

* ☐ Each major computational claim is backed by a **RunPack**
* ☐ RunPack includes:
  + ☐ unambiguous code reference (repo + commit/tag)
  + ☐ environment/dependency info (requirements.txt, environment.yml, container spec, etc.)
  + ☐ configuration + seeds (when relevant)
  + ☐ logs + outputs (best effort)
  + ☐ VERIFY.md reproduction guidance (best effort)

**Submission integrity (only if the project uses submissions and claims this layer)**

* ☐ SubmitPack is committed **before** submission (timestamp before submit action)
* ☐ DecisionPack records the outcome policy-aware
* ☐ Reviewer/editor text is published only if permitted; otherwise hash-only + summary
* ☐ RevisionPack(s) preserve additive revision history

**Audit enablement**

* ☐ Project keeps PUBLIC bundles accessible (links maintained and mirrored where feasible)
* ☐ Project invites and respects independent **AuditPacks**
* ☐ Corrections and disagreements are handled through artifacts (not social suppression)

**F.2 “Publish a milestone” checklist (applies to any Pack)**

Use this before you timestamp any new milestone.

**Bundle preparation**

* ☐ Bundle bytes are final (you will not re-zip or re-export after hashing)
* ☐ Filename includes pack type + version (Appendix C)
* ☐ README.md included (PUBLIC/REDACTED‑PUBLIC)
* ☐ MANIFEST.json included (PUBLIC/REDACTED‑PUBLIC)
* ☐ If superseding a prior version:
  + ☐ CHANGELOG.md included (recommended)
  + ☐ supersedes relationship is declared (manifest + index)

**Hashing**

* ☐ SHA‑256 computed from the **zip file** (not extracted contents)
* ☐ Hash recorded exactly (no truncation, no formatting ambiguity)

**Timestamping**

* ☐ Timestamp record created under the provenance identity
* ☐ Timestamp reference recorded (tx/proof/anchor ID)
* ☐ Public verification path exists (someone else can check it)

**Publishing**

* ☐ Bundle uploaded to at least one PUBLIC host (if PUBLIC)
* ☐ (Recommended) Bundle mirrored to a second host
* ☐ Links tested (download works; file size matches)

**Artifacts Index**

* ☐ Index entry created immediately (don’t let hashes accumulate unexplained)
* ☐ Entry includes: pack type, version, hash, timestamp reference/method, visibility class
* ☐ PUBLIC entries include download mirror(s)
* ☐ HASH‑ONLY entries include a clear hash\_only\_reason
* ☐ Relationships (uses/supports/supersedes) are consistent

**F.3 Pack-specific mini checklists (quick expectations)**

**F.3.1 RulesPack quick checklist**

* ☐ Criteria and definitions are explicit
* ☐ Scope and exclusions are explicit
* ☐ Timestamped **before** outcome work
* ☐ Any later change is a new version with justification

**F.3.2 DataPack quick checklist**

* ☐ Source provenance described (where it came from, when acquired, constraints)
* ☐ If public: dataset snapshot or strong pointers + integrity metadata included
* ☐ If hash-only: index clearly explains why, and how others can legally obtain inputs if possible

**F.3.3 RunPack quick checklist**

* ☐ Code reference is unambiguous (repo + commit/tag)
* ☐ Environment is documented
* ☐ Configs and seeds recorded (when relevant)
* ☐ Outputs/logs included (best effort)
* ☐ VERIFY.md provides reproducible steps or clearly states limitations

**F.3.4 ReleasePack quick checklist**

* ☐ Release artifact is present (PDF, doc, etc.)
* ☐ Release notes summarize what’s new and what changed
* ☐ References supporting artifacts (Rules/Data/Run) where applicable
* ☐ Older releases are not removed—only superseded

**F.3.5 SubmitPack quick checklist**

* ☐ Timestamped **before** submission
* ☐ Venue metadata recorded
* ☐ Visibility strategy is explicit (PUBLIC vs HASH‑ONLY)
* ☐ No silent changes after timestamp; revisions become new packs

**F.3.6 DecisionPack quick checklist**

* ☐ Outcome type and date recorded
* ☐ Public summary included (authored by project)
* ☐ Confidential reviewer text handled policy-aware (don’t publish if not allowed)
* ☐ Links back to the corresponding SubmitPack

**F.3.7 AuditPack quick checklist**

* ☐ Scope is explicit (“what I checked” / “what I didn’t check”)
* ☐ Audited artifact IDs/hashes listed precisely
* ☐ Steps and environment documented
* ☐ Outputs and discrepancies included
* ☐ Conclusion is clear (reproduced / partial / failed / integrity issue)

**F.4 Submission integrity checklist (policy-aware)**

Use this when you want the “commit-before-submit” credibility layer.

**Before submission**

* ☐ Submission package assembled as a single bundle (SubmitPack)
* ☐ SubmitPack hashed and timestamped **before** sending to venue
* ☐ Index entry created with venue metadata (named or coded)

**After decision**

* ☐ DecisionPack created with decision type and date
* ☐ Public decision summary written (what happened, what’s next)
* ☐ Venue confidentiality rules reviewed
* ☐ Verbatim reviewer/editor text published only if policy/permission allows
* ☐ If not allowed: store verbatim materials hash-only (optional) + publish summary

**Revision cycle**

* ☐ Revisions published as RevisionPacks (new versions)
* ☐ Response-to-reviewers included (authored by project)
* ☐ If results changed: publish new RunPack(s) and reference them
* ☐ Resubmissions become new SubmitPack versions

**Conduct**

* ☐ No doxxing, no harassment, no personal targeting
* ☐ Critique stays artifact-focused

**F.5 Artifacts Index maintenance checklist (continuous operations)**

A Proof Trail stays credible only if the map stays accurate.

**Index hygiene**

* ☐ Every new timestamped hash is indexed promptly
* ☐ No “mystery hashes” exist in the official provenance record without an index explanation
* ☐ Links are tested periodically (at least for major PUBLIC milestones)
* ☐ Mirrors are updated if hosts change
* ☐ Supersession chains are explicit and coherent

**Machine-readable index (recommended)**

* ☐ wsp\_index.json published at a stable URL
* ☐ wsp\_index.csv published at a stable URL
* ☐ Human-readable page matches machine-readable index
* ☐ (Recommended) Index validator run as part of release process

**Index anchoring (recommended)**

* ☐ Periodic index snapshots hashed and timestamped (IndexPack or equivalent)

**F.6 Operational security checklist (protect the credibility spine)**

WSP credibility depends heavily on the continuity and integrity of the provenance identity.

**Wallet role separation (recommended)**

* ☐ Provenance identity wallet is separate from treasury and operations wallets
* ☐ Provenance wallet used minimally (milestones only)

**Key protection**

* ☐ Hardware wallet used where feasible
* ☐ Secure backups exist (risk-managed)
* ☐ Access controls documented

**Incident readiness**

* ☐ Public “official identifiers” page exists (wallet roles + official links)
* ☐ Key rotation / migration policy exists
* ☐ Plan exists for how to publish a continuity milestone if migration is required

**Continuity**

* ☐ Critical bundles are mirrored
* ☐ Long-lived URLs are used where possible
* ☐ Documentation exists so the project can continue if a maintainer disappears

**F.7 Community norms quick reference (receipts-first culture)**

If you are building a WSP community (token or no token), these are the norms that keep it aligned with the scientific ideal:

* ☐ Major claims link to artifacts (bundle hash / Artifact ID)
* ☐ Criticism references specific artifacts and versions
* ☐ Audits and replications are celebrated—even when negative
* ☐ Corrections are treated as credibility-positive
* ☐ No harassment, no doxxing, no personal targeting
* ☐ Token price is not treated as proof
* ☐ Verification is not gated behind token ownership

**Appendix G — Public Statement Templates**

**Compliance declaration, provenance statements, and policy-aware submission language**

This appendix provides **copy-ready public statements** that projects can reuse when adopting Work Speaks Protocol (WSP). These templates are designed to:

* reduce ambiguity for outsiders,
* make compliance claims falsifiable,
* prevent identity confusion, and
* keep submission integrity disclosures policy-aware.

**Template notation:**  
This appendix uses angle‑bracket variables like <PROJECT\_NAME> and <PROVENANCE\_ID> as template variables that each project substitutes for its own implementation.

**G.1 WSP Compliance Declaration Template (Bronze / Silver / Gold)**

A WSP Compliance Declaration is a public statement that tells the world:

* where your canonical record lives,
* what provenance identity anchors your timestamps,
* what tier you claim, and
* what limitations apply.

Publish this on your canonical project home (or link it prominently), and keep old declarations accessible when you update.

**G.1.1 Short form (recommended)**

**Work Speaks Protocol (WSP) Compliance Declaration — <PROJECT\_NAME>**

* **Date (UTC):** <YYYY-MM-DD>
* **WSP Spec Version:** <WSP\_VERSION>
* **Claimed Tier:** <Bronze | Silver | Gold>
* **Canonical Home:** <CANONICAL\_HOME\_URL>
* **Artifacts Index:** <ARTIFACTS\_INDEX\_URL>
* **Provenance Identity:** <PROVENANCE\_WALLET\_OR\_ID>

**Verification quick-start:**

1. Visit the Artifacts Index.
2. Download any PUBLIC bundle.
3. Compute SHA‑256 of the zip and match it to the index hash.
4. Verify the timestamp reference contains that hash and is attributable to the provenance identity.

**Limitations / exceptions (if any):**

* <e.g., Certain DataPacks are HASH‑ONLY due to licensing/privacy constraints.>
* <e.g., Full reproduction requires specialized hardware; partial reproduction guidance provided.>

**Notes:**  
This compliance declaration describes **auditability and integrity discipline**, not scientific correctness.

**G.1.2 Extended form (recommended for Silver/Gold)**

**Work Speaks Protocol (WSP) Compliance Declaration — <PROJECT\_NAME>**

**1) Identity anchors**

* **Canonical Home:** <CANONICAL\_HOME\_URL>
* **Artifacts Index:** <ARTIFACTS\_INDEX\_URL>
* **Machine-readable index:**
  + JSON: <WSP\_INDEX\_JSON\_URL>
  + CSV: <WSP\_INDEX\_CSV\_URL>
* **Official Identifiers Page:** <OFFICIAL\_IDENTIFIERS\_URL> *(recommended)*
* **Verification Guide:** <VERIFY\_GUIDE\_URL> *(recommended)*
* **Provenance Identity:** <PROVENANCE\_WALLET\_OR\_ID>

**2) Claimed WSP tier**

* **Tier:** <Bronze | Silver | Gold>
* **WSP Spec Version:** <WSP\_VERSION>
* **Declaration date (UTC):** <YYYY-MM-DD>

**3) Evidence pointers (examples)**  
*(List the milestone entries that demonstrate the claimed tier.)*

* ReleasePack: <ARTIFACT\_ID\_OR\_LINK>
* RulesPack: <ARTIFACT\_ID\_OR\_LINK> *(Silver/Gold)*
* DataPack: <ARTIFACT\_ID\_OR\_LINK> *(Silver/Gold)*
* RunPack: <ARTIFACT\_ID\_OR\_LINK> *(Gold where applicable)*
* (Optional) Index anchoring entry: <ARTIFACT\_ID\_OR\_LINK>

**4) Constraints and disclosure policy**

* **HASH‑ONLY usage:** <Describe which categories are hash-only and why>
* **Redaction policy:** <Describe redaction approach, if used>
* **Submission integrity policy:**
  + “We publish our own submissions and outcomes policy-aware. We publish verbatim review text only when permitted.”

**5) Integrity statement**  
We follow WSP’s core principles:

* no silent rewrite,
* explain every hash,
* version discipline, and
* receipts-first norms.

This declaration is a public commitment. If any anchor changes (wallet migration, index relocation), we will record it as a WSP milestone and preserve continuity.

**G.2 WSP Provenance Statement Template (for papers, releases, or README files)**

This statement is designed to be pasted into:

* a paper appendix,
* a preprint “data/code availability” section,
* release notes, or
* the top of a repository README.

**WSP Provenance Statement — <PROJECT\_NAME>**  
This work is released under the Work Speaks Protocol (WSP). The canonical public record for this project is:

* **Canonical Project Home:** <CANONICAL\_HOME\_URL>
* **Artifacts Index (Proof Trail):** <ARTIFACTS\_INDEX\_URL>
* **Provenance Identity:** <PROVENANCE\_WALLET\_OR\_ID>

The Artifacts Index maps timestamped hashes to downloadable bundles (or documented hash-only constraints). To verify a milestone, download the referenced bundle, compute its SHA‑256 hash, and confirm it matches the index entry and its timestamp record.

This statement describes auditability and version integrity—not scientific correctness.

**G.3 Submission Integrity Statement Template (policy-aware)**

Use this if your project performs journal/conference submissions and wants to make its posture clear without provoking policy conflicts.

**WSP Submission Integrity Statement — <PROJECT\_NAME>**  
When we submit work to evaluation venues (journals, conferences, review services), we aim to preserve a policy-aware integrity trail:

* We create a **SubmitPack** (a submission package bundle) and timestamp its hash **before submission**.
* We record outcomes as **DecisionPacks** (decision type/date and a public summary).
* We publish verbatim reviewer/editor text only where venue policy or permission allows.
  + When confidentiality applies, we may store verbatim materials as **HASH‑ONLY** while publishing a public summary.

We do not disclose reviewer identities, and we do not encourage harassment or personal targeting. Critique should focus on artifacts, methods, and reasoning.

Our canonical submission integrity record (when applicable) is maintained in the project’s Artifacts Index:  
<ARTIFACTS\_INDEX\_URL>

**G.4 Token / Community Statement Template (receipts-first participation)**

This template is designed to clarify what the token layer is for and prevent “price = truth” confusion.

**WSP Community Participation Statement — <PROJECT\_NAME>**  
<PROJECT\_NAME> includes a community participation layer intended to support:

* public participation in the project network,
* coordination around what should be audited, replicated, or tested next, and
* sustained support for serious work.

This participation layer is **receipts-first**: major claims should map to verifiable artifacts in our Proof Trail.

**Important boundaries:**

* Participation is not proof of correctness.
* Token dynamics are not treated as scientific validation.
* The Proof Trail is the audit surface; the community layer is coordination.
* We do not gate essential verification behind token ownership.

**Official identifiers:**

* Canonical Home: <CANONICAL\_HOME\_URL>
* Artifacts Index: <ARTIFACTS\_INDEX\_URL>
* Provenance Identity: <PROVENANCE\_WALLET\_OR\_ID>
* Official Token Identifiers (if applicable): <TOKEN\_NAME / SYMBOL / CONTRACT / NETWORK>
* Official channels: <OFFICIAL\_CHANNELS\_URL>

*(General note: this statement is informational and does not constitute financial or legal advice.)*

**G.5 Wallet / Identity Migration Notice Template (continuity under key rotation)**

If you ever need to change your provenance identity, you should make the migration auditable and hard to impersonate. Publish this as a website notice **and** record it as a WSP milestone bundle.

**WSP Provenance Identity Migration Notice — <PROJECT\_NAME>**

* **Date (UTC):** <YYYY-MM-DD>
* **Reason for migration:** <lost key / compromise / operational upgrade / other>
* **Old provenance identity:** <OLD\_PROVENANCE\_ID>
* **New provenance identity:** <NEW\_PROVENANCE\_ID>
* **Canonical Home:** <CANONICAL\_HOME\_URL>
* **Artifacts Index:** <ARTIFACTS\_INDEX\_URL>

**Continuity statement:**  
All previous milestone entries remain valid and are preserved in the Artifacts Index. Future WSP milestones will be timestamped under the new provenance identity. This migration is recorded as a WSP milestone bundle with a timestamped hash, so third parties can verify continuity.

**Security note:**  
Ignore claims that conflict with the canonical home and Artifacts Index. Official identifiers are listed here: <OFFICIAL\_IDENTIFIERS\_URL>

**G.6 Genesis Identity Declaration Template (Recommended)**

**WSP Genesis Identity Declaration — <PROJECT\_NAME>**

* **Date (UTC):** <YYYY-MM-DD>
* **Protocol:** Work Speaks Protocol (WSP)
* **Project ID:** <PROJECT\_ID>
* **Canonical Home:** <CANONICAL\_HOME\_URL>
* **Artifacts Index (Proof Trail):** <ARTIFACTS\_INDEX\_URL>
* **Provenance Identity:** <PROVENANCE\_WALLET\_OR\_ID>

**Official token identifiers (if applicable):**

* **Token name/symbol:** <TOKEN\_NAME / SYMBOL>
* **Network:** <NETWORK>
* **Contract / mint address:** <TOKEN\_ADDRESS>
* *(If multiple networks, list each separately.)*

**Notes:**  
This declaration identifies the project’s official anchors for verification and impersonation resistance. Future changes to these identifiers will be recorded as additive WSP milestones (no silent rewrite).

**Appendix H — Threat Model and Risk Considerations**

**What WSP defends against, what it cannot, and how to handle incidents**

Work Speaks Protocol (WSP) improves credibility by making declared work **auditable**: bundles are committed with hashes, timestamped under a public provenance identity, and mapped in a legible Artifacts Index. This raises the cost of deception and lowers the cost of verification.

However, WSP is not magic. It is not a truth oracle, and it does not eliminate adversaries, mistakes, platform failures, or social manipulation. This appendix clarifies the threat model WSP assumes, the most common failure modes, recommended mitigations, and how to respond when something goes wrong.

**H.1 Security objectives**

**What WSP is trying to protect**

WSP’s security posture is centered on four practical objectives:

1. **Integrity of released artifacts**  
   Outsiders can verify that a downloaded bundle matches the declared hash.
2. **Timeline integrity of declarations**  
   Outsiders can verify that a given bundle hash was publicly recorded by time **T** under the project’s provenance identity.
3. **Legibility and continuity**  
   Outsiders can follow the project’s declared history over time (no silent rewrite), with clear supersession and correction paths.
4. **Resistance to impersonation and confusion**  
   Outsiders can identify “what is official” using stable identity anchors (canonical home, Artifacts Index, provenance identity, and—if applicable—official token identifiers).

These objectives do **not** guarantee correctness of conclusions. They protect the *audit surface*.

**H.2 Adversary model**

**Who might attack, and why**

WSP assumes adversaries can exist for many reasons, including:

* **Impersonators / scammers** seeking to exploit public attention (especially if a token exists).
* **Malicious critics or rivals** attempting to discredit a project by forging artifacts or spreading confusion.
* **Opportunists** attempting to benefit from misinformation (social or economic).
* **Platform-level failures** (hosting outages, link rot, censorship, account takeovers).
* **Internal mistakes** (mis-hashed files, incorrect index entries, poor key security, accidental overwrites).

WSP assumes adversaries may have these capabilities:

* copying your website design and publishing lookalike “official” pages,
* phishing community members into trusting wrong addresses or wrong downloads,
* attempting to compromise wallets/keys,
* attempting to replace bundles on a hosting platform,
* attempting to create false narratives about what was “really” done.

WSP assumes adversaries do **not** have unlimited power. Specifically, WSP’s verification logic remains robust as long as:

* your provenance identity remains trustworthy (not compromised), and
* your mapping layer is clear and consistently maintained.

**H.3 What WSP mitigates well**

**H.3.1 “Bundle substitution” after release (integrity drift)**

If someone changes a hosted file (or swaps it on a mirror), WSP verification catches it because:

* the new bytes will not match the declared hash, and
* auditors can demonstrate the mismatch objectively.

**H.3.2 “Revisionist history” inside the project**

WSP strongly discourages quiet rewriting because:

* milestones are additive,
* older hashes remain visible,
* supersession is explicit, and
* corrections become public versions rather than silent edits.

**H.3.3 Ambiguity about what a claim refers to**

WSP reduces the “what version are you talking about?” problem by encouraging:

* pack types (RulesPack, RunPack, ReleasePack, etc.),
* stable artifact IDs, and
* explicit relationships in the index (uses/supports/supersedes).

This improves debate quality and reduces narrative-driven confusion.

**H.4 What WSP cannot prove (and does not claim to)**

WSP cannot guarantee any of the following:

**H.4.1 Correctness of results or truth of conclusions**

A timestamped bundle can still be wrong. Integrity is not correctness.

**H.4.2 Completeness of the private process**

WSP can only prove what the project **declared**. A project can always do private exploration before publishing milestones. WSP raises the cost of deception, but it does not eliminate it.

**H.4.3 That a community signal equals reality**

Token/community participation can coordinate attention and support, but it is not a truth mechanism. WSP explicitly rejects “price = proof.”

**H.4.4 That institutions behaved fairly**

WSP can make submissions and outcomes more legible over time (policy-aware), but it cannot prove intent or fairness in individual editorial choices.

**H.5 Common failure modes and recommended mitigations**

This section lists practical risks and what WSP recommends doing about them.

**H.5.1 Risk: Provenance identity compromise (worst-case credibility event)**

If an attacker gains control of the provenance wallet/key, they can:

* publish timestamp records that look “official,”
* confuse outsiders about what is real, and
* damage the project’s credibility spine.

**Mitigations (SHOULD):**

* Use a **dedicated provenance wallet** used only for milestones.
* Protect it with a **hardware wallet** where feasible.
* Keep the provenance wallet **offline and unused** except for publishing milestones.
* Publish a **wallet roles page** (provenance vs treasury vs operations).
* Publish a **key rotation / incident policy** in advance.
* Avoid signing arbitrary messages or doing day-to-day operations with the provenance wallet.

**Residual reality:** even with strong discipline, compromise is possible. WSP therefore emphasizes continuity planning (H.6).

**H.5.2 Risk: Hosting-level compromise or mirror drift**

If your hosting provider is compromised or misconfigured, an attacker might:

* replace downloadable bundles,
* break links,
* or selectively serve different bytes to different users.

**Mitigations (SHOULD):**

* Mirror important bundles on **at least two independent hosts**.
* Publish size\_bytes in the index to help detect partial downloads.
* Encourage verifiers to **hash locally** and compare to the index.
* Keep local, immutable copies of released bundles (exact bytes).
* Prefer stable archival hosts for major releases when possible.
* Periodically run link checks and publish corrections as new index entries (superseding when needed).

**H.5.3 Risk: Index tampering or “quiet edits”**

A project might accidentally (or intentionally) edit index entries in ways that create confusion or retrospective rewriting.

**Mitigations (SHOULD):**

* Treat the index as **append-only** in spirit:
  + new milestones add new entries,
  + corrections supersede older entries rather than rewriting them silently.
* Publish machine-readable index files (wsp\_index.json/csv) generated from a canonical source.
* Periodically **anchor index snapshots** (hash + timestamp).
* Use automated validators to check schema completeness and link integrity.

**H.5.4 Risk: Impersonation and “official identity confusion”**

Attackers may create fake websites, fake social accounts, or fake token identifiers that look official.

**Mitigations (SHOULD):**

* Publish a canonical **Official Identifiers** page listing:
  + canonical home,
  + Artifacts Index URLs,
  + provenance identity,
  + official token identifiers (if any),
  + official community channels.
* Repeat canonical links inside every PUBLIC bundle README.md.
* Record identity changes (new wallets, migrations) as WSP milestones.

**Cultural rule:**  
If it isn’t anchored at the canonical home + Artifacts Index + provenance identity, it isn’t official.

**H.5.5 Risk: Timestamp ambiguity (hash not clearly discoverable in the record)**

If timestamp records are hard to interpret, outsiders can’t verify them reliably.

**Mitigations (SHOULD):**

* Use a timestamp method where the hash is **clearly recoverable**.
* In the Artifacts Index, include:
  + timestamp method label,
  + timestamp reference ID,
  + and a short verification hint (“where in the record the hash appears”).
* Provide a non-technical verification guide (Appendix E) linked from the index.

**H.5.6 Risk: Selective disclosure and “hash-only overuse”**

HASH-ONLY milestones are legitimate when privacy/licensing/policy requires it, but overuse can degrade verifiability.

**Mitigations (SHOULD):**

* Prefer **PUBLIC** bundles whenever feasible.
* Use HASH-ONLY sparingly and always provide a clear hash\_only\_reason.
* When possible, publish a **REDACTED-PUBLIC** bundle instead of fully hash-only.
* For restricted data, publish instructions on how auditors can legally obtain the same inputs.

**H.5.7 Risk: Social manipulation and narrative warfare**

A project may be attacked socially through misquotes, selective framing, or hype cycles.

**Mitigations (SHOULD):**

* Keep communications receipts-first:
  + “Here is the artifact.”
  + “Here is the version.”
  + “Here is the timestamp record.”
* Encourage third-party audits and publish AuditPacks prominently.
* Maintain a Credibility Scorecard independent of token price.
* Apply clear community conduct standards (no harassment, no doxxing).

**H.6 Incident response and continuity**

**What to do when something goes wrong**

A credible WSP project assumes incidents can happen and prepares a clear response.

**H.6.1 Minimum incident response steps (recommended)**

If you suspect compromise, corruption, or serious integrity failure:

1. **Stop and preserve evidence**

* Freeze the release pipeline temporarily.
* Preserve current index files and relevant logs.
* Avoid “quick edits” that destroy traceability.

1. **Publish a clear incident notice**

* Post on canonical home (and official channels) what happened, what is known, and what is uncertain.
* Avoid speculation. Separate facts from hypotheses.

1. **Record the incident as a WSP milestone**

* Publish an IncidentPack (or ReleasePack) that includes:
  + incident summary,
  + affected artifact IDs/hashes,
  + remediation actions,
  + continuity plan.

1. **Repair continuity**

* If wallet compromise is suspected, execute the published key rotation plan:
  + migrate to a new provenance identity,
  + publish a migration notice (Appendix G),
  + record the migration as a milestone.

1. **Restore verifiability**

* Re-establish stable mirrors for PUBLIC bundles.
* Update the index additively (supersede entries if required).
* Publish verification instructions for the new continuity state.

**H.6.2 What “good behavior” looks like under WSP**

Under WSP, the best response to failure is:

* public, calm disclosure,
* additive records (no deletion),
* explicit supersession,
* and fast restoration of verifiability.

Transparent incident handling is credibility-positive—because it demonstrates the protocol’s core ethic: **integrity under pressure**.

**H.7 Summary: what makes WSP robust**

WSP is robust not because it guarantees truth, but because it creates a disciplined, auditable environment where:

* artifacts can be verified independently,
* version history remains legible,
* corrections are additive rather than hidden,
* and identity anchors reduce confusion.

When combined with a receipts-first culture and a community that values audits, WSP turns many credibility disputes into something simpler:

* **Download, hash, compare, verify the timestamp, and publish the audit.**

That is the practical purpose of the protocol.

**End of Appendix H.**