Common Calendar Local Timescales

Local Time Timestamp System

Brooks Harris Version 9 2024-05-21

Table of Contents

Notation

"YMDhms" is shorthand for year-month-day hour:minute:second representation.

ISO 8601 representation is supplemented with suffixes (UTC) and (TAI), for example 1970-01-01 00:00:10 (TAI) = 1970-01-01T00:00:00 (UTC).

"UTC1970" is shorthand for 1970-01-01 00:00:10 (TAI) = 1970-01-01T00:00:00 (UTC).

"UTC1972" is shorthand for 1972-01-01 00:00:10 (TAI) = 1972-01-01T00:00:00 (UTC).

1 Introduction

Common Calendar (CCT) has been developed to resolve the ambiguities and inconsistencies in current timekeeping specifications to realize a practical engineering solution for accurate local timekeeping.

Common Calendar takes the form of defining a set of local timescales together with technical specifications for interoperable timestamps and associated calculation formulas. By carefully defining the environment and parameters it is possible to create a self-contained environment that is both reverse compatible to existing timekeeping technologies and provides a set of deterministic local timescales.

Definitions of timescales are fundamental to any timekeeping system. This paper describes the timescales used by CCT.

2 References

Common Calendar Date and Time Terms and Definitions

Common Calendar TAI-UTC API

Common Calendar YMDhms API

Common Calendar Time Zone Database API

Common Calendar Binary Format

Common Calendar Character Format

Common Calendar Timestamp API

Common Calendar Reference Implementation

Recommendation ITU-R TF.460-6 (02/02), Standard-Frequency and Time-Signal Emissions
http://www.itu.int/rec/R-REC-TF.460/en

Theory and pragmatics of the tz code and data
https://www.ietf.org/timezones/code/theory.html

IETF Rfc 6557 Procedures for Maintaining the Time Zone Database
http://tools.ietf.org/html/rfc6557

3 Common Calendar Local Timescales

Any timescale must have an origin. CCT uses a fixed origin of UTC1970, as described below.

Any timescale must select a time measurement unit. CCT supports many second and decimal fractions of second resolutions, from one-second to picoseconds.

There are two primary sources of metadata required to create a CCT timestamp: leap-seconds and local time parameters CCT obtains leap-seconds from the IERS as described in Common Calendar TAI-UTC API. Local time parameters are gathered from IANA Time Zone Database (TzDb) as described in Common Calendar Time Zone API.

IERS leap-seconds are typically introduced on local timescales simultaneous with UTC leap-seconds. An alternate method inserts leap-seconds at the end of the local day. This is sometimes called "rolling leap-seconds". CCT supports both schemes, as described below,

TzDb describes the many variations of time zone UTC-offsets and Daylight Saving Time (DST) changes, called "transitions", used across the world. CCT gathers this information to preserve accurate refection of local time as provided by TzDb. The many rule sets are logically complex and intricate and the CCT reference implementation uses adapted versions of the c-code supplied by TzDb, zic.c and zdump.c to replicate the behavior of TzDb and other systems that rely on it.

TzDb represents the common methods of introduction of local time transitions, such as a DST transition at at 02:00. CCT supports these traditional schemes and also provides an alternate counting method that delays the transition to the end of the local day. This produces an uninterrupted sequence during the day. Both approaches are supported by CCT, as described below.

Local time is based on offsets from UTC time at the prime meridian and UTC time includes leap-seconds. Most systems, such as Posix time, rely on this basic idea and CCT carries on that useful tradition. A key function in CCT is CCct::SetSecsFrac(); that takes a count of UTC1970-based UTC time in seconds or seconds and fractions of seconds to populate the Common Calendar Binary Format (CBF) and Character Format (CCF).

The following sections describe detail of the CCT origin and counting methods for leap-seconds and Daylight Saving Time. Section Compatibility to NTP, PTP, Posix, and GPS shows the constant offsets to other important timescales.

3.1 CCT Origin

CCT uses a fixed origin of 1970-01-01 00:00:00 (UTC), called UTC1970. This origin is chosen to facilitate direct interface to Posix time "the Epoch" and IANA Time Zone Database (TzDb).

UTC1970 is exactly two years (730 days, or 63072000 seconds) before the alignment point between TAI and UTC as defined by ITU-R Recommendation 460:
1972-01-01 00:00:10 (TAI) = 1972-01-01 00:00:00 (UTC), called UTC1972.

3.2 Leap-seconds

Common Calendar employs a counting method called Time-of-day (TOD) Count Mode to define the relation of date and time-of-day portions of the YMDhms representation of the underlying seconds-since-1970 count:

- **Leap-second at UTC** – Leap-seconds are introduced in the local timescale YMDhms sequence simultaneous with introduction on the UTC timescale. This supports the common practice employed

in many systems and time dissemination protocols. Three variations are available to support the system counting behaviors of UTC, NTP, and Posix.

- **Leap-second at Midnight** - Leap-seconds are introduced in the local timescale YMDhms sequence at second before midnight in all time zones. This variation on common practice produces a simplified and symmetrical matrix of local time representations to support systems that must avoid discontinuities in the YMDhms counting sequence. The method is sometime referred to as "rolling leap-seconds"
- **None** – there is no relation between the date and time-of-day: the hours:minutes:seconds portion of the YMDhms representation are independent of the year-month-day portion. This supports representation of simple running clock values obtained on a given date, like a “stop watch”, or “game clock”.

The TOD Count Modes are enumerated in struct `CBFTodCountMode_et`.

```
typedef enum CBFTodCountMode_et
{
TOD_NONE = 0,             // Not Time-of-day
                          // Time has no relation to Date,
                          // Time has zero or "arbitrary" epoch
                          // CCF char indicator "a" (arbitrary)
TOD_LEAPSECOND_MIDNIGHT,  // Leap-seconds introduced at
                          // midnight on local timescales
                          // (Rolling leap-second)
                          // CCF char indicator "m" (midnight)
TOD_LEAPSECOND_UTC_UTC,   // Leap-seconds introduced
                          // simultaneous with UTC on
                          // local timescales
                          // leap-second label
                          // 23:59:60
                          // CCF char indicator "u" (utc)
TOD_LEAPSECOND_UTC_NTP,   // Leap-seconds introduced
                          // simultaneous with UTC on
                          // local timescales
                          // leap-second label
                          // 23:59:59 ("freeze")
                          // CCF char indicator "n" (ntp)
TOD_LEAPSECOND_UTC_POSIX, // Leap-seconds introduced
                          // simultaneous with UTC on
                          // local timescales
                          // leap-second label
                          // 00:00:00 ("roll over and reset")
                          // CCF char indicator "p" (posix)
TOD_24HOUR_DAY_DATE,  // 86400-second-days of calendar
                      // (leap-seconds unknown or unavailable)
                      // CCF char indicator "g" (gregorian)
TOD_NA                // not set or logic error (default)
} CBFTodCountMode_et;
```

See CBF.h

The leap-second at UTC and Leap-second at Midnight TOD Count Modes are further described in the following sections.

3.2.1 TOD Count Mode - Leap-second at UTC

Midnight on any local timescale is defined by its YMDhms representation of UTC Offset and Daylight Saving shifts, if used. Many timekeeping systems adopt the common practice of introducing leap-seconds on local timescales simultaneous with their introduction on the UTC timescale.

Unfortunately this generates a discontinuity in the YMDhms counting sequence sometime during the day between the local midnights (midday), and this produces a complex and asymmetrical relationship amongst the local timescales during leap-second days. Despite its being difficult to understand,

implement, and test, this convention is supported by Common Calendar to provide compatibility with these widely deployed systems..

Common Calendar supports this conventional Leap-second counting scheme in three variations:

1. TOD_LEAPSECOND_UTC_UTC - Leap-seconds are introduced simultaneous with UTC on local timescales and the Leap-second is labeled xx:59:60 as per UTC specification. Example Europe/Berlin.

Common Calendar Character Format (CCF)	Seconds since UTC1970
D2015-07-01T01:59:59U+02Zeurope/berlinAcestV2021aL25+S01cMuX	1435708824
D2015-07-01T01:59:60U+02Zeurope/berlinAcestV2021aL25*S01cMuX	1435708825
D2015-07-01T02:00:00U+02Zeurope/berlinAcestV2021aL26+S01cMuX	1435708826

2. TOD_LEAPSECOND_UTC_NTP - Leap-seconds are introduced simultaneous with UTC on local timescales and the Leap-second is labeled xx:59:59 as per NTP specification ("freeze").

Common Calendar Character Format (CCF)	Seconds since UTC1970
D2015-07-01T01:59:59U+02Zeurope/berlinAcestV2021aL25+S01cMnX	1435708824
D2015-07-01T01:59:59U+02Zeurope/berlinAcestV2021aL25*S01cMnX	1435708825
D2015-07-01T02:00:00U+02Zeurope/berlinAcestV2021aL26+S01cMnX	1435708826

3. TOD_LEAPSECOND_UTC_POSIX - Leap-seconds are introduced simultaneous with UTC on local timescales and the Leap-second is labeled xx:00:00 as per POSIX specification ("roll over and reset").

Common Calendar Character Format (CCF)	Seconds since UTC1970
D2015-07-01T01:59:59U+02Zeurope/berlinAcestV2021aL25+S01cMpX	1435708824
D2015-07-01T02:00:00U+02Zeurope/berlinAcestV2021aL25*S01cMpX	1435708825
D2015-07-01T02:00:00U+02Zeurope/berlinAcestV2021aL26+S01cMpX	1435708826

The following illustration and table shows examples of selected time zones for the 2015 Leap-second in TOD_LEAPSECOND_UTC_UTC Count Mode.

Illustration - TOD_LEAPSECOND_UTC_UTC Count Mode - 2015 Leap-second Introduction

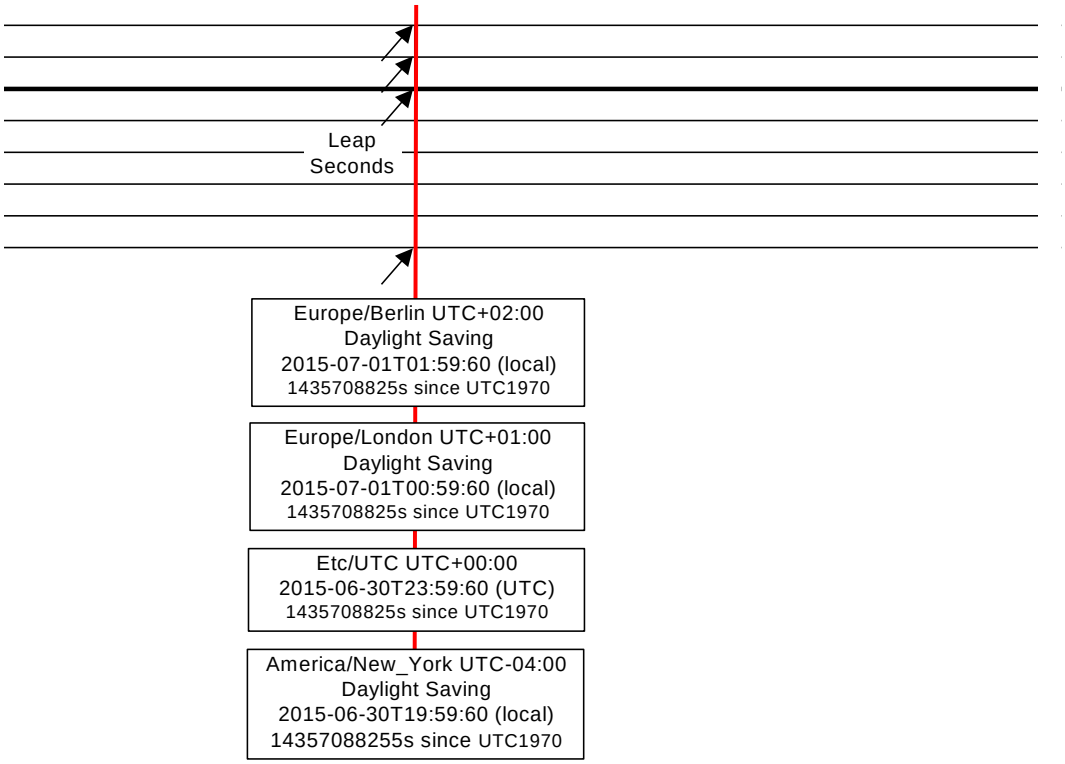

Table – TOD_LEAPSECOND_UTC_UTC Count Mode - 2015 Leap-second Introduction

Time zone	Common Calendar Character Format (CCF)	Seconds since UTC1970

Europe/Berlin	D2015-07-01T01:59:60U+02Zeurope/berlinAcestV2021aL25*S01cMuX	1435708825
Europe/London	D2015-07-01T00:59:60U+01Zeurope/londonAbstV2021aL25*S01cMuX	1435708825
Etc/UTC	D2015-06-30T23:59:60U+00Zetc/utcAutcV2021aL25*MuX_	1435708825
America/New_York	D2015-06-30T19:59:60U-04Zamerica/new_yorkAedtV2021aL25*S01cMuX	1435708825

3.2.2 TOD Count Mode - Leap-second at Local Midnight

The Common Calendar “Leap-second at Midnight” counting scheme introduces the Leap-second at the end of a Leap-second day in the local YMDhms representation to avoid the midday discontinuity produced by the conventional “Leap-second at UTC” method. This is sometimes known as "Rolling Leap-seconds".

This method results in a logically consistent and symmetrical matrix of local timescales, each with identical Leap-second counting methods and uniform phase, duration, and offset relationships amongst them. This simplifies implementation and testing which can lead to more consistent local timestamp representation.

The “Leap-second at Midnight” counting scheme was designed to support timekeeping systems that cannot tolerate midday YMDhms discontinuities. It may be chosen by systems seeking to simplify timekeeping procedures for more consistent and deterministic local timestamp implementation.

TOD_LEAPSECOND_MIDNIGHT Count Mode introduces Leap-seconds labeled 23:59:60 at the end of a Leap-second day on each Local Timescale. For Example:

Time zone	Common Calendar Character Format (CCF)	Seconds since UTC1970
Europe/Berlin	D2015-06-30T23:59:59U+02Zeurope/berlinAcestV2021aL25+S01cMmX	1435701624
Europe/Berlin	D2015-06-30T23:59:60U+02Zeurope/berlinAcestV2021aL25*S01cMmX	1435701625
Europe/Berlin	D2015-07-01T00:00:00U+02Zeurope/berlinAcestV2021aL26S01cMmX	1435701626

This count mode eliminates the midday discontinuities introduced in YMDhms representations by the common practice of introducing Leap-seconds simultaneous with its appearance in UTC. Instead, it places the Leap-second introduction at the end of a leap-second day on local timescales, consistent with the UTU-R Rec 460 specification.

The following illustration and table shows examples of selected time zones for the 2015 Leap-second in TOD_LEAPSECOND_MIDNIGHT Count Mode.

Illustration - TOD_LEAPSECOND_MIDNIGHT Count Mode - 2015 Leap-second Introduction

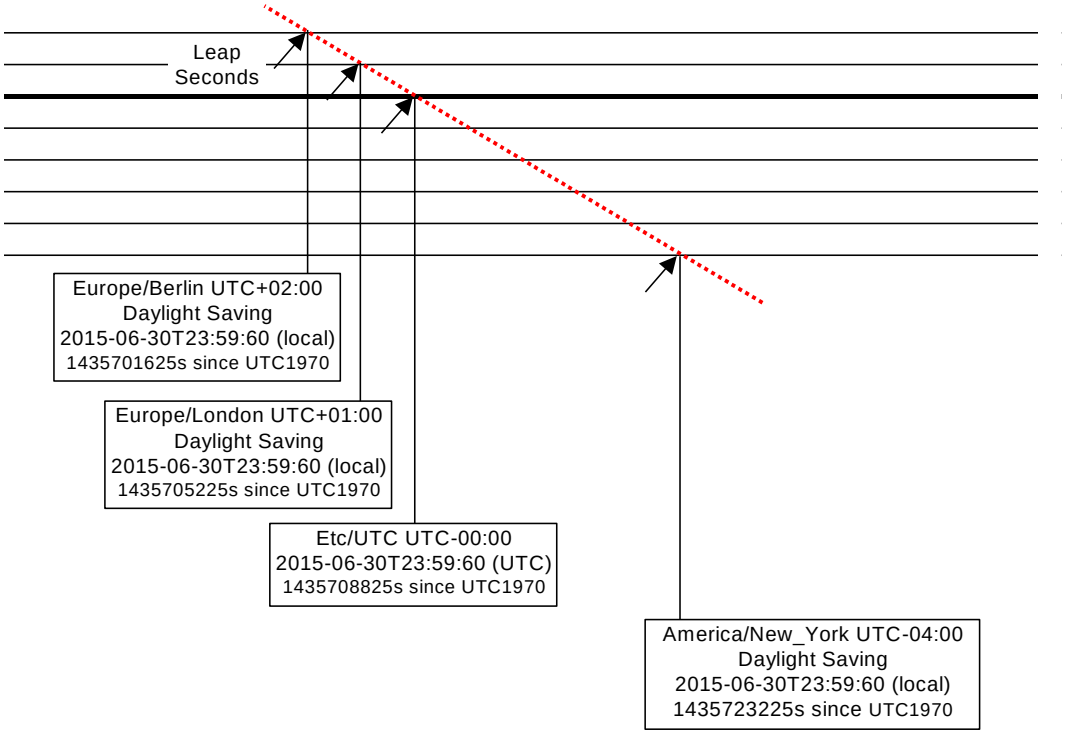

Table – TOD_LEAPSECOND_MIDNIGHT Count Mode - 2015 Leap-second Introduction

Leap-second Time zone	Common Calendar Character Format (CCF)	Seconds since UTC1970
Europe/Berlin	D2015-06-30T23:59:60U+02Zeurope/berlinAcestV2021aL25*S01cMmX	1435701625
Europe/London	D2015-06-30T23:59:60U+01Zeurope/londonAbstV2021aL25*S01cMmX	1435705225
Etc/UTC	D2015-06-30T23:59:60U+00Zetc/utcAutcV2021aL25*MmX	1435708825
America/New_York	D2015-06-30T23:59:60U-04Zamerica/new_yorkAedtV2021aL25*S01cMmX	1435723225

3.3 IANA Time Zone Database

IANA Time Zone Database (TzDb) is the de facto standard describing time zones and time zone rules. CCT's handling of local time is based directly on TzDb and the reference implementation makes direct use of adapted versions of TzDb supplied c-code functions.

Common Calendar specifies a policy of recording the local date and time as known to the emitting system in all cases. This brings clarity and specificity to the meaning of the data and consistency to generating and interpreting Common Calendar timestamps.

Conventional time zones and Daylight Saving Time are mapped to the local timescales as described in Common Calendar Time Zone API.

TzDb represents the common methods of introduction of local time transitions, such as a DST transition at at 02:00. CCT supports these traditional schemes and also provides an alternate counting method that delays the transition to the end of the local day. This produces an uninterrupted sequence during the day. Both approaches are supported by CCT, as described below.

3.4 Daylight Saving Time (DST) Count Mode

Two DST “count modes” are defined for YMDhms representations: “conventional” and “uninterrupted”.

- “DST Conventional Count Mode” produces the familiar and expected discontinuity in the YMDhms sequence with DST onset (DST on) or retreat (DST off).
- “DST Uninterrupted Count Mode” is provided for systems or applications that cannot tolerate, or choose to avoid, these discontinuities. This count mode delays the DST shift in the YMDhms sequence until the end of the day yielding an uninterrupted incrementing YMDhms value until midnight.

This capability is supported by rules and data of the Tz Database API and defined by the YMDhms API. See Annex - c++ Listings, Listing - Common Calendar Binary Format, excerpts from CBF.h, CBFDstCountMode_et

Enumeration of DST application to the YMDhms count sequence in CCF character format.

```
typedef enum CBFDstCountMode_et
{
DSTCOUNTMODE_NOTSETORNOTAPPLICABLE = 0,
DSTCOUNTMODE_CONVENTIONAL,
DSTCOUNTMODE_UNINTERRUPTED
} CBFDstCountMode_et;
```

See TzDatabaseApi.h `CBFDstCountMode_et`

See CBF.h `CBFDst_st::m_eDSTCountMode`

3.4.1 DST Count Mode - Conventional

The Conventional DST Count Mode produces the familiar and expected discontinuity in the YMDhms sequence at the DST change. Example America/New_York.

Common Calendar Character Format (CCF)	Seconds since UTC1970
DST Onset	
D2016-03-13T01:59:58U-05Zamerica/new_yorkAestV2021aL26S00t01a02cMuX	1457852424
D2016-03-13T01:59:59U-05Zamerica/new_yorkAestV2021aL26S00t01a02cMuX	1457852425
D2016-03-13T03:00:00U-04Zamerica/new_yorkAedtV2021aL26S01t01a02cMuX	1457852426
D2016-03-13T03:00:01U-04Zamerica/new_yorkAedtV2021aL26S01t01a02cMuX	1457852427
DST Retreat	
D2016-11-06T01:59:58U-04Zamerica/new_yorkAedtV2021aL26S01t-01a02cMuX	1478415624

D2016-11-06T01:59:59U-04Zamerica/new_yorkAedtV2021aL26S01t-01a02cMuX	1478415625
D2016-11-06T01:00:00U-05Zamerica/new_yorkAestV2021aL26S00t-01a02cMuX	1478415626
D2016-11-06T01:00:01U-05Zamerica/new_yorkAestV2021aL26S00t-01a02cMuX	1478415627

3.4.2 DST Count Mode - Uninterrupted

The Uninterrupted DST Count Mode produces a 23 hour day on DST Onset days and a 25 hour day for DST Retreat days for the typical 1 hour DST bias. This requires the seconds-to-YMDhms algorithm support counting to 24:59:59. Example America/New_York.

Common Calendar Character Format (CCF)	Seconds since UTC1970
DST Onset delayed until midnight (no midday discontinuity)	
D2016-03-13T01:59:58U-05Zamerica/new_yorkAestV2021aL26Swo23+01cMuX	1457852424
D2016-03-13T01:59:58U-05Zamerica/new_yorkAestV2021aL26Swo23+01cMuX	1457852425
D2016-03-13T02:00:00U-05Zamerica/new_yorkAestV2021aL26Swo23+01cMuX	1457852426
D2016-03-13T02:00:01U-05Zamerica/new_yorkAestV2021aL26Swo23+01cMuX	1457852427
DST Onset midnight rollover at 23 hours	
D2016-03-13T22:59:58U-04Zamerica/new_yorkAedtV2021aL26Swo23+01cMuX	1457924424
D2016-03-13T22:59:59U-04Zamerica/new_yorkAedtV2021aL26Swo23+01cMuX	1457924425
D2016-03-13T00:00:00U-04Zamerica/new_yorkAedtV2021aL26Ss+01cMuX	1457924426
D2016-03-13T00:00:01U-04Zamerica/new_yorkAedtV2021aL26Ss+01cMuX	1457924427
DST Retreat delayed until midnight (no midday discontinuity)	
D2016-11-06T01:59:58U-05Zamerica/new_yorkAestV2021aL26Ssr25+01cMmX	1478415624
D2016-11-06T01:59:59U-05Zamerica/new_yorkAestV2021aL26Ssr25+01cMmX	1478415625
D2016-11-06T02:00:00U-05Zamerica/new_yorkAestV2021aL26Ssr25+01cMmX	1478415626
D2016-11-06T02:00:01U-05Zamerica/new_yorkAestV2021aL26Ssr25+01cMmX	1478415627
DST Retreat midnight rollover at 25 hours	
D2016-11-06T24:59:58U-05Zamerica/new_yorkAestV2021aL26Ssr25+01cMmX	1478494824
D2016-11-06T24:59:59U-05Zamerica/new_yorkAestV2021aL26Ssr25+01cMmX	1478494825
D2016-11-06T00:00:00U-05Zamerica/new_yorkAestV2021aL26SwcMmX	1478494826
D2016-11-06T00:00:01U-05Zamerica/new_yorkAestV2021aL26SwcMmX	1478494827

3.5 Compatibility to NTP, PTP, Posix, and GPS

There are several established time dissemination systems widely used in civil timekeeping: NTP, PTP, Posix, and GPS. To facilitate interoperability with these systems Common Calendar Timescale defines a set of unique names of offsets from the Common Calendar native UTC1970 origin. Each defines an exact integral second offset to the origins of these timescales.

The Local Timescales can be converted to existing and traditional timescales. However the reverse, conversion of traditional to CCT Local Timescales, may not yield deterministic results in all cases, in particular at time points of leap-second introduction or when local time metadata is unavailable, out of date, or insufficient.

Table - Origins of Established Timescales

Origin Name	Seconds Since UTC1970	Coincides with Origin of	UTC	TAI-UTC	TAI
UTC1601	-11636784000	Windows	1601-01-01 00:00:00 (UTC)	10	1601-01-01 00:00:10 (TAI)
UTC1900	-2207520000	NTP Era 0	1900-01-01 00:00:00 (UTC)	10	1900-01-01 00:00:10 (TAI)
TAI1958	-378432000	TAI	1957-12-31 23:59:50 (UTC)	10	1958-01-01 00:00:00 (TAI)
TAI1970	-10	1588/PTP	1969-12-31 23:59:50 (UTC)	10	1970-01-01 00:00:00 (TAI)
UTC1970	0	Posix	1970-01-01 00:00:00 (UTC)	10	1970-01-01 00:00:10 (TAI)
TAI1972	63071990	UTC - 10s	1971-12-31 23:59:50 (UTC)	10	1972-01-01 00:00:00 (TAI)
UTC1972	63072000	UTC	1972-01-01 00:00:00 (UTC)	10	1972-01-01 00:00:10 (TAI)
UTC1972_7_1	+78796801	1st leap-second	1972-07-01 00:00:00 (UTC)	11	1972-07-01 00:00:11 (TAI)
UTC1980_1_6	+315964809	GPS	1980-01-06 00:00:00 (UTC)	19	1980-01-06 00:00:19 (TAI)

4 Annex A - Timescales, Origins, and Epochs

Timescales and their origins deserve more careful explanation. This annex describes in more detail how the CCT specifications and implementation treat this fundamental topic.

4.1 Timescales

ISO 8601 (ISO 8601-1, Date and time — Representations for information interchange — Part 1: Basic rules) defines "time scale" as having three basic factors:

3.1.1.2
time
mark attributed to an instant (3.1.1.3) or a time interval (3.1.1.6) on a specified time scale (3.1.1.5)

3.1.1.4
time axis
mathematical representation of the succession in time according to the space-time model of instantaneous events along a unique axis

3.1.1.5
time scale
system of ordered marks which can be attributed to instants (3.1.1.3) on the time axis (3.1.1.4), one instant being chosen as the origin

This can be summarized as three factors:

1) Time measurement unit
2) Origin
3) Labelling scheme

This annex is particularly concerned with the definition of the origins of important timescales: TAI, UTC, NTP, Posix, and PTP.

4.2 TAI to UTC Alignment

Most modern timescales and timekeeping systems rely on the alignment of TAI and UTC in the TAI/UTC system. This was established by the International Radio Consultative Committee (CCIR) when they published CCIR Recommendation 460 (Rec 460) in CCIR Report 517 as of 1971-02-23. Rec 460 established the TAI-to-UTC alignment to be:

1972-01-01 00:00:10 (TAI) = 1972-01-01T00:00:00 (UTC).

This alignment definition is at the foundation of modern timekeeping. Most systems define their timescales and origins with respect to UTC.

The difference between TAI and UTC, DTAI (TAI-UTC), at the alignment point is exactly 10 seconds. This was the number of positive leap-seconds that a had accumulated during the 10 year development period of the TAI/UTC system between 1960 until the final and unique adjustments were made on UTC1972. This 10 second offset is the initial calibration between TAI and UTC, and DTAI was set to 10. The first positive leap-second occurred on 1972-06-30 23:59:60 (UTC) and DTAI became 11 immediately after the leap-second at midnight on 1972-07-01 00:00:00 (UTC).

The CCIR was reorganized to become the ITU Radiocommunication Sector (ITU-R) in 1993 and remains the organization responsible for maintenance of Rec 460. The current version as of this writing is ITU-R TF Recommendation 460-6.

It may be worth noting that Rec 460 defines the difference between TAI and UTC in terms of their YMDhms representations, that is, DTAI (TAI-UTC) is the difference between YMDhms (TAI) and YMDhms (UTC). Some systems treat TAI as a zero-based uninterrupted incrementing counter, a useful means of representing TAI. However Rec 460 states that TAI is "*... in the form of a continuous scale, e.g. in days, hours, minutes and seconds from the origin 1 January 1958.*" and UTC is similarly treated. There is no meaningful relation of TAI to UTC in the form of seconds counting; DTAI is defined in terms of TAI and UTC as YMDhms representations.

4.3 Proleptic UTC

UTC did not exist before UTC1972 so any representation before that date must be a proleptic extrapolation. (In fact, while the initialisation "UTC" was used in various reports it was not officially endorsed until 1975.).

But the question is, does proleptic UTC include leap-seconds or not? For some purposes, such as astronomy, solar date and time must include leap-second compensation into the deep past. But for purposes of contemporary civil timekeeping this is not as important and it is convenient to assume there were no leap-seconds before UTC1972. All the computer timescales considered here make this convenient simplification. So, for these timescales "UTC" (before UTC1972) should be taken to mean *proleptic UTC without leap-seconds*.

However it must be noted that while these timescales are devoid of leap-seconds before UTC1972 they do retain the 10 second calibration offset at the TAI-UTC alignment point. That is, the value of DTAI (TAI-UTC) is 10 at and before UTC1972.

4.4 The term "Epoch"

The term "epoch" has many meanings in the timekeeping literature. The appearance of the term in the original Posix documents seems to have established its use in computer timekeeping.

Posix refers to the term "the epoch" as "Seconds Since the Epoch" in section 4.16 Seconds Since the Epoch.

Later, IEEE 1588 Precision Time Protocol defined "epoch" as:

7.2.3 Epoch
The epoch is the origin of the timescale of a domain.

This usage makes "epoch" and "origin" synonymous and this is often how "epoch" is used in some computer timekeeping documents and the literature.

4.5 Posix Time

The time management portion of the Portable Operating System Interface (Posix) lies at the root of most computer timekeeping. It specifies a wide range of data types and functions relevant to timekeeping.

Section 4.16 Seconds Since the Epoch of the standard states (in rather cumbersome terms) that "the Epoch" is 1970-01-01 00:00:00 (UTC). The use of term "UTC" here may be seen as controversial because UTC did not officially exist until UTC1972.

Note the line in that section: "*The relationship between the actual time of day and the current value for seconds since the Epoch is unspecified.*" This allows some old Posix implementations to be compliant with the specification. But modern systems treat 1970-01-01 00:00:00 (UTC) as being exactly 63072000 seconds before the TAI/UTC alignment at 1972-01-01 00:00:00 (UTC).

4.6 Network Time Protocol (NTP)

NTP is used to discipline the time of nearly all systems. NTP was designed to be compatible with Posix time and it's origin is explicitly and clearly stated in RFC 5905 as "*the prime epoch, or base date of era 0, is 0 h 1 January 1900 UTC*".

RFC 5905 goes on to point out:

"*It should be noted that strictly speaking, UTC did not exist prior to 1 January 1972, but it is convenient to assume it has existed for all eternity, even if all knowledge of historic leap seconds has been lost.*"

It is worth noting the table in RFC 5905 called *Figure 4: Interesting Historic NTP Dates*, here shown to highlight NTP, Unix/Posix, and UTC:

Date	MJD	NTP Era	NTP Timestamp Era Offset	Epoch
1 Jan 1900	15020	0	0	First day NTP Era 0
1 Jan 1970	40,587	0	2,208,988,800	First day UNIX
1 Jan 1972	41,317	0	2,272,060,800	First day UTC

This clearly states the relationship amongst these timescales.

- The NTP Era 0 origin is 1900-01-01 00:00:00 (UTC)
- The Unix/Posix origin is at 2208988800 wrt NTP
- The UTC (proper) origin is at 2272060800 wrt NTP

So: UTC1972 2272060800 - UTC1970 2208988800 = 63072000s

Thus UTC1970 is exactly 63072000s before UTC1972. This is how modern systems interpret and use UTC1970 as the Posix "the epoch", 1970-01-01 00:00:00 (UTC).

Note "UTC" before UTC1972 here is actually *proleptic UTC without leap-seconds" but it does include the initial 10s TAI-UTC calibration offset.

4.7 IEEE Precision Time Protocol

IEEE Std 1588-2019 specifies the PTP time-link protocol. It states:

7.2.3 Epoch
The epoch is the origin of the timescale of a domain.
The PTP epoch (the epoch of timescale PTP) is 1 January 1970 00:00:00 TAI.

Note this is 10 seconds before UTC1970 at 1970-01-01 00:00:00 (TAI) = 1969-12-31 23:59:50 (UTC)

4.8 Summary

The origin of Unix/Posix time is:

1970-01-01 00:00:10 (TAI) = 1970-01-01T00:00:00 (UTC)

This is also timescale and origin used by IANA Time Zone Database and most computer timekeeping systems. Common Calendar uses the UTC1970 origin to be directly compatible with the Posix and Tz Database.

Additional important timescales and their origins are shown in section 3.5 Compatibility to NTP, PTP, Posix, and GPS

4.9 References

ISO 8601-1, Date and time — Representations for information interchange — Part 1: Basic rules
https://www.iso.org/standard/70907.html

ITU-R TF.460-6, RECOMMENDATION ITU-R TF.460-6, Standard-frequency and time-signal emissions
https://www.itu.int/rec/R-REC-TF.460-6-200202-I/en

Time Scales - Steve Allen
https://www.ucolick.org/~sla/leapsecs/timescales.html

Portable Operating System Interface, IEEE Std 1003.1-2017
https://pubs.opengroup.org/onlinepubs/9699919799.2018edition/

ISO/IEC9899:2017, Programming languages — C
https://web.archive.org/web/20181230041359/http://www.open-std.org/jtc1/sc22/wg14/www/abq/c17_updated_proposed_fdis.pdf

IETF RFC 5905, Network Time Protocol Version 4: Protocol and Algorithms Specification
https://www.rfc-editor.org/rfc/rfc5905

The NTP Timescale and Leap Seconds - David Mills
https://www.eecis.udel.edu/~mills/leap.html

IEEE Std 1588™-2019, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems
https://standards.ieee.org/ieee/1588b/10396/

IANA Time Zone Database
https://www.iana.org/time-zones