2025

FTU-SPEC v1.7.1.1 — Fadi Tempo Unit Canonical Specification (MASTER UNIVERSAL FLAGSHIP ABSOLUTE)

BY FADI GHALI

Status: Final Specification (Hash-Sealed Release Package; signature optional)

Spec Version: 1.7.1

Release ID: FTU-SEAL-2025-12-31-v1.7.1

Index (Quick Navigation)

Contents

0. Abstract

This document defines **FTU (Fadi Tempo Unit)** as a canonical base time unit for FTU-compliant specifications and implementations.

FTU is **normatively defined** to be equal to **Planck time** (*t_p*), and provides conversion rules for:

- **Durations ↔ FTU counts**
- **Frequencies (Hz) ↔ FTU-period counts**
- **Domain tempo units (DTU)** as declared multiples of FTU for hardware/biological clocks
- **Compliance metadata** exposure for machine verification (file or /.well-known endpoint)
- **Normative conformance vectors** for testability

This specification is designed for machine-ingestible reproducibility: it pairs naturally with a reference implementation (ftu-core),

conformance vectors (ftu-test), and a public verifier (ftu-verify) bound by a signed **Master Manifest**.

0.0 Problem, Solution, Why Now (NON-NORMATIVE)

Problem. Modern systems (AI, telemetry, media pipelines, sensors, distributed logs) often use incompatible "ticks" and hidden assumptions when stating durations, rates, and timing claims. This makes cross-system results hard to compare, reproduce, or audit.

Solution. FTU provides a canonical base tick anchored to **Planck time** (*t_p*), plus explicit rules for:

- converting between **seconds ↔ FTU** without ambiguity,
- publishing **Domain Tempo Units (DTU)** as exact FTU multiples for real-world clocks,
- shipping **machine-verifiable manifests + conformance vectors** so any third party can verify the release.

Why now. Verification is becoming the default expectation: standards that are **hash-addressable, reproducible, and independently checkable** adopt faster than narrative-only documents.

Design goals

- **Interoperability:** consistent timing claims across domains and stacks.
- **Auditability:** independent verification without private context.
- **Disclosure:** constants and assumptions must be explicit and testable.
- **Longevity:** stable references suitable for long-horizon archival citation.

Non-goals

- Proposing "new physics," replacing SI, or claiming empirical discreteness of time.
- Forcing a single implementation language or platform.
- Substituting for uncertainty analysis in measurements (FTU is a canonicalization layer).

0.1 Concept + Formula Primer (NON-NORMATIVE, READ FIRST)

Origin note (GULF Law coherence). In the GULF Law framework, **t_f** denotes the foundational tempo variable. This specification makes that quantity operational and verifiable by anchoring it to a canonical base tick: **t_f ≡ FTU ≡ t_p** (Planck time). Under this anchoring, any duration can be expressed as a **dimensionless tick-count** *$N = t / t_p$*, enabling cross-domain comparability and reproducible verification.

FTU-SPEC defines **FTU (Fadi Tempo Unit)** as a **canonical base time unit** for cross-system timing/tempo declarations.

Normatively:

- **FTU := t_p** (Planck time)
- **$t_p = \sqrt{\hbar G / c^5}$**

The four core quantities (at a glance)

1. **Duration in FTU (count of FTU ticks)**

   - If a duration is **T seconds**, then:
     - **N_FTU := T / t_p**
     - and **T = N_FTU · t_p**

2. **Frequency in FTU-period counts**

   - If a signal is **f Hz**, its period is **1/f seconds**. In FTU:
     - **T_FTU := (1/f) / t_p = 1 / (f · t_p)**
     - and **f = 1 / (T_FTU · t_p)**

3. **Domain Tempo Unit (DTU) as an exact FTU multiple**

   - A DTU is an **exact rational multiple** of FTU:
     - **DTU := (p/q) · FTU** where **p, q** $\in \mathbb{N}$ and **q ≠ 0**
   - This lets domains define a preferred “tick” while remaining interoperable.

4. **The bridge lemma (dimensionless time)**

   - Define a dimensionless counter:
     - **N := t / t_p**
   - Then **t = N · t_p**, and any equation that uses **t** can be rewritten in **N** plus constant factors.

Boundary (important)

This specification does **not** claim to change physical theories or assert that time is discrete. It standardizes a

verifiable measurement convention for timing/tempo declarations across implementations.

0.2 Historical Partition: Before and After the Sealing Release (non-normative)

This section is **archival context** for readers and search engines. It is not required for implementation conformance.

Before this sealing release

Before FTU-SPEC exists as a sealed, machine-verifiable standard, systems typically:

- choose an internal "tick" or sampling period (CPU cycles, oscillator periods, frame times, audio sample rates, etc.),
- publish limited or non-standard metadata about the chosen clock, and
- lack a uniform way to compare timing/tempo claims across domains (hardware, software, media, and bio-like oscillators).

This pre-standard state is workable locally, but it is brittle for *cross-system* verification and long-horizon archival.

After this sealing release

After FTU-SPEC is published and referenced as a canonical standard, any system can:

- declare a Domain Tempo Unit (DTU) in exact FTU multiples,
- expose a minimal, well-known metadata record for independent verification, and
- prove conformance via shared test vectors and a sealed release manifest.

The intent is not to "rewrite physics", but to provide a stable, auditable **common ruler** for time/tempo declarations across implementations.

0.3 Canonicalization Rules and Precedence (NORMATIVE)

1. **Canonical definition**: FTU is defined as Planck time (t_p; ASCII: tp or t_p) as declared in this specification. (See §2.1 and §3.)
2. **Spec precedence**: If any downstream document conflicts with this specification, **this specification prevails**.
3. **Version pinning**: Implementations MUST declare the FTU-SPEC version they implement (e.g., 1.5) in their metadata. (See §8.)
4. **Sealed releases**: A release claiming to be "Sealed" MUST publish the seal fields in §13 AND a sealed release manifest that can be independently verified. (See §14.)
5. **Citable identifier**: A sealed release SHOULD publish a stable citation string containing at least (spec_version, release_id, manifest_sha256).

1. Conformance Language

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL**

in this document are to be interpreted as described in **RFC 2119** and **RFC 8174** (only when they appear in all capitals).

1.1 Glossary and Symbol Table (NON-NORMATIVE)

Terms

- **FTU**: Fadi Tempo Unit, normatively equal to Planck time ($\mathbf{t_p}$).
- **DTU**: Domain Tempo Unit, an exact rational multiple of FTU used by a domain as its preferred tick.
- **Sealed release**: a published FTU-SPEC release whose artifacts are listed in a manifest with hashes (and optionally signatures) enabling independent verification.
- **Artifact**: a released file covered by the manifest (e.g., this specification text, test vectors, schemas).

- **Verifier**: a tool or party that checks the manifest and hashes and concludes Sealed/Not Sealed.

Symbols

Symbol	Meaning	Units	Notes
t_p	Planck time	seconds	Canonical FTU definition anchor
FTU	Fadi Tempo Unit	seconds	**FTU := t_p** (normative)
T	Duration	seconds	Any real-world duration
N_FTU	Duration expressed in FTU	(count)	**N_FTU = T/t_p**
f	Frequency	Hz	**f = 1/T**
T_FTU	Period expressed in FTU	(count)	**T_FTU = (1/f)/t_p**
DTU	Domain Tempo Unit	seconds	**DTU = (p/q)·FTU**
N	Dimensionless time counter	unitless	**N = t/t_p** (bridge lemma)

2. Normative Definitions

2.1 FTU (Fadi Tempo Unit)

FTU SHALL be defined exactly as Planck time *(t_p; ASCII: tp or t_p)*:

$$FTU := t_p$$

where Planck time is:

$$t_p = \sqrt{\hbar G/c^5}$$

Implementations MUST treat this definition as the canonical meaning of FTU in FTU-compliant contexts.

2.2 Non-normative note (scientific interpretation)

This specification's use of Planck time is a **measurement convention** grounded in standard Planck units.

It **does not claim** that time is proven discrete at *t_p*, nor that intervals smaller than *t_p* are physically impossible.

3. Constants and Reference Values

3.1 Planck time numeric value (reference)

The normative definition is symbolic (**FTU := t_p**). For numeric conversions, implementations SHOULD use a recognized constant source (e.g., NIST/CODATA)

and MUST declare the constant set used (see §3.2).

- **Reference value:** *t_p* ≈ **5.391247×10^{-44} s**

- **Reference uncertainty (1σ):** ≈ **6.0×10^{-49} s** (as published by NIST/CODATA)

Canonical reference (online): https://physics.nist.gov/cgi-bin/cuu/Value?plkt=

> *NOTE: If a different constant set or revision is used, the implementation MUST disclose it.*

3.2 Mandatory constant metadata (normative)

Any implementation that claims **FTU-compliance** MUST expose at minimum:

- tp_source (e.g., "NIST/CODATA")
- tp_value_seconds (value used; see §3.3 encoding rules)
- tp_uncertainty_seconds (if provided by the source; else null with explanation)
- tp_reference_url (source URL)
- constants_revision (e.g., "CODATA 2022" or equivalent identifier)

3.3 JSON numeric encoding rules (normative)

To prevent false non-compliance from floating-point serialization:

- tp_value_seconds and tp_uncertainty_seconds **SHOULD** be encoded as **decimal strings**, e.g. "5.391247e-44".
- If encoded as JSON numbers, the implementation **MUST** also publish:
  - precision_digits (decimal digits of precision guaranteed), and
  - rounding_policy (§4.2).

4. Duration Representation

4.1 FTU count

For any duration **T** in seconds:

- **FTU count:**

$$N\_FTU = T / t_p$$

Implementations:

- SHOULD support high-precision arithmetic.
- SHOULD prefer integer or rational representations when derived from declared clocks or symbolic inputs (see §5).
- MAY use high-precision decimal representations when values originate from measurement, provided rounding and constant metadata are declared.

4.2 Rounding policy (normative)

If an implementation must round, it MUST declare:

- rounding_policy = "nearest_even" | "floor" | "ceil" | "truncate"
- precision_digits or precision_bits (as applicable)

5. Domain Tempo Unit (DTU) for Real Systems (Normative)

FTU is fixed by definition. Real hardware/biological systems operate on practical ticks.

To prevent incorrect interpretation that FTU "shrinks" per system, this specification defines:

DTU := k × FTU

where:

- **k** is a positive integer, or a rational number expressed as a fraction **num/den**,
- **k MUST be declared** in system metadata.

5.1 DTU metadata (minimum)

Implementations that use DTU MUST expose at least:

- dtu_to_ftu_ratio as an object:
  - num (integer ≥ 1)
  - den (integer ≥ 1)
- domain_name (e.g., "display_refresh", "audio_sample_clock", "cpu_clock", "biological_heart")
- domain_clock_hz (if derived from a clock)
- rounding_policy
- constants_revision

Backward compatibility: Implementations MAY also expose dtu_to_ftu_ratio_k (string or decimal) for human readability, but dtu_to_ftu_ratio is the canonical machine field.

6. Frequency and Refresh Rates (Normative)

Any periodic phenomenon with frequency **f** (Hz) has period **T = 1/f**.

6.1 FTU-period count

- **T_FTU = (1/f) /** $\mathbf{t_p}$

6.2 Convert back to frequency

Given an FTU-period count **T_FTU**:

- **f = 1 / (T_FTU ×** $\mathbf{t_p}$**)**

Guardrail: These conversions are representational; they do not alter device physics.

7. Planck-Scale Normalization (Informative, non-normative)

Define a dimensionless "fraction of Planck scale" for frequency:

- **Cycle-frequency reference (convention): $f_p := 1/t_p$**
- **Normalized cycle frequency: $\nu := f \cdot t_p = f/f_p$**

7.1 Angular-frequency clarification

Some domains use **angular frequency ω** (radians/second), related to cycle frequency **f** (cycles/second) by:

- **$\omega = 2\pi f$**

Accordingly, a convenient angular-frequency reference scale is:

- **$\omega_p := 1/t_p$**

These choices differ by a factor of **2π**:

- If **$\omega_p := 1/t_p$**, then the corresponding cycle-frequency reference is **$f_{p,alt} := 1/(2\pi\, t_p)$**.

This specification permits either convention for *normalization* so long as the implementation discloses which was used:

- normalization_mode = "cycle" | "angular"
- If "cycle", publish fp = 1/tp.
- If "angular", publish wp = 1/tp and use omega = 2*pi*f.

Important: These are **normalization conventions for computation and comparison**. They MUST NOT be stated as proof of hard physical maxima.

7.2 Planck energy scale (definition)

For energy normalization, define the Planck energy scale symbolically as:

- **$E_p := \sqrt{\hbar c^5/G}$** (equivalently, **$E_p = m_p c^2$**)

For quanta, both forms may appear depending on frequency representation:

- **$E = h f$** (cycle frequency)
- **$E = \hbar \omega$** (angular frequency)

Define a dimensionless normalized energy:

- **$\varepsilon := E / E_p$**

These are normalization conventions useful for comparative modeling and preventing "unbounded" numeric narratives in simulations.

They MUST NOT be stated as proof of hard physical maxima.

8. Security, Integrity, Resource Limits, and Metadata Exposure (Normative)

8.1 Evidence-based verification

Implementations MUST NOT claim verification or compliance unless evidence is produced (e.g., conformance test report, verifier output, or manifest validation).

8.2 Resource limits (normative)

Because FTU counts can be extremely large, implementations MUST publish the following limits:

- max_span_seconds (maximum duration span accepted for conversion)
- max_precision_digits (maximum precision supported for decimal operations)
- max_bigint_bits (maximum bits for integer representations, where applicable)

Implementations MUST fail safely (clear error) when limits are exceeded.

8.3 Mandatory metadata exposure location (normative)

An implementation that claims **FTU-compliance** MUST expose the metadata required by §§3.2–3.3, 4.2, 5.1, and 8.2 in at least one of:

1. a machine-readable JSON file named **`ftu_meta.json`** distributed with the implementation; OR
2. an HTTPS endpoint at **`/.well-known/ftu/meta`** returning the same JSON.

If both are provided, they MUST be consistent.

8.4 Well-known endpoint requirements (normative)

If /.well-known/ftu/meta is provided, it MUST:

- comply with the Well-Known URI convention (RFC 8615),
- return Content-Type: application/json,
- return an integrity validator: either ETag **or** Digest header,
- support GET without authentication for read-only metadata retrieval.

8.5 Minimal metadata schema (recommended)

```
{
 "ftu_spec_version": "1.7.1",
 "tp_source": "NIST/CODATA",
 "tp_value_seconds": "5.391247e-44",
 "tp_uncertainty_seconds": "6.0e-49",
```

```
 "tp_reference_url": "https://physics.nist.gov/cgi-bin/cuu/Value?plkt=",
 "constants_revision": "CODATA 2022",
 "rounding_policy": "nearest_even",
 "precision_digits": 80,
 "dtu_to_ftu_ratio": { "num": 1, "den": 1 },
 "domain_name": null,
 "domain_clock_hz": null,
 "normalization_mode": "cycle",
 "max_span_seconds": 1.0e9,
 "max_precision_digits": 200,
 "max_bigint_bits": 1000000
}
```


9. Licensing, Certification, and Trademarks (Policy)

9.1 Spec license

This specification text is licensed **CC BY 4.0**.

9.2 Reference implementation licensing (recommended)

The reference implementation (`ftu-core`) is recommended as **dual-licensed**:

- **AGPLv3** for open deployments, OR
- **Commercial license** for closed-source embedding.

9.3 Trademarks and "FTU Certified" mark

This document does not grant permission to use trademarks. Use of **FTU Certified** SHOULD be conditioned on:

- passing `ftu-test` conformance vectors, and
- agreeing to the trademark license terms.

10. Versioning and Compatibility

- **FTU-SPEC 1.x**: backward compatible within major version.
- Any change to the normative FTU definition is forbidden (FTU := t_p is fixed).
- Enhancements MUST be added as optional annexes or extensions.

11. References (non-exhaustive)

- NIST/CODATA Planck time (Web Version 9.0): https://physics.nist.gov/cgi-bin/cuu/Value?plkt
- CODATA 2022 recommended values of the fundamental physical constants: https://pubs.aip.org/aip/jpr/article/54/3/033105/3363695/

- Planck units background (non-normative): https://en.wikipedia.org/wiki/Planck_units
- SciPy constants (CODATA-derived): https://docs.scipy.org/doc/scipy/reference/constants.html
- RFC 2119 (requirement words): https://www.rfc-editor.org/rfc/rfc2119
- RFC 8174 (uppercase interpretation): https://www.rfc-editor.org/rfc/rfc8174
- RFC 8615 (Well-Known URIs): https://www.rfc-editor.org/rfc/rfc8615

12. Conformance Test Vectors (NORMATIVE)

A compliant implementation MUST pass the vectors below **within the declared relative tolerance** using the declared rounding policy.

Normative artifact: The exact same vector set MUST be published alongside sealed releases as:

- ftu_test_vectors_v1.7.1.json with SHA-256: cebbdeb47811f9cb95161a647a863ed7c2e3b19e89849739a1f8cc2a4203e7ea


```
{
 "ftu_spec_version": "1.7.1",
 "tp_value_seconds": "5.391247e-44",
 "rounding_policy": "nearest_even",
 "precision_digits_internal": 80,
 "vectors": [
  {
   "id": "dur_1s_to_ftu",
   "kind": "duration_to_ftu",
   "input_seconds": "1",
   "expected_ftu": "1.85485843998615e43",
   "tolerance_rel": "1e-12"
  },
  {
   "id": "dur_1ms_to_ftu",
   "kind": "duration_to_ftu",
   "input_seconds": "0.001",
   "expected_ftu": "1.85485843998615e40",
   "tolerance_rel": "1e-12"
  },
  {
   "id": "dur_1min_to_ftu",
   "kind": "duration_to_ftu",
   "input_seconds": "60",
   "expected_ftu": "1.11291506399169e45",
   "tolerance_rel": "1e-12"
  },
```

```
{
  "id": "dur_1hour_to_ftu",
  "kind": "duration_to_ftu",
  "input_seconds": "3600",
  "expected_ftu": "6.67749038395013e46",
  "tolerance_rel": "1e-12"
},
{
  "id": "hz_1_to_tftu",
  "kind": "hz_to_ftu_period",
  "input_hz": "1",
  "expected_tftu": "1.85485843998615e43",
  "tolerance_rel": "1e-12"
},
{
  "id": "hz_60_to_tftu",
  "kind": "hz_to_ftu_period",
  "input_hz": "60",
  "expected_tftu": "3.09143073331025e41",
  "tolerance_rel": "1e-12"
},
{
  "id": "hz_59_94_to_tftu",
  "kind": "hz_to_ftu_period",
  "input_hz": "59.94",
  "expected_tftu": "3.09452525856882e41",
  "tolerance_rel": "1e-12"
},
{
  "id": "hz_120_to_tftu",
  "kind": "hz_to_ftu_period",
  "input_hz": "120",
  "expected_tftu": "1.54571536665512e41",
  "tolerance_rel": "1e-12"
},
{
  "id": "hz_44100_to_tftu",
  "kind": "hz_to_ftu_period",
  "input_hz": "44100",
  "expected_tftu": "4.20602820858537e38",
  "tolerance_rel": "1e-12"
},
{
```

```
    "id": "hz_48000_to_tftu",
    "kind": "hz_to_ftu_period",
    "input_hz": "48000",
    "expected_tftu": "3.86428841663781e38",
    "tolerance_rel": "1e-12"
  },
  {
    "id": "hz_3_2ghz_to_tftu",
    "kind": "hz_to_ftu_period",
    "input_hz": "3200000000",
    "expected_tftu": "5.79643262495671e33",
    "tolerance_rel": "1e-12"
  },
  {
    "id": "tftu_back_to_hz_60",
    "kind": "ftu_period_to_hz",
    "input_tftu": "3.09143073331025e41",
    "expected_hz": "60",
    "tolerance_rel": "1e-12"
  }
 ]
}
```


Notes (normative):

- Implementations MUST treat values as **decimal strings** (do not rely on binary floats).
- Tolerances are **relative** (|got-expected|/|expected| ≤ tolerance_rel).
- A sealed release SHOULD publish the verifier script hash in the manifest (Annex F).

13. Seal Fields (NORMATIVE)

> ***Rule:** A release MUST NOT be described as **Sealed** unless the fields below are **populated** and **verifiable** (via the manifest in §14 and any published signature material, if used).*

A sealed FTU release MUST publish a **Seal Card** (plain text or JSON) that contains at minimum:

- **release_id** (string): a stable identifier (example: FTU-SEAL-2025-12-31-v1.7.1).
- **manifest_sha256** (hex, lowercase): SHA-256 of the published ftu_manifest.json.
- **spec_artifact_sha256** (hex, lowercase): SHA-256 of the published canonical spec artifact (PDF or Markdown).
- **verifier_reference** (URL or repository path): where a verifier can be obtained or executed.

- **conformance_report_reference** (URL or repository path): where the conformance report (PASS/FAIL) is published.
- **signature_method** (enum): gpg | sigstore | none.
- **signature_reference** (optional): signature block or URL (required if signature_method != none).
- **identity_reference** (optional): public-key fingerprint, certificate identity, or transparency-log entry (required if signature_method != none).

Template: See **Annex G** for an informative Seal Card template.

14.Sealed Release Manifest (NORMATIVE)

A sealed FTU release MUST ship a machine-readable manifest that binds the release ID, the spec version, and the exact artifact hashes. This enables independent verification without private context.

14.1 Required file and location

A sealed release MUST publish a manifest at one of the following locations:

- repository root as ftu_manifest.json, OR
- /.well-known/ftu/manifest.json (recommended for hosted deployments; see RFC 8615).

14.2 Minimal schema (normative)

A manifest MUST be a JSON object with at minimum:

- **release_id** (string): the stable release identifier.
- **spec_version** (string): the specification version (e.g., "1.7.1").
- **artifacts** (array): each entry MUST contain:
  - **path** (string): relative path within the release bundle or repository.
  - **media_type** (string): MIME type.
  - **sha256** (string): lowercase hex SHA-256 of the artifact bytes.
  - **size_bytes** (integer): exact byte length.

Optionally, a manifest MAY include a **seal** object with:

- **manifest_sha256** (string): SHA-256 of the manifest itself (for convenience).
- **signature_method** (string): gpg | sigstore | none.
- **signature_ref** (string, optional): URL or inline signature (required if signed).
- **identity_ref** (string, optional): public identity reference (required if signed).

Implementations MUST treat the artifacts[*].sha256 values as the canonical integrity commitments.

14.3 Verification rules (normative)

A verifier MUST:

1. fetch the manifest,

2. hash each referenced artifact, and
3. confirm all hashes match.

If any required field is missing or a hash mismatch occurs, the release MUST be treated as **Not Sealed**.

14.4 How to Seal a Release (NON-NORMATIVE, IMPLEMENTER CHECKLIST)

This checklist operationalizes the normative sealing rules above.

1. **Freeze artifacts** (no further edits): at minimum the spec text (MD/DOCX) and any referenced test vectors.
2. **Compute hashes** for each artifact (e.g., SHA-256).
3. **Create `ftu_manifest.json`** containing:
    - `release_id` (stable, date-stamped),
    - `spec_version`,
    - `artifacts[]` with `{path, sha256, bytes}` (recommended),
    - and an optional `signature` block.
4. **Sign the manifest** (recommended):
    - use a verifiable scheme (e.g., minisign, SSH sig, or GPG),
    - publish the public key / identity reference used to verify.
5. **Publish the sealed release**:
    - tag a GitHub release (recommended),
    - attach the artifacts + manifest,
    - mirror the manifest at `/.well-known/ftu/manifest.json` for hosted deployments.
6. **Run a verifier** end-to-end:
    - fetch manifest,
    - hash artifacts,
    - compare hashes,
    - verify signature (if present),
    - publish a short conformance report URL.

Example (illustrative):

```
# hash artifacts
sha256sum FTU-SPEC_v1.7_MASTER_UNIVERSAL_PUBLISH_OFFICIAL_FINAL.md \
     FTU-SPEC_v1.7_MASTER_UNIVERSAL_PUBLISH_OFFICIAL_FINAL.docx

# verify manifest integrity (implementation-dependent)
cat ftu_manifest.json | jq .
```

These steps ensure third parties can independently conclude **Sealed** vs **Not Sealed** using only public artifacts.

Annex E — Conformance Profiles (NORMATIVE)

This annex defines conformance **levels** so adopters can implement FTU in phases while remaining auditable.

E.1 Profiles

FTU-CORE (Minimum)

An implementation claiming **FTU-CORE** conformance MUST:

- Publish required metadata fields (§3.2).
- Use decimal-string serialization for tp_value_seconds and vector IO (§3.3, §12).
- Pass all vectors in §12 with stated tolerances.
- Expose metadata via ftu_meta.json or a well-known endpoint (§8.1–§8.2).

FTU-STANDARD

An implementation claiming **FTU-STANDARD** conformance MUST satisfy FTU-CORE, and additionally MUST:

- Publish precision_digits and rounding_policy in metadata (§3.3).
- Publish a stable constants_revision and tp_reference_url (§3.2).
- Provide a deterministic build/version identifier for the FTU module (non-normative if not available).

FTU-SEALED

An implementation claiming **FTU-SEALED** conformance MUST satisfy FTU-STANDARD, and additionally MUST:

- Publish a sealed release manifest (§14) with populated seal fields (§13).
- Publish SHA-256 hashes for the spec artifact, manifest, and test vectors in the manifest (§14).
- Publish a verifier URL and conformance report URL (§13–§14).
- Ensure all “sealed” claims are independently verifiable (no private-only verification).

E.2 Claim format

Implementations SHOULD state claims in the form:

- FTU-CORE@1.7
- FTU-STANDARD@1.7
- FTU-SEALED@1.7

Annex F — Reference Verifier (NON-NORMATIVE)

A small reference verifier is provided for independent checks.

F.1 Files

- ftu_verify_v1.7.1.py (SHA-256: e13c75aabbdc8f5f9326ea0d5a4a339ffe711895729201d6ad17cae0b2d73cd6)
- ftu_test_vectors_v1.7.1.json (SHA-256: c0eab0a07b0b411f98e047e800a84f2e33d6b33aa6f9a18bdc2295d0efb125db)
- ftu_manifest_template_v1.7.1.json (SHA-256: 93f0bbd127faefce0c976707db264c62bee8e8131c379c5e6a51d6c9e45eff7d)

F.2 One-command verification

Example:

```
python3 ftu_verify_v1.7.1.py   --manifest ftu_manifest.json   --spec FTU-SPEC_v1.7_MASTER_UNIVERSAL_PUBLISH_OFFICIAL_FINAL.md   --vectors ftu_test_vectors_v1.7.1.json   --out ftu_conformance_report_v1.7.1.json
```

The verifier:

- hashes artifacts and compares them to the manifest (if present),
- runs the vectors using Decimal arithmetic, and
- emits a JSON conformance report with PASS/FAIL.

15. Change Log and Historic Record (non-normative)

v1.7.1 (FLAGSHIP ABSOLUTE)

- Added Problem/Solution/Why-Now framing and explicit design goals/non-goals for adoption.
- Added GULF Law coherence note: t_f ≡ FTU ≡ t_p.
- Removed in-body placeholder templates; replaced with normative field definitions.
- Clarified hash-sealed release packaging (signature optional).
- **v1.7 (this document)** — Added fully-valid normative test vectors + published vector hash; added conformance profiles (CORE/STANDARD/SEALED); added reference verifier + manifest template hashes.
- **v1.6**: Adds a front-loaded concept/formula primer, a quick navigation index, a glossary/symbol table, an operational sealing checklist, and JSON Schemas for machine validation. Normative FTU definition and conversion rules remain unchanged.
- **v1.4**: Added refresh-rate clarification, strengthened well-known metadata expectations, and clarified angular-frequency normalization semantics.

- **v1.4.1**: "Master Final" packaging update. Adds an explicit before/after historical partition (§0.2), canonical precedence rules (§0.3), and a normative sealed release manifest schema (§14). The normative FTU definition and conversion rules are unchanged.
- **v1.5**: Adds mathematically explicit FTU bridge mapping across established theories (Annex A), reproducible empirical demonstration hooks (Annex B), and a compact bridge index for auditors/search engines (Annex C). The normative FTU definition and conversion rules remain unchanged.

Annex A — Mathematical Mapping Across Established Theories (NON-NORMATIVE, AUDITABLE)

This annex provides a **mathematical re-parameterization** showing how FTU (defined as Planck time) can be used as a common ruler for the

time variable appearing across established theories and engineering practice.

ISOTruth boundary (important):

- The mappings below are **algebraic substitutions** and **dimensional normalizations**.
- They **do not** claim new physics, do not prove time is discrete, and do not assert hard maxima beyond what the referenced theories already state.

A.1 The core substitution (the "FTU bridge" lemma)

Let **t** be time in seconds. Define the dimensionless FTU count:

- $\mathbf{N := t / t_p}$ (so $\mathbf{t = N \cdot t_p = N \cdot FTU}$)

  Then:

- $\mathbf{dt = t_p \cdot dN}$
- $\mathbf{d/dt = (1/t_p) \cdot d/dN}$
- $\mathbf{\partial/\partial t = (1/t_p) \cdot \partial/\partial N}$
- $\mathbf{\partial^2/\partial t^2 = (1/t_p^2) \cdot \partial^2/\partial N^2}$

This is the entire bridge: any equation that uses **t** can be rewritten in terms of the **dimensionless counter N** plus a single constant factor.

A.2 Bridge to Planck units (why FTU is an "anchor")

Because $\mathbf{FTU := t_p}$, standard Planck-unit relations become **directly expressible**:

- Planck length: $\boldsymbol{\ell_p := c \cdot t_p = c \cdot FTU}$
- Planck mass: $\mathbf{m_p := \sqrt{(\hbar c/G)}}$
- Planck energy: $\mathbf{E_p := m_p c^2 = \sqrt{(\hbar c^5/G)}}$
- Planck angular frequency scale: $\boldsymbol{\omega_p := 1/t_p = 1/FTU}$

- Planck cycle frequency scale: $\mathbf{f_p := 1/t_p = 1/FTU}$ (cycle convention)

  Result: FTU ties the time dimension used in SI to the canonical Planck-unit scaffold with a single declared base unit.

A.3 Special relativity (SR) mapping (no theory change)

Time dilation:

- $\mathbf{\Delta t = \gamma \cdot \Delta\tau}$, where $\mathbf{\gamma = 1/\sqrt{(1-v^2/c^2)}}$

  In FTU counts:

- $\mathbf{\Delta N\_t = \Delta t/t_p = \gamma \cdot (\Delta\tau/t_p) = \gamma \cdot \Delta N\_\tau}$

  So SR is unchanged; FTU simply expresses proper and coordinate times in the same **dimensionless** count space.

A.4 General relativity (GR) mapping (no theory change)

Any metric that uses coordinate time **t** (seconds) can be rewritten with $\mathbf{t = N \cdot t_p}$.

  Example: for a line element schematically containing $\mathbf{c^2dt^2}$:

  - $\mathbf{c^2dt^2 = c^2t_p^2\ dN^2}$

  This makes the FTU scaling explicit and aids archival reproducibility of coordinate choices.

A.5 Quantum mechanics (QM) mapping

Schrödinger equation:

- $\mathbf{i\hbar\ \partial\psi/\partial t = H\psi}$

  Substitute $\mathbf{\partial/\partial t = (1/t_p)\ \partial/\partial N}$:

- $\mathbf{i\hbar\ (1/t_p)\ \partial\psi/\partial N = H\psi}$
- or $\mathbf{i(\hbar/t_p)\ \partial\psi/\partial N = H\psi}$

Define a convenient energy scale:

- $\mathbf{E\backslash\_FTU := \hbar/t_p}$ (a constant energy scale proportional to Planck energy)

  Then:

- $\mathbf{i\ E\backslash\_FTU\ \partial\psi/\partial N = H\psi}$

This shows how FTU converts the time derivative into a derivative over a pure counter **N**, with a single constant factor.

A.6 Classical waves / electromagnetism mapping

Wave equation (schematic):

- **$\partial^2u/\partial t^2 = c^2\nabla^2u$**

  Using **$\partial^2/\partial t^2 = (1/t_p^2)\ \partial^2/\partial N^2$**:

- **$(1/t_p^2)\ \partial^2u/\partial N^2 = c^2\nabla^2u$**
- or **$\partial^2u/\partial N^2 = (c^2\ t_p^2)\ \nabla^2u$**

Again: a pure substitution that makes the time scaling explicit.

A.7 Statistical physics / thermodynamics mapping

Boltzmann factor uses energy:

- **exp(−E/(k_B T))**

FTU helps by pairing with **Planck-scale normalized energy $\varepsilon := E/E_p$** (Annex §7.2), which prevents "unit drift" in cross-system archival.

A.8 Engineering / computing mapping (clocks, sampling, refresh)

Any clock frequency **f** has period **T = 1/f**, and FTU-period count:

- **T_FTU = T/t_p = $(1/f)/t_p$**

This is already the normative mapping in §6. The annex adds the "why": it unifies *display*, *audio*, *CPU*, *sensor*, and *bio* clocks into one audit grammar.

Annex B — Empirical Anchors and Demonstrations (NON-NORMATIVE, REPRODUCIBLE)

This annex provides **reproducible demonstrations** (not claims of new physics).

B.1 Primary constant anchor (required for reproducibility)

Use **NIST/CODATA** Planck time as the public reference for *t_p* (see §3.1).

If a different constants revision is used, it MUST be declared (see §3.2).

B.2 Cross-domain example map (computed with t_p = 5.391247e−44 s)

Planck frequency scale (cycle): $f_p = 1/t_p \approx$ **1.854858e43 Hz**

Phenomenon	Input	FTU mapping
1 second	T = 1 s	N_FTU ≈ 1.854858e43
1 millisecond	T = 1e−3 s	N_FTU ≈ 1.854858e40
1 microsecond	T = 1e−6 s	N_FTU ≈ 1.854858e37
60 Hz display	f = 60 Hz	T_FTU ≈ 3.091431e41

120 Hz display	f = 120 Hz	T_FTU ≈ 1.545715e41
24 fps video |	f = 24 Hz	T_FTU ≈ 7.728577e41
48 kHz audio	f = 48000 Hz	T_FTU ≈ 3.864289e38
44.1 kHz audio	f = 44100 Hz	T_FTU ≈ 4.206028e38
1 GHz CPU tick	f = 1e9 Hz	T_FTU ≈ 1.854858e34
Cs-133 atomic clock	f = 9,192,631,770 Hz	T_FTU ≈ 2.017766e33
Red light (illustrative)	f ≈ 4.74e14 Hz	T_FTU ≈ 3.913203e28

*Interpretation: these numbers are **representational** counts (how many FTU ticks per period).*

They are useful for archival comparison and compliance, not as claims of discretization.

B.3 Empirical protocol hooks (recommended add-ons for the sealed release)

A sealed release SHOULD also ship:

- a small ftu-demo/ folder with scripts that:

1. ingest a public time-series dataset (e.g., gravitational-wave strain, EEG, audio),
2. express its sampling and event times as FTU counts,
3. emit a deterministic report whose hash is recorded in the release manifest (§14).

This turns “empirical evidence” into **reproducible audit artifacts** rather than narrative.

Annex C — “Bridge Map” Index (NON-NORMATIVE)

This index is designed for search engines and auditors.

Domain	Canonical time appearance	FTU mapping
SR	Δt, Δτ	$\Delta N = \Delta t/t_p$; ΔN_t = γ·ΔN_τ
GR	dt in metric	$dt = t_p\, dN$; $c^2dt^2 = c^2t_p^2\, dN^2$
QM	∂/∂t	$\partial/\partial t = (1/t_p)\partial/\partial N$
Waves/EM	∂²/∂t²	$\partial^2/\partial t^2 = (1/t_p^2)\partial^2/\partial N^2$
Sampling/clocks |	T=1/f |	T_FTU = $(1/f)/t_p$

Annex D — JSON Schemas (NON-NORMATIVE, MACHINE-VALIDATION)

This annex provides JSON Schema documents that implementers can use to automatically validate ftu_meta.json and ftu_manifest.json.

These schemas are **non-normative**; the normative requirements are in §§3.2, 10, and 14.

D.1 Schema for `ftu_meta.json` (draft 2020-12)

```
{
 "$schema": "https://json-schema.org/draft/2020-12/schema",
 "$id": "https://example.org/ftu/ftu_meta.schema.json",
 "title": "FTU Meta",
 "type": "object",
 "required": ["spec_version", "constants", "dtu"],
 "properties": {
  "spec_version": {"type": "string"},
  "generated_utc": {"type": "string"},
  "constants": {
   "type": "object",
   "required": ["planck_time_seconds"],
   "properties": {
    "planck_time_seconds": {"type": "string"},
    "codata_revision": {"type": ["string", "null"]},
    "uncertainty_seconds": {"type": ["string", "null"]}
   },
   "additionalProperties": true
  },
  "dtu": {
   "type": "object",
   "required": ["name", "p", "q"],
   "properties": {
    "name": {"type": "string"},
    "p": {"type": "integer", "minimum": 0},
    "q": {"type": "integer", "minimum": 1}
   },
   "additionalProperties": true
  }
 },
 "additionalProperties": true
}
```

D.2 Schema for `ftu_manifest.json` (draft 2020-12)

```
{
 "$schema": "https://json-schema.org/draft/2020-12/schema",
 "$id": "https://example.org/ftu/ftu_manifest.schema.json",
 "title": "FTU Sealed Release Manifest",
 "type": "object",
 "required": ["release_id", "spec_version", "artifacts"],
 "properties": {
  "release_id": {"type": "string"},
```

```
  "spec_version": {"type": "string"},
  "artifacts": {
    "type": "array",
    "minItems": 1,
    "items": {
      "type": "object",
      "required": ["path", "sha256"],
      "properties": {
        "path": {"type": "string"},
        "sha256": {"type": "string", "pattern": "^[A-Fa-f0-9]{64}$"},
        "bytes": {"type": ["integer", "null"], "minimum": 0}
      },
      "additionalProperties": true
    }
  },
  "signature": {
    "type": ["object", "null"],
    "properties": {
      "method": {"type": "string"},
      "public_key_ref": {"type": "string"},
      "signature": {"type": "string"}
    },
    "additionalProperties": true
  }
 },
 "additionalProperties": true
}
```


This annex provides an informative template for publishing a **Seal Card** alongside a sealed FTU release. The recommended format is JSON.

`json

{

"release_id": "FTU-SEAL-2025-12-31-v1.7.1",

"manifest_sha256": "",

"spec_artifact_sha256": "",

"verifier_reference": "",

"conformance_report_reference": "",

"signature_method": "none",

"signature_reference": null,

"identity_reference": null,

"published_utc": "2025-12-31 23:59:59"

}

`