Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 May 2027

O. Sigurdsson

1 November 2026

T32: A Compact Universal Timestamp

`draft-sigurdsson-t32-00`

Abstract

This document specifies T32, a text and binary encoding of instants. A T32 stamp writes a moment as 8 characters of Crockford base32: a count of International Atomic Time (TAI) seconds since the start of 2000, offset so that the past encodes too. 2026-10-31 03:13:00 UTC is `G0S7G6B1`.

A stamp's length is its resolution. Dropping characters names a coarser span and appending characters names finer time; every stamp names an interval, never a false instant. Byte-wise sorting is time order, and a prefix is a range query. The canonical 8-character form covers 15,423 BCE to 19,421 CE at one-second resolution.

Note to Readers

RFC Editor: please remove this section before publication.

Source for this draft, its test vectors and two implementations are at `https://github.com/oddurs/t32`. Errata and questions are welcome as issues there, and proposals in its Discussions. This draft is self-contained: its requirements are those of draft 0.3 of the T32 specification [T32], restated here in full, and a difference between the two is a defect in one of them. The motivation, terminology, figures and worked example are informative additions made for this draft. It is prepared for review; whether to submit it to the IETF is a decision for the project's working group, not its editor.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at `https://datatracker.ietf.org/drafts/current/`.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 May 2027.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (`https://trustee.ietf.org/license-info`) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Contents

1. Introduction 5
1.1. Motivation 5
1.2. Goals and Non-Goals 5
1.3. Requirements Language 6
1.4. Terminology 6
1.5. Conformance Classes 6
2. Definitions 7
3. Syntax 8
4. Encoding 8
5. Decoding and Interval Semantics 9
6. Ordering and Prefix Properties 9
7. Length and Resolution 10
8. Durations 10
9. Conversion from UTC 11
9.1. Steps in ΔAT 11
9.2. Unix Time 11
9.3. Before 1961 12
9.4. The Table and the Future 12
9.5. TAI Clocks, Smeared Clocks and Stepped Clocks 12
10. Text Form, Display and Parsing 12
11. Binary Form 13
12. Stability 14
13. Implementation Status 14
14. Security Considerations 14
15. Privacy Considerations 15
16. IANA Considerations 15
17. References 15
17.1. Normative References 15
17.2. Informative References 15
A. TAI − UTC 16
B. Test Vectors 18

B.1. Encoder and Converter (UTC Labels) 18
B.2. Converter Errors 19
B.3. Decoder (Interval Start in TAI Seconds since E) 19
B.4. Decoder with Converter (Interval as UTC Labels, Floored to 1 ns) 19
B.5. Parser 20
B.6. Durations 20
C. Worked Example 20
D. Prior Art 21
E. Known Limits 21
F. Open Issues 22
Author's Address 23

1. Introduction

T32 counts TAI seconds, which are uniform, and writes the count in base32 so that text order is time order. It makes the length of a stamp its resolution: each character added divides the interval it names by 32. A stamp is then both a name for an interval and a sort key whose every prefix is a range. 2026-10-31T03:13:00Z is `G0S7G6B1`; `G0S7` names the 12.1 days that contain it, and `G0S7G6B1G0` the 1/1024 s interval that starts half a second later.

1.1. Motivation

Each format in use today gives up something T32 aims to keep:

- **RFC 3339** [RFC3339], a profile of ISO 8601 [ISO8601], is readable but has several spellings of one instant (`Z` or `+00:00`, any number of fraction digits), so equal instants can differ as text. It sorts as time only within one spelling and one offset, says how precise a value is only through how many digits were written, and takes 20 characters for one second.
- **POSIX seconds** [POSIX] omit leap seconds: one count repeats at each insertion, and a difference across one is a second short. A count has no precision of its own, and its decimal text sorts as time only when zero-padded.
- **TAI64** [TAI64] uses the right time scale and an offset-binary bias, but its text is hexadecimal at a precision fixed by the variant, 17 to 33 characters.
- **ULID** [ULID] and **UUIDv7** [RFC9562] sort by time but are identifiers: a millisecond of POSIX time with random bits appended, at a resolution that cannot vary.

T32 is meant for places where a time is stored, compared or carried as text or as a fixed-width integer: log records and events; database keys and partitions, where a prefix is a range; file names and URLs, which want short, case-insensitive text without punctuation; and data whose times are known only to the day or the hour, where an instant would overstate them.

1.2. Goals and Non-Goals

The goals of T32 are:

1. **Uniform.** From 1961 onward, a T32 second is an SI second of TAI, and subtracting two stamps gives elapsed time. (Before 1961, T32 follows Universal Time; see Section 9.3.)
2. **Compact.** 8 characters at one second, against 20 for [RFC3339] (`2026-10-31T03:13:00Z`).
3. **Sortable.** Byte-wise order of canonical text is time order.
4. **Honest resolution.** A stamp's length states the resolution of the record.
5. **Hierarchical.** Every prefix is a valid, coarser stamp that contains the longer one.
6. **Human-safe.** Case-insensitive, without the most confusable letters, and with fewer accidental words.
7. **Stable.** A stamp always denotes the same interval of the T32 time scale. It never needs to change.

The following are not goals:

- Replacing the civil calendar. People read local dates; T32 sits underneath them.
- Time zones. A stamp names an instant everywhere at once.
- Unique identifiers. T32 carries no randomness. Any random part belongs in a separate field, or after a character outside the alphabet: appended alphabet characters would read as

resolution.

1.3. Requirements Language

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The grammar in Section 3 uses ABNF [RFC5234].

1.4. Terminology

Stamp
: A sequence of 1 to 16 digits from the T32 alphabet, naming an interval of the T32 time scale (Section 5).

Length (n)
: The number of digits in a stamp.

Unit ($u(n)$)
: The width of the interval a stamp of length n names: 32^{8-n} seconds (Section 2).

Value (V)
: A stamp read as an unsigned base-32 integer.

Biased time
: TAI seconds since the epoch E, plus the bias B (Section 2).

Canonical form
: A stamp in uppercase alphabet digits only, with no hyphens (Section 10).

UTC label
: A UTC date and time counted in POSIX seconds (Section 9).

ΔAT
: TAI − UTC as a function of the UTC label (Appendix A).

Inserted and removed time
: Instants of TAI that UTC gives no ordinary label, and labels that name no instant (Section 9.1).

Duration
: A signed whole number of 2^{-40} s (Section 8).

1.5. Conformance Classes

This document defines four conformance classes:

Class	Does	Sections
Encoder	Instant to stamp	4, 7, 10, 11
Decoder	Stamp to interval	5
Parser	Text to stamp or duration	3, 10, 8
Converter	UTC (or Unix time) to and from TAI	9, Appendix A

An implementation conforms to a class when it meets every requirement of that class and reproduces every test vector in Appendix B. The text, those vectors and Appendix A are normative; where a vector and the text disagree, the text decides.

2. Definitions

Five constants define T32. Nothing else is configurable.

Time scale
: TAI (International Atomic Time) from 1961; Universal Time before (see Section 9). TAI has no leap seconds: every second is equal and arithmetic is exact.

Epoch E
: 2000-01-01T00:00:00 TAI, which is 1999-12-31T23:59:28 UTC and Unix 946684768. A round, recent instant that keeps present-day values small. It is not J2000 (2000-01-01T12:00:00 TT).

Bias B
: 2^{39} s = 549,755,813,888 s, about 17,421 years. Offset binary: it centres the range on the epoch, so past instants stay non-negative and sortable.

Alphabet
: `0123456789ABCDEFGHJKMNPQRSTVWXYZ`, Crockford base32 [CROCKFORD]. It omits I, L, O and U, and is in ASCII order, so text order equals numeric order.

Lengths
: 1 to 16 characters; 8 is canonical. Each character carries 5 bits, so 8 characters are 40 bits at 1 s.

A stamp of length n has **unit** $u(n) = 32^{8-n}$ seconds: 1 s at length 8, 32 s at length 7, 1/32 s at length 9.

The **biased time** of an instant t is its TAI seconds since E, plus B. Every representable instant has a biased time in $[0, 2^{40})$ seconds.

TAI itself is computed after the fact by the BIPM; in real time it is approximated as UTC(k) + ΔAT. T32 does not distinguish these realizations.

3. Syntax

```
stamp          = 1*16digit32                  ; canonical form

digit32        = DIGIT / %x41-48 / %x4A-4B / %x4D-4E
                 / %x50-54 / %x56-5A
                 ; 0-9 A-H J K M N P-T V-Z

stamp-input    = in-digit 0*15( [ "-" ] in-digit )

in-digit       = digit32 / lower32 / alias
lower32        = %x61-68 / %x6A-6B / %x6D-6E / %x70-74 / %x76-7A
alias          = %x4F / %x6F / %x49 / %x69 / %x4C / %x6C
                 ; O o read as 0; I i L l read as 1

duration       = sign 1*8digit32 [ "." 1*8digit32 ]
duration-input = sign 1*8in-digit [ "." 1*8in-digit ]
sign           = "+" / "-"

utc-label      = year "-" month "-" day ( "T" / SP )
                 hour ":" minute ":" second [ "." 1*9DIGIT ] "Z"
year           = 4DIGIT / sign 4*6DIGIT
month          = 2DIGIT          ; 01-12
day            = 2DIGIT          ; 01-31, as the month allows
hour           = 2DIGIT          ; 00-23
minute         = 2DIGIT          ; 00-59
second         = 2DIGIT          ; 00-60, 60 only in a leap second
```

Hyphens in `stamp-input` stand only between two digits, singly. Canonical durations have further constraints (Section 8). `utc-label` is [RFC3339] in UTC, with ISO 8601's expanded years for those outside 0000 to 9999; as in all ABNF, `"T"` and `"Z"` also match lowercase. Converters read and write UTC labels in this form, with the fewest fraction digits that are exact.

4. Encoding

An encoder writes instant t at length n as the floor of its biased time divided by the unit, in n digits:

$$V = \left\lfloor \frac{t - E + B}{32^{8-n}} \right\rfloor, \qquad 0 \le V < 32^n$$

1. Obtain $t - E$ in TAI seconds, exactly (Section 9).
2. Add B, divide by $u(n)$, and floor. Encoders MUST floor exactly once, at this step, toward negative infinity; intermediate values MUST NOT be rounded. Rounding would let a stamp name an interval that does not contain t.
3. If V is outside $[0, 32^n)$, the encoder MUST fail. It MUST NOT wrap or clamp.
4. Encoders MUST write V as exactly n digits from the alphabet, most significant first, uppercase.

Because there is one floor, encoding at length n and truncating to length k gives the stamp that encoding at length k gives. Reducing resolution is a string operation.

5. Decoding and Interval Semantics

A stamp of length n and value V denotes the half-open interval of the T32 time scale (TAI from 1961):

$$\left[\, E - B + V \cdot 32^{8-n},\;\; E - B + (V+1) \cdot 32^{8-n} \,\right)$$

- **Start** is the earliest instant the stamp can mean. Sorting and range queries use it.
- **Midpoint** is the best single estimate when one instant is required.
- **Containment.** Stamp A contains stamp B exactly when A is a prefix of B.
- **No partial overlap.** Two stamps are nested or disjoint; there is no third case.
- **Trailing zeros carry meaning.** `G0S7G6B1` and `G0S7G6B10` are different stamps; the second is one thirty-second of the first.
- **Nanosecond readings.** A decoder that writes an instant as a nanosecond UTC label MUST write the latest nanosecond label whose length-16 stamp is no later than the instant's length-16 stamp; within a leap second, labels 23:59:60.f count as labels. Every nanosecond label then survives a round trip exactly. A label's instant need not be a multiple of 2^{-40} s, so the instant encoded from a label is its floor, which can fall in time inserted just before that label (Section 9.1); it still reads back as the label.

Each digit selects one of 32 equal parts of the interval named by the digits before it. Figure 1 shows the fifth digit of `G0S7G6B1` selecting part `G`, the 17th, of `G0S7`.

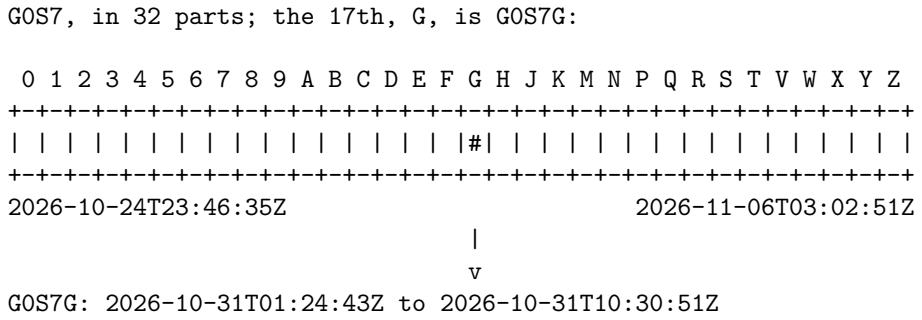


Figure 1: G0S7G is the 17th of the 32 parts of G0S7

6. Ordering and Prefix Properties

Byte-wise comparison of canonical stamps orders them by start, and on equal starts puts the shorter (containing) stamp first.

- **Same length:** text order is time order.
- **Mixed lengths:** still ordered by start. `G0S7` sorts immediately before everything it contains.
- **Collation:** comparisons MUST be byte-wise (ASCII, or SQL `COLLATE "C"`). Locale-aware collation can break the order.
- **Range query:** everything inside `G0S7`, at any length, is `t >= 'G0S7' AND t < 'G0S8'`. The upper bound is the prefix with its last digit incremented, carrying left and dropping every digit that carries to zero (`G0SZ` becomes `G0T`, `G0ZZ` becomes `G1`); keeping them would admit the shorter stamp `G0T`, which `G0SZ` does not contain. A prefix of all `Z` has no upper bound.

7. Length and Resolution

Length records the resolution of the record: the width of the interval in which the instant is stated to lie. It does not describe the accuracy of the clock that produced it.

Length	Unit (exact)	Unit (approx.)	Example	Typical source
1	2^{35} s	1,089 years	`G`	
2	2^{30} s	34.0 years	`G0`	
3	2^{25} s	388 days	`G0S`	
4	2^{20} s	12.1 days	`G0S7`	
5	2^{15} s	9.10 hours	`G0S7G`	
6	2^{10} s	17.1 minutes	`G0S7G6`	
7	2^{5} s	32 seconds	`G0S7G6B`	
8	**1 s**	**1 second**	`G0S7G6B1`	**Seconds (canonical)**
10	2^{-10} s	0.977 ms		Milliseconds
12	2^{-20} s	0.954 µs		Microseconds
14	2^{-30} s	0.931 ns		Nanoseconds
16	2^{-40} s	0.909 ps		Picoseconds

Odd lengths are valid. A source's resolution r is the unit in which it reports time, such as 1 ns for a POSIX `timespec`, not the accuracy of the clock behind it. An encoder MUST NOT emit a stamp longer than the longest length n with $u(n) > r/\sqrt{32}$: the length whose unit is nearest r on a logarithmic scale, the shorter where two are equally near. In integers, n is the longest length with $32 \cdot u(n)^2 > r^2$. A millisecond clock therefore yields at most 10 characters, a nanosecond clock at most 14. Encoders SHOULD use one fixed length per field.

8. Durations

A duration is a signed whole number of 2^{-40} s, written exactly as a sign, base32 seconds, and optional base32 fraction digits: `+2MC0` is one day.

Duration	Seconds	T32 duration
1 minute	60	`+1W`
1 hour	3,600	`+3GG`
1 day	86,400	`+2MC0`
1 week	604,800	`+JEM0`
Half a second	0.5	`+0.G`

- **Value.** sign × (I + F / 32^k) seconds, where I is the integer part and F the k fraction digits. Every duration is exact in this form, and its magnitude is less than 2^{40} s.
- **Canonical form.** The sign is REQUIRED. No leading zeros in the integer part (zero is `0`), no trailing zeros in the fraction, no fraction if it is zero. Zero is `+0`, never `-0`.
- **From other units.** Converting a duration from decimal units (nanoseconds, say) MUST truncate toward zero, so negating a duration negates its text.

- **Stamp minus stamp.** B − A is start(B) − start(A), exactly. The difference of the instants that B and A stand for lies within $(-u(A), +u(B))$ of it.
- **Range.** Implementations MUST reject a duration whose magnitude reaches 2^{40} s.
- **Stamp plus duration.** Add the duration to the stamp's start, then floor to the stamp's own length. If the result leaves $[0, 2^{40})$ s, the operation MUST fail.
- Durations are SI seconds. `+2MC0` is 86,400 s even across a leap second; a civil day is a calendar notion.

9. Conversion from UTC

T32 is TAI-native. UTC converts through ΔAT = TAI − UTC, given in Appendix A as a function of the UTC label. ΔAT has been 37 s since 2017-01-01.

A **UTC label** L is a time written in the proleptic Gregorian calendar with astronomical year numbering (year 0 is 1 BCE), counted in seconds since 1970-01-01T00:00:00 by the POSIX formula [POSIX]: days × 86,400 + seconds of the day. A Unix timestamp is exactly such a label. Then:

$$t - E = L + \Delta\mathrm{AT}(L) - 946684800 \text{ s}$$

Converters MUST evaluate ΔAT exactly (every value in Appendix A is an exact decimal) and MUST NOT round before the encoder's single floor.

9.1. Steps in ΔAT

At each boundary b in Appendix A, ΔAT changes by $s = \Delta\mathrm{AT}_{\mathrm{after}}(b) - \Delta\mathrm{AT}_{\mathrm{before}}(b)$. Appendix A lists every non-zero step.

- **Removed time ($s < 0$).** A label before b whose TAI instant, under the old formula, would reach the TAI instant of b under the new one never occurred. Converters MUST reject it. This happened on 1961-08-01 and 1968-02-01, removing just under 0.05 s and 0.1 s of labels (from 23:59:59.950000001 and 23:59:59.900000003 respectively, at nanosecond resolution). Defined this way, every instant of TAI still has a label. A negative leap second would remove 23:59:59.
- **Inserted time ($s > 0$).** For s seconds of TAI before the instant labelled b, the formula gives no label.
  - From 1972-07-01 each insertion is one leap second, written 23:59:60 on the day before b. Converters MUST accept second 60 then and only then, and convert 23:59:60.f to the TAI instant one second after that of 23:59:59.f. (The insertion of 0.107758 s at 1972-01-01 is fractional and has no second 60.)
  - Decoders MUST write an instant in a leap second as 23:59:60.f, and any other inserted instant as the label b. They MUST report that the instant was inserted.
  - Before 1972, UTC clocks repeated fractions of a second at these steps; T32 does not represent the repetition.

9.2. Unix Time

A Unix value is a UTC label and never names 23:59:60. The Unix value at a leap-second boundary therefore converts to 00:00:00 of the following day. By the decoder rule, an instant inside a leap second converts back to that same value.

9.3. Before 1961

Before 1961-01-01, ΔAT is defined as 0 and the label is read as Universal Time (mean solar time at Greenwich), extended proleptically. Labels are Gregorian: a date in the Julian calendar is converted to Gregorian before it is a label.

- Pre-1961 stamps are **stable**: they do not change when models of Earth's rotation are revised.
- Pre-1961 T32 time follows Earth's rotation, not atomic seconds. Elapsed time between such stamps differs from physical time by the change in ΔT (TT − UT): under about a minute across the last three centuries, hours in antiquity. Applications that need physical elapsed time in the deep past apply a ΔT model when they read a stamp, never when they write one.
- The definition changes at 1961-01-01, where ΔAT steps from 0 to 1.4228180 s; the step is inserted time.

9.4. The Table and the Future

In Resolution 5 of October 2026, the 28th CGPM decided that UTC becomes continuous on 2027-05-20, allowing |UT1 − UTC| to reach 3,600 s [CGPM2026]. IERS Bulletin C 72 announced no leap second for 2026-12-31 [IERS-C]. No further step in ΔAT is therefore expected: Appendix A is expected to be final, with ΔAT = 37 s for every label from 2017-01-01.

- A table carries an expiry date: Appendix A expires at 2027-06-28T00:00:00Z, as does the `leap-seconds.list` it reproduces. Converting a label at or after the expiry of the table in use, in either direction, is **provisional**. A converter MUST return that status with the result, and MUST NOT fail a conversion for it alone. The status belongs to the conversion, not to the stamp.
- Converters SHOULD be able to report which table they used, identified by its expiry date.
- If UTC is ever stepped again, Appendix A gains rows in a new version of this document. Stamps already issued keep their meaning; only conversions of later labels change.

9.5. TAI Clocks, Smeared Clocks and Stepped Clocks

- A converter MAY read TAI directly, for example Linux `CLOCK_TAI`, where $t - E =$ `CLOCK_TAI` − 946684800 s. It MUST first confirm that the kernel's TAI offset equals ΔAT from its own table (on Linux, `adjtimex()` reports it as `tai`, which must be 37). An unset offset is 0, and a stale one is wrong by a whole second or more.
- Around a leap second, some clocks smear it across many hours and others repeat a second. Readings from those windows can be up to 1 s from the label they claim. An encoder SHOULD undo a known smear before encoding. Otherwise, stamps from the window are not guaranteed to contain the true instant, and SHOULD be marked as such by the application.

10. Text Form, Display and Parsing

Parse leniently, emit strictly.

- **Canonical form.** Encoders MUST emit `stamp`: uppercase alphabet digits only. Only the canonical form may be stored, compared, hashed or indexed.

- **Display form.** For people, a stamp MAY be shown in groups of four joined by hyphens: `G0S7-G6B1`, `G0S7-G6B1-3XYE`. The display form MUST NOT be stored.
- **Parsing.** Parsers MUST accept exactly `stamp-input` and MUST produce the canonical form.
  - Input is ASCII. Any other character, including whitespace and non-ASCII letters, MUST be rejected. Callers that accept surrounding whitespace strip it before parsing.
  - Letters are case-insensitive. `O` is read as `0`, and `I` or `L` as `1`. `U` MUST be rejected.
  - A hyphen MAY appear only between two digits. Leading, trailing and doubled hyphens MUST be rejected, so a negative duration such as `-1W` is never read as a stamp.
  - More than 16 digits MUST be rejected.
  - A parser reads its input from left to right and reports the first error it meets: at a character, the error that character causes; at the end of the input, a missing sign, a missing part (`empty`) or a trailing hyphen. Converters check a UTC label in this order: `invalid_date`, `not_a_leap_second`, `removed_time`, `out_of_range`. The error codes are listed in Appendix B.
- **No check character.** Length carries meaning, so a trailing check digit would be indistinguishable from one more digit of resolution. Integrity belongs to the transport.

11. Binary Form

In binary, a stamp is its value V as an unsigned integer. The length is not stored: a binary field declares it, as type **T32-***n*, and every value in the field has that length.

Field type	Bits	Packed bytes	Fits in	Resolution	Typical source
T32-8	40	5	64-bit integer	1 s	Seconds
T32-10	50	7	64-bit integer	0.977 ms	Milliseconds
T32-12	60	8	64-bit integer	0.954 µs	Microseconds
T32-14	70	9	128-bit integer	0.931 ns	Nanoseconds
T32-16	80	10	128-bit integer	0.909 ps	Picoseconds

- Packed, V is written big-endian in the fewest whole bytes, and byte order is time order, like the text. Lengths 8 and 16 pack with no spare bits.
- As an integer, V sorts in time order within the field.
- A field SHOULD declare the length that matches its source, by the rule in Section 7. Lengths up to 12 fit a signed 64-bit integer, so T32-10 suits millisecond sources and T32-12 microsecond sources.

Figure 2 shows `G0S7G6B1` as digits, bits and a packed T32-8 field. At length 10, `G0S7G6B1G0` is 50 bits; packed as T32-10 in 7 bytes, its first 6 bits are zero: `0x02 0x00 0xC9 0xE0 0x65 0x86 0x00`.

```
Text          G     0     S     7     G     6     B     1
Digit        16     0    25     7    16     6    11     1
Bits      10000 00000 11001 00111 10000 00110 01011 00001
T32-8     10000000 00110010 01111000 00011001 01100001
          0x80     0x32     0x78     0x19     0x61
```

Figure 2: G0S7G6B1 as text, digit values, bits and packed bytes

12. Stability

This document restates draft 0.3 of the T32 specification [T32]. Until version 1.0, any part may change. From 1.0, Definitions (Section 2), Encoding (Section 4), Decoding (Section 5), the grammar (Section 3) and the reading of labels before 1961 (Section 9.3) are frozen; later versions may only add, for example rows in Appendix A. A stamp carries no version, and its meaning never changes.

13. Implementation Status

RFC Editor: please remove this section before publication.

This section records the status of known implementations at the time of posting of this Internet-Draft, following [RFC7942]. The description of implementations here is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Listing an implementation here does not imply endorsement by the IETF. The current list is maintained at `https://github.com/oddurs/t32/blob/main/IMPLEMENTATIONS.md`.

`t32` (Rust)
: The reference implementation, published as the `t32` crate. All four classes; passes every vector from version 0.3.0. Integer arithmetic only, no required dependencies. Maintained by the author.

`t32.ts` (TypeScript)
: The same algorithm in one file, using BigInt. All four classes; passes every vector. Maintained by the author.

`oracle.py` (Python)
: An independent check in exact rational arithmetic, written from the text and sharing no code with the implementations. It generates the test vectors. Encoder, decoder, converter and parser.

Both libraries hold biased time as fixed point in units of 2^{-40} s, and hold TAI − UTC as exact multiples of $1/(10^7 \times 86{,}400 \times 10^9)$ s, the finest step any coefficient in Appendix A needs. Converting a label is then exact, and inverting it is a single exact division per period.

No implementation independent of the author exists yet.

14. Security Considerations

- **Canonicalize first.** Different input texts can denote the same stamp (`g0s7-g6bi`, `G0S7G6B1`). Compare, hash and use as keys only canonical forms.
- **Bounded input.** A valid input is at most 31 characters (16 digits and 15 hyphens), and a duration at most 18. A parser that reports the first error it meets has met one by the 32nd character, so it never reads further.
- **Confusables.** Parsing is ASCII-only, so look-alike Unicode letters are rejected rather than mapped.
- **Table integrity.** A tampered leap-second table shifts every conversion after the change. Obtain it from tzdata or the IERS, and check its expiry date and hash (`leap-seconds.list` carries one).
- **Range.** Out-of-range instants and overflowing arithmetic MUST fail. They MUST NOT

wrap.

- **No authenticity.** A stamp is a claim about time, as trustworthy as the clock that produced it and the channel that carried it. T32 provides no integrity or authenticity. Protocols that use time in security decisions, such as expiry or the order of authorization events, need to protect a stamp as they would any other field.

15. Privacy Considerations

Resolution is information. A long stamp can reveal timing, such as when an action happened to the microsecond. Truncate a stamp to the resolution you mean to publish; for T32 that is a string slice.

16. IANA Considerations

This document has no IANA actions. Registrations that could follow, such as a suffix tag under [RFC9557] or a URN namespace, are listed as an open issue in Appendix F.

17. References

17.1. Normative References

[IERS-C]
IERS, "Bulletin C" (announcements of leap seconds), and `leap-seconds.list` as distributed with the IANA Time Zone Database, `https://www.iers.org/IERS/EN/Publications/Bulletins/bulletins.html`.

[POSIX]
IEEE, "IEEE Standard for Information Technology—Portable Operating System Interface (POSIX) Base Specifications, Issue 8", IEEE Std 1003.1-2024, Base Definitions, "Seconds Since the Epoch".

[RFC2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

[RFC5234]
Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008.

[RFC8174]
Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.

[TAI-UTC]
U.S. Naval Observatory, `tai-utc.dat`: TAI − UTC from 1961, `https://maia.usno.navy.mil/ser7/tai-utc.dat`, as also embodied in the IAU SOFA routine `iauDat` (`https://www.iausofa.org/`).

17.2. Informative References

[CGPM2026]
28th General Conference on Weights and Measures (CGPM), Resolution 5, "On the tech-

nical actions needed to ensure the continuity of UTC", October 2026, https://www.bipm.org/en/cgpm-2026/resolution-5.

[CROCKFORD]
Crockford, D., "Base 32", https://www.crockford.com/base32.html.

[ISO8601]
International Organization for Standardization, "Date and time — Representations for information interchange — Part 1: Basic rules", ISO 8601-1:2019.

[RFC3339]
Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002.

[RFC7942]
Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016.

[RFC9557]
Sharma, U. and C. Bormann, "Date and Time on the Internet: Timestamps with Additional Information", RFC 9557, DOI 10.17487/RFC9557, April 2024.

[RFC9562]
Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024.

[T32]
Sigurdsson, O., "T32: A Compact Universal Timestamp", draft 0.3, 1 November 2026, tag spec-draft-0.3, https://github.com/oddurs/t32/blob/spec-draft-0.3/SPEC.md.

[TAI64]
Bernstein, D. J., "TAI64, TAI64N, and TAI64NA", https://cr.yp.to/libtai/tai64.html.

[ULID]
"Universally Unique Lexicographically Sortable Identifier", https://github.com/ulid/spec.

A. TAI − UTC

This appendix is normative. $\Delta AT = TAI - UTC$ as a function of the UTC label, from [IERS-C] and [TAI-UTC]; it expires at 2027-06-28T00:00:00Z. It has been 37 s since 2017-01-01, and with UTC continuous from 2027-05-20 it is expected to stay 37 s (Section 9.4). Before 1961-01-01, ΔAT is defined as 0 for T32.

1972 onward (whole seconds; each value applies from 00:00:00 UTC on its date):

From	ΔAT (s)	From	ΔAT (s)
1972-01-01	10	1985-07-01	23
1972-07-01	11	1988-01-01	24
1973-01-01	12	1990-01-01	25
1974-01-01	13	1991-01-01	26
1975-01-01	14	1992-07-01	27
1976-01-01	15	1993-07-01	28
1977-01-01	16	1994-07-01	29
1978-01-01	17	1996-01-01	30
1979-01-01	18	1997-07-01	31
1980-01-01	19	1999-01-01	32
1981-07-01	20	2006-01-01	33
1982-07-01	21	2009-01-01	34
1983-07-01	22	2012-07-01	35
		2015-07-01	36
		2017-01-01	37

1961 to 1971 ($\Delta\text{AT} = a + (\text{MJD} - m) \times r$ seconds, where MJD is the Modified Julian Date of the UTC label, fraction of the day included):

From	a (s)	m (MJD)	r (s/day)
1961-01-01	1.4228180	37300	0.0012960
1961-08-01	1.3728180	37300	0.0012960
1962-01-01	1.8458580	37665	0.0011232
1963-11-01	1.9458580	37665	0.0011232
1964-01-01	3.2401300	38761	0.0012960
1964-04-01	3.3401300	38761	0.0012960
1964-09-01	3.4401300	38761	0.0012960
1965-01-01	3.5401300	38761	0.0012960
1965-03-01	3.6401300	38761	0.0012960
1965-07-01	3.7401300	38761	0.0012960
1965-09-01	3.8401300	38761	0.0012960
1966-01-01	4.3131700	39126	0.0025920
1968-02-01	4.2131700	39126	0.0025920

Steps (from these tables):

Boundary	Step (s)	Kind
1961-01-01	+1.4228180	Inserted (start of atomic definition)
1961-08-01	−0.05	Removed
1963-11-01, 1964-04-01, 1964-09-01, 1965-01-01, 1965-03-01, 1965-07-01, 1965-09-01	+0.1 each	Inserted
1968-02-01	−0.1	Removed
1972-01-01	+0.107758	Inserted
Each date from 1972-07-01 in the first table	+1	Inserted (leap second, 23:59:60)

The boundaries 1962-01-01, 1964-01-01 and 1966-01-01 change the formula but not its value.

B. Test Vectors

This appendix is normative: a conforming implementation reproduces every vector in it for its class. A larger set of 782 vectors, including both sides of every boundary in Appendix A, is published with [T32] as `vectors/t32-vectors.json` at the tag `spec-draft-0.3`. It is informative here, because an Internet-Draft carries its normative content itself; an implementation that passes it passes the vectors below.

Every vector was computed by an exact rational-arithmetic oracle written directly from the text (Section 13), errors included, and is reproduced by both libraries. A disagreement between this text, the vectors and an implementation is a defect.

Errors in the vectors file are named by code:

Code	Condition
`empty`	No digit where one is required: the whole stamp, or either part of a duration
`too_long`	More than 16 digits in a stamp, or 8 in either part of a duration
`invalid_char`	A character outside the input grammar
`misplaced_hyphen`	A hyphen not between two digits
`missing_sign`	A duration that does not begin with `+` or `-`
`invalid_date`	Text that is not a `utc-label`, or a day or time the calendar lacks
`not_a_leap_second`	Second 60 where no leap second was inserted
`removed_time`	A UTC label in removed time
`out_of_range`	An instant, stamp or duration outside its range

B.1. Encoder and Converter (UTC Labels)

Case	UTC	Length	Stamp
T32 epoch	1999-12-31T23:59:28	8	`G0000000`
Unix epoch (ΔAT = 8.000082 s)	1970-01-01T00:00:00	8	`FZ3S5F48`
Apollo 11 landing	1969-07-20T20:17:40	8	`FZ3BMMKB`
Rubber era, exact (ns clock)	1966-06-15T12:00:00.123456789	14	`FZ0EDKA4VPC2A1`

Case	UTC	Length	Stamp
UTC becomes integer	1972-01-01T00:00:00	8	FZ5NA8WA
Before last leap second	2016-12-31T23:59:59	8	G0FZP0S3
Last leap second	2016-12-31T23:59:60	8	G0FZP0S4
After last leap second	2017-01-01T00:00:00	8	G0FZP0S5
Unix 1483228800	(2017-01-01T00:00:00)	8	G0FZP0S5
Spec date	2026-10-31T03:13:00	8	G0S7G6B1
Spec date + 0.5 s (ms clock)	2026-10-31T03:13:00.500	10	G0S7G6B1G0
Spec date + 0.123 s (µs clock)	2026-10-31T03:13:00.123000	12	G0S7G6B13XYE
Before 1961, taken as UT	1675-08-20T00:00:00	8	FPEY6N00

B.2. Converter Errors

UTC	Result
1961-07-31T23:59:59.95	Accepted
1961-07-31T23:59:59.950000001	Error: removed time
1968-01-31T23:59:59.900000002	Accepted
1968-01-31T23:59:59.900000003	Error: removed time
2016-12-30T23:59:60	Error: not a leap second

B.3. Decoder (Interval Start in TAI Seconds since E)

Stamp	Start (s)	Width (s)
00000000	−549,755,813,888	1
FZ3BMMKB	−960,867,733	1
G0000000	0	1
G0FZP0S4	536,544,036	1
G0S7	846,200,832	1,048,576

B.4. Decoder with Converter (Interval as UTC Labels, Floored to 1 ns)

Stamp	Start	End (exclusive)
00000000	−15422-12-06T11:41:52	−15422-12-06T11:41:53
FZ3BMMKB	1969-07-20T20:17:39.425406217	1969-07-20T20:17:40.425406187
G0FZP0S4	2016-12-31T23:59:60 (inserted; label 2017-01-01T00:00:00)	2017-01-01T00:00:00
G0S7	2026-10-24T23:46:35	2026-11-06T03:02:51
G0S7G6	2026-10-31T03:07:07	2026-10-31T03:24:11

The FZ3BMMKB interval is 0.99999997 UTC seconds wide, because UTC seconds were stretched in 1969. Years are astronomical (−15422 is 15423 BCE).

B.5. Parser

Input	Result
`g0s7-g6bi`	`G0S7G6B1`
`oooo`	`0000`
`G0S7G6B1-3XYE`	`G0S7G6B13XYE`
`G0SU`	Error: invalid character
␣`G0S7` (leading space)	Error: invalid character
`ı` (U+0131)	Error: invalid character
`-1W`, `1W-`, `G0S7--G6B1`	Error: misplaced hyphen
17 digits	Error: too long
empty	Error: empty

B.6. Durations

Input	Value	Canonical
86,400 s	one day	`+2MC0`
−1.5 s		`-1.G`
−1 ns (truncated toward zero)	$-1{,}099 \times 2^{-40}$ s	`-0.0000012B`
0		`+0`
`G0S8` start − `G0S7` start		`+10000`
`G0S7G6B1` start − `G0S7` start		`+G6B1`
`1W` (no sign)		Error

C. Worked Example

This appendix is informative. It follows one instant through encoding, truncation, decoding, a range query and a duration.

Encoding 2026-10-31T03:13:00Z at length 8 (Section 4):

1. The UTC label L is 1793416380 POSIX seconds.
2. ΔAT(L) is 37 s (Appendix A, from 2017-01-01).
3. $t - E = L + \Delta\mathrm{AT}(L) - 946684800 = 846{,}731{,}617$ s.
4. Adding B = 549,755,813,888 s gives a biased time of 550,602,545,505 s.
5. The unit at length 8 is 1 s, so V = 550,602,545,505.
6. In base 32, V has the digits 16, 0, 25, 7, 16, 6, 11, 1, written `G0S7G6B1` (Figure 2).

A finer stamp. From a millisecond clock reading 03:13:00.500, the biased time is 550,602,545,505.5 s. At length 10 the unit is 2^{-10} s, so

$$V = \lfloor 550{,}602{,}545{,}505.5 \times 1024 \rfloor = 563{,}817{,}006{,}597{,}632,$$

written `G0S7G6B1G0`. Its first 8 digits are the length-8 stamp, as the single floor guarantees.

Truncation and decoding. Each prefix of `G0S7G6B1` is a coarser stamp that contains it:

Stamp	Starts (UTC)	Ends, exclusive (UTC)	Width
`G0S7`	2026-10-24T23:46:35	2026-11-06T03:02:51	12.1 days
`G0S7G`	2026-10-31T01:24:43	2026-10-31T10:30:51	9.10 hours
`G0S7G6`	2026-10-31T03:07:07	2026-10-31T03:24:11	17.1 minutes
`G0S7G6B`	2026-10-31T03:12:59	2026-10-31T03:13:31	32 seconds
`G0S7G6B1`	2026-10-31T03:13:00	2026-10-31T03:13:01	1 second

For `G0S7`, $V = 16 \times 32^3 + 0 \times 32^2 + 25 \times 32 + 7 = 525{,}095$ and the unit is $2^{20} = 1{,}048{,}576$ s, so the interval starts at $525{,}095 \times 1{,}048{,}576 - B = 846{,}200{,}832$ s after E (Section 5). Subtracting ΔAT and adding 946684800 gives the UTC label 1792885595, which is 2026-10-24T23:46:35Z.

Range query. Every stamp inside `G0S7`, at any length, satisfies `t >= 'G0S7' AND t < 'G0S8'` under byte-wise comparison (Section 6).

Duration. The start of `G0S7G6B1` minus the start of `G0S7` is 530,785 s exactly, written `+G6B1` (Section 8).

D. Prior Art

This appendix is informative.

The example column writes 2026-10-31T03:13:00Z in each format.

Format	Example	Chars	Time scale	Text sorts as time	Resolution in the value
RFC 3339, ISO 8601	`2026-10-31T03:13:00Z`	20	UTC	Within one profile and offset	Partly
Unix time	`1793416380`	10	UTC, POSIX	When zero-padded	No
TAI64 label	`@400000006ae55ce1`	17	TAI	Yes	Fixed per variant
ULID, time part	`01M67Z4FK0`	10	Unix, ms	Yes	No
UUIDv7	48-bit Unix ms	36	Unix, ms	Yes	No
T32	`G0S7G6B1`	**8**	**TAI**	**Yes**	**Yes, by length**

TAI64 [TAI64] is the closest relative: the same timescale and the same offset-binary idea (its bias is 2^{62}), written in hexadecimal at a precision fixed by the variant (TAI64, TAI64N, TAI64NA). ULID [ULID] showed that sortable Crockford base32 works for time-ordered keys, and UUIDv7 [RFC9562] brought a time-ordered layout into a standard identifier. T32 adds three things: length as resolution, with interval semantics and prefix containment; the shortest case-insensitive text that sorts as time; and an exact UTC mapping at every step of the table.

E. Known Limits

This appendix is informative. These limits are deliberate; removing any of them would cost a goal.

- **Calendar alignment.** Buckets do not fall on days, months or years. Eras are aligned to the epoch: `F` covers about 911 to 2000 and `G` about 2000 to 3089, so dates either side of 2000 differ in the first character.
- **Relativity.** TAI, like Terrestrial Time (TT = TAI + 32.184 s), is a time scale for clocks on Earth. Clocks in orbit, on the Moon or on Mars convert through a defined relativistic transformation, as GPS does, before encoding.
- **Before 1961.** Stamps follow Universal Time, not atomic seconds (Section 9.3).
- **Range.** The canonical form ends in 19,421 CE. Extending it would add a character to every stamp in use.
- **Uniqueness.** Two events in the same unit share a stamp. T32 is a time, not an identifier; see Section 1 for where randomness may go.

F. Open Issues

RFC Editor: please remove this appendix before publication.

These are the questions the author considers open. Each states the current position and the kind of evidence that would change it.

The name
: "T32" is also the name of Arm's 32-bit Thumb instruction set and the common name of Lauterbach's TRACE32 debugger. The current position is to keep the name, since the contexts rarely meet and the name says what the format is: T for time, 32 for its radix. Evidence of real confusion, or a better name, would reopen it.

The epoch
: T32 counts from 2000-01-01T00:00:00 TAI. Alternatives considered were J2000 (2000-01-01T12:00:00 TT), which suits astronomy, and 1972-01-01, when UTC's offset from TAI became whole seconds. The current position favours a round instant in the scale T32 counts. Since the epoch fixes every stamp, this question closes before publication.

Before 1961
: Labels before 1961 are read as Universal Time, which keeps stamps stable but makes elapsed time non-physical across centuries. The alternative is to apply a ΔT model, which is physical but changes stamps whenever the model is revised. The current position favours stability. The reading of these labels is frozen at version 1.0, so this question closes then.

The resolution rule
: An encoder may use the length whose unit is nearest its source's resolution on a logarithmic scale, so a stamp can claim up to $\sqrt{32}$ (about 5.7) times more resolution than its source has. The stricter alternative is the longest length whose unit is no finer than the source, which gives 9 for a millisecond clock instead of 10. The current position prefers the familiar lengths 8, 10, 12 and 14.

Letter case
: Canonical stamps are uppercase. Lowercase reads better in URLs and file names, and some systems fold case. Since parsing is case-insensitive, the question is only which form is canonical. The current position is uppercase, as in Crockford base32.

A human rendering
: This document defines a display form for stamps, but not how to show a stamp's interval to people in their own calendar and time zone. The question is whether that belongs here or in a companion document.

Identifiers and registration
: T32 defines no URI scheme, media type or registered tag. Candidates include a suffix tag under [RFC9557] and a URN namespace. Registration would follow only if implementers ask

for it.

Other time scales

TAI suits clocks on Earth. Lunar and Martian time scales are being defined, and a T32 profile for one of them would need its own epoch and conversion. The current position is to leave the format TAI-only and to define profiles, if any, in separate documents.

Author's Address

Oddur Sigurdsson
Email: oddurs@gmail.com