Common Calendar Character Format

Local Time Timestamp System

Brooks Harris Version 3 2024-04-25

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).

1 Introduction

People generally expect calendar date and time-of-day to be shown in some form of YMDhms, not some form of machine efficient integer count. A seconds-since-UTC1970 value of 78796801s is unintelligible to a human, while a YMDhms encoding of this number as 1972-07-01 00:00:00 (UTC) imparts immediate familiar meaning.

Machine readable character based interchange formats using a YMDhms form are common. They provide a machine readable format that is also meaningful to humans. However none are capable of representing unambiguous UTC accurate local time because they lack sufficient metadata to fully describe the meaning of the YMDhms values.

Common Calendar Character Format (CCF) specifies a character based machine-readable interchange format in YMDhms form to impart familiar meaning to human users with the necessary and sufficient metadata to fully describe local date and time for interoperable machine interchange. It also supports representation of time points and intervals unrelated to calendar date.

CCF is designed to reflect equivalent data and metadata contained in Common Calendar Binary Format (CBF) to provide symmetrical and complete conversion between the two. They are designed as tightly coupled pair, and this specification references the CBF specification and the CCT reference implementations of CCF and CBF to make clear the connections between the CBF data and metadata, the corresponding CCF character representations, and the details of conversion algorithms required.

The CCF format can be used independently of CBF if sufficient information is available. It may be most convenient to implement CCF in combination with CBF.

The CCT reference implementation implements CCF and CBF in the CCTLib library. CCF is implemented as class CCcf, and delimiter characters are explicitly defined in the CCF.h header. See CCTLib/CCF.h, CCcf.h, and CCcf.cpp. CBF is implemented as class CCbf and the CBF data types are defined in CBF.h header. See CCTLib/CBF.h, CCbf.h, and CCct.cpp.

2 Scope

This interoperability standard specifies a character based machine-readable format consisting of data and metadata elements to fully represent deterministic time points and time intervals including UTC accurate local date and time.

3 Normative References

Common Calendar Date and Time Terms and Definitions

Common Calendar TAI-UTC API

Common Calendar YMDhms API

Common Calendar Time Zone API

Common Calendar Local Timescales

Common Calendar Binary Format (CBF)

ISO 8601 2004-12-01, Data elements and interchange formats — Information interchange —
Representation of dates and times
http://www.iso.org/iso/iso8601

Common Calendar Geostamp

National Marine Electronics Association
NMEA 0183 Interface Standard
GGA Global Positioning System Fix Data. Time,

Position and fix related data for a GPS receiver $GPGGA
https://www.nmea.org/nmea-0183.html

4 Common Calendar Character Format (CCF)

Common Calendar Character Format (CCF) specifies a character based machine-readable format of date and time in a YMDhms form to impart familiar meaning to human users. CCF reflects equivalent data of the CBF binary format providing complete symmetrical conversion between the two.

CCF construction does not rely on external metadata, only the values contained in a CBF. CCF contains sufficient information to populate a binary CBF without reference to any external information.

A CCF, like its corresponding CBF format, can have one of five meanings:

- time point with UTC accurate local date and time-of-day
- time point with UTC accurate local date with time portion having no relation to the date
- time point less than 24 hours with no relation to date
- time point 24 hours or greater with no relation to date
- time interval less than 24 hours (< 86400 seconds)
- time interval 24 hours or greater (>= 86400 seconds)

In the case of representation of a time-point of UTC accurate local date and time-of-day a CCF and its corresponding CBF shall contain the local date, time-of-day, and local metadata as known to the emitting system when generated in accordance with the rules and guidelines set out in Common Calendar Local Timescales.

Examples of these configurations are shown in *Annex C - Example Listing from CCT Reference Implementation.* This listing is generated by the CCT reference implementations. See `CCTDemoConsole, CCTDemoConsole.cpp, CCTDemoTests.cpp, TestSelectedConfigurationsAndShowCbfValues().`

4.1 ISO 8601 Variation

The CCF format is based on the guidelines set out in ISO 8601 with important variations. The 8601 scheme is augmented in several ways including:

- Formatting of hms shall use the full hh:mm:ss form
- If date is given formatting of date and time shall use the full YYYY-MM-DDThh:mm:ss form and shall include all required applicable metadata elements.
- Clock rate is added.
- Leap-second value is added.
- Time zone , DST and UTC-offset shift metadata is added
- The character "Z" is used to delimit the time zone identification data field, replacing the ISO 8601 use of “Z” for "Zulu"

CCF does not support any form of partial representation of local date and time such as date only, "YYYY-DD-MM", or date and time only, "YYYY-MM-DDThh:mm:ss" because these are ambiguous without accompanying metadata. If date is given, CCF and CBF support only complete and deterministic representation of local date and time including the required local time metadata. Other representations are outside the scope of CBF and CCF.

4.2 Character Set

CCF shall be encoded using the ASCII character set excluding control characters and white space as detailed in *Annex A - CCF Character Set*.

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
!"#$%&'()*+,-./:;<=>?@[]^_`{|}~
```

4.3 Hard Terminator

The CCF string shall be terminated with an upper case "X" in all cases.

4.4 Data Field Elements

A CCF is a variable length string and its total length depends on the included optional elements and their contents. Some elements are fixed length, others variable length.

CCF shall be encoded as several optional data field elements delimited by a single designated upper-case letter and terminated with "X" to facilitate parsing. The length of variable length elements is bounded by its delimiting character and the next delimiting character or terminator. The syntax is variable length with no white space.

CCF delimiter characters and encoding characters are explicitly defined in CCTLib/CCF.h. The required elements and order of assembly are described in section *4.5 Assembly and Order*. Each element is described in sections below.

Delimiting Character	Element	Example fragment(s)
T	Time: • singly - time point < 86400s • with Date and TOD_LEAPSECOND_MIDNIGHT or TOD_LEAPSECOND_UTC_UTC or TOD_LEAPSECOND_UTC_NTP or TOD_LEAPSECOND_UTC_POSIX - time-of-day of UTC accurate local date • with Date and TOD_NONE - accurate local date and time having no relation to the date • with Event - time point >= 86400s	`T23:59:59n999999999`
E	Event with Time - time point >= 86400s	`E123T23:59:59n999999999`
I	Interval: • singly - time Interval < 86400s • with Period - time interval >= 86400s	`I23:59:59n999999999`
P	Period with Interval - time interval >= 86400s	`P123I23:59:59n999999999`
D	Date	`D2016-02-22`
U	UTC Offset	`U-5, U+5, U+02:30:17, U+3w01+02`
Z	(Z)one (time zone)	`Zamerica/new_york`
A	Posix zone abbreviation	`Aest`
Q	Posix zone abbreviation Change	`Qewt_ept+19`
V	Tz Database tzdata release version	`V2021a`
L	Leap-seconds	`L27`
S	Daylight Saving bias, state and mode	`Ss+01c`
M	Time-of-day (TOD) Count Mode:	`Mm`
X	Hard terminator	`X`

4.4.1 Time Element

The time element represents a time point or time interval in the familiar hours:minutes:seconds form, that is; 60 seconds = 1 minute, 60 minutes = 1 hour, and 24 hours = one 24 hour period (86400 seconds). In the case of representing UTC accurate data and time, a leap-second day includes an additional second (61 seconds, "23:59:60") and has a duration of 86401 seconds.

The corresponding CBF binary data structure is CBFTime_st. The CBF member variable: CBFTime_st::m_bIsInterval flags if the data represents a time point (m_bIsInterval == false) or interval (m_bIsInterval == true).

The first eight characters of the time element shall be fixed length containing two-digit values of hours, minutes, and seconds, separated by colons.

If the character following the hh:mm:ss field is a delimiter or terminator, the resolution of the time measurement unit shall be seconds. The corresponding value of a CBF CBFTime_st::m_eRateEnumeration member is CLOCK_0. Illustration: `"00:00:00X"`

For higher rates designated characters following the seconds field encode the decimal fractions of seconds enumerated clock rates and indicate the number of digits following the indicator character.

If the character following the hh:mm:ss field is one of the lower-case rate indicators the time measurement unit shall correspond to the value of a CBFRate_et enumeration in the CBFTime_st::m_eRateEnumeration member and the number of digits of the decimal fractions of seconds shall be as shown in the following table.

Character Encoding	CBF rate CBFRate_et	rate resolution	digits
none	CLOCK_0	1/1 second	none
t	CLOCK_1	1/10 tenths	1
h	CLOCK_2	1/100 hundredths	2
m	CLOCK_3	1/1000 Millisecond	3
u	CLOCK_6	1/1000000 Microsecond	6
h	CLOCK_7	1/10000000 100-Nanosecond	7
n	CLOCK_9	1/1000000000 Nanosecond	9
p	CLOCK_12	1/1000000000000 Picosecond	12
f	CLOCK_15	1/1000000000000000 Femtosecond	15
a	CLOCK_18	1/1000000000000000000 Attosecond	18

Example fragment:

```
00:00:00h1234567X
        ^ rate indicator CLOCK_7
```

4.4.1.1 UTC accurate local date and time-of-day

If the Time Element appears in combination with the Date Element and the TOD Count Mode is TOD_LEAPSECOND_MIDNIGHT or TOD_LEAPSECOND_UTC_UTC or TOD_LEAPSECOND_UTC_NTP or TOD_LEAPSECOND_UTC_POSIX the Time Element shall represent the time-of-day portion of a complete UTC accurate local date and time-of-day time-point including DST and.or UTC-offset Shift metadata if applicable. See Date Element and DST Element.

The corresponding CBF binary data structures are CBFTime_st., CBFLocalDate_st, and CBFDst_st.

Delimiters	Description	CBF member variable state
D and T plus applicable metadata delimiters	UTC accurate local date and time-of-day time point	CBFTime_st::m_bIsInterval == false CBFLocalDate_st::m_eTODMode == TOD_LEAPSECOND_MIDNIGHT or TOD_LEAPSECOND_UTC_UTC or TOD_LEAPSECOND_UTC_NTP or TOD_LEAPSECOND_UTC_POSIX

Example:

```
D1972-06-30T19:59:60U-04Zamerica/new_yorkAedtV2021aL00*Ss+01cMuX
```

4.4.1.2 UTC accurate local date and time having no relation to date

If the time element appears in combination with the Date Element and the TOD Count Mode is TOD_NONE, character Ma ,it shall represent a time having no relation to the UTC accurate local date. The DST Element is not allowed. See Date Element. No

The corresponding CBF binary data structures are CBFTime_st., CBFLocalDate_st, and CBFDst_st.

Delimiters	Description	CBF member variable state
D and T plus applicable metadata delimiters	Time-of-day unrelated to UTC date	CBFTime_st::m_bIsInterval == false CBFLocalDate_st::m_eTODMode == TOD_NONE

Example:
`D1972-06-30T12:34:56U-04Zamerica/new_yorkAedtV2021aL00MaX`

4.4.1.3 Time point less than 86400s

If the Time Element appears alone delimited by uppercase "T" it shall represent a time point less than 86400s.

Delimiter	Description	CBF member variable state
`T`	Time point ((T)ime) less than 24 hours (< 86400 seconds) with no relation to date	`CBFTime_st::m_bIsInterval == false`

Example:
`T23:59:59X` – time-point (Time) in seconds

4.4.1.4 Time point equal or greater than to 86400s

If the Time Element appears in combination with an Event Element it shall represent a time point greater than or equal to 86400s and be delimited by uppercase "T". See Event Element.

Delimiters	Description	CBF member variable state
`E` and `T`	Time point ((E)vent) 24 hours or greater (>= 86400 seconds) with no relation to date	`CBFTime_st::m_bIsInterval == false`

Example:
`E123T23:59:59X` - time-point (Event) >= 24 hours in seconds

4.4.1.5 Interval less than 86400s

If the Time Element appears alone delimited by uppercase "I" it shall represent an interval less than 86400s.

Delimiter	Description	CBF member variable state
`I`	Interval ((I)nterval) less than 24 hours (< 86400 seconds)	`CBFTime_st::m_bIsInterval == true.`

Example:
`I23:59:59u999999X` – interval (Interval) < 24 hours in microseconds

4.4.1.6 Interval equal or greater than 86400s

If the Time Element appears in combination with an Period Element it shall represent an interval equal or greater than 86400s and be delimited by uppercase "I". See Period Element.

Delimiters	Description	CBF member variable state
`P` and `I`	Interval ((P)eriod) 24 hours or greater (>= 86400 seconds)	`CBFTime_st::m_bIsInterval == true.`

Example:
`P123I23:59:59u999999X` – interval (Period) >= 24 hours in microseconds

Examples summary:

`T23:59:59X`	– time-point (Time) in seconds
`T23:59:59m999X`	– time-point (Time) in milliseconds
`T23:59:59n999999999X`	– time-point (Time) in nanoseconds
`I23:59:59X`	– interval (Interval) < 24 hours in seconds
`I23:59:59u999999X`	– interval (Interval) < 24 hours in microseconds
`E1T23:59:59X`	– time-point (Event) >= 24 hours in seconds
`E12T23:59:59m999X`	– time-point (Event) >= 24 hours in milliseconds
`E123T23:59:59n999999999X`	– time-point (Event) >= 24 hours in nanoseconds
`P1I23:59:59X`	– interval (Period) >= 24 hours in seconds
`P2I23:59:59u999999X`	– interval (Period) >= 24 hours in microseconds
`D1972-06-30T19:59:60U-05:00Znew_yorkV2016cL10*Sc4sMuX`	– Local Date and time

See CBF.h,
`CBFTime_st`

```
   CBFRATE_et
```

See CCct.h

```
   CCct::SetIntervalFromSecondsFrac_st()
   CCct::Set24HourTimepointFromSecondsFrac_st
```

See CCcf.cpp,

```
   CCcf::SetTimeFromCCbf()
   CCcf::ParseTimeSetCCbf()
```

4.4.2 Event Element

Encodes a 24 hour period (86400 seconds) count value. If given, shall be used in combination with a Time Element to represent a time interval equal to or greater than 86400 seconds. See Time Element.

The term "24 hour period" is used in this context to avoid the word "day" because only UTC, with its occasional 86401 second leap-second days, represents accurate calendar dates. An Event Element indicates a count of fixed-length 86400 second periods, not UTC days.

The corresponding CBF binary data structure is CBF24HourPeriod_st.

This is a range of approximately 27378 years, far greater than the approximate 3000 year range supported by Common Calendar. The corresponding CBF 21-bit data member CBF24HourPeriod_st:: m_ul24HourPeriods has a range of 2^21-bits = 2097152 MAX.

Delimiters	Description	CBF member variable state
E and T	Time point ((E)vent) 24 hours or greater (>= 86400 seconds) with no relation to date	CBFTime_st::m_bIsInterval == false

Examples:

```
E1T23:59:59X            - time-point (Event) >= 24 hours in seconds
E12T23:59:59m999X       - time-point (Event) >= 24 hours in  milliseconds
E123T23:59:59n999999999X - time-point (Event) >= 24 hours in nanoseconds
```

See CBF.h, CBF24HourPeriod_st
CBF24HourPeriod_st::m_ul24HourPeriods
See CCct.h

```
   CCct::SetIntervalFromSecondsFrac_st()
   CCct::Set24HourTimepointFromSecondsFrac_st
```

See CCcf.cpp, CCcf::SetCcfFromCCbf()
CCcf::SetCcfFromCCbf_Interval()
CCcf::SetCcfFromCCbf_Timepoint()
CCcf::ParseIntervalSetCCbf()
CCcf::Parse24HourIntervalSetCCbf()
CCcf::Parse24HourEventSetCCbf()

4.4.3 Period Element

Encodes a 24 hour interval (86400 seconds) value. If given, shall be used in combination with an Interval Element to represent a interval equal to or greater than 86400 seconds. See Interval Element.

The corresponding CBF binary data structure is CBF24HourPeriod_st.

Delimiters	Description	CBF member variable state
P and I	Interval ((P)eriod) 24 hours or greater (>= 86400 seconds)	CBFTime_st::m_bIsInterval == true.

Examples:

```
P1T23:59:59X            - interval (Period) >= 24 hours in seconds
P2T23:59:59u999999X     - interval (Period) >= 24 hours in microseconds
```

See CBF.h, CBF24HourPeriod_st
CBF24HourPeriod_st::m_ul24HourPeriods
See CCct.h

```
   CCct::SetIntervalFromSecondsFrac_st()
   CCct::Set24HourTimepointFromSecondsFrac_st
```

See CCcf.cpp, CCcf::SetCcfFromCCbf()

CCcf::SetCcfFromCCbf_Interval()
CCcf::SetCcfFromCCbf_Timepoint()
CCcf::ParseIntervalSetCCbf()
CCcf::Parse24HourIntervalSetCCbf()
CCcf::Parse24HourEventSetCCbf()

4.4.4 Date Element

Encodes the local date.

The Date Element shall be fixed length 11 characters including delimiter in the form DYYYY-MM-DD, as shown in the following table.

Delimiter	Year (YYYY)	Dash	Month (MM)	Dash	Day (DD)
D	4 digit year (YYYY)	-	2 digit month (MM) with leading zeros	-	2 digit day of month (DD) with leading zeros

Example fragment:
`D1972-01-01`

If the optional YYYY-MM-DD Date Element is present it shall represent the complete local calendar date and time-of-day together with sufficient metadata to fully describe local data and time. The optional DST Element shall add Daylight Saving parameters if applicable and the UTC-Offset Shift Element shall add UTC-Offset shift parameters if applicable.

If date is given it shall be accompanied by

- Time Element (time-of-day)
- Local UTC Offset Element
- Time Zone Element
- Tz Database Version Element
- Leap-seconds Element
- DST Element (if applicable)
- UTC-Offset Shift Element (if applicable)
- TOD Count Mode Element

A Date element together with its required Time element and metadata elements describe a time-point, indicated by CBF `CBFTime_st::m_bIsInterval == false`.

A CBF shall include the CBFLocalDate_st structure, indicated by CBFTime_st::m_bLocalDateExt == true

If the YYYY-MM-DD date element is present the hh:mm:ss time element values shall represent the time-of-day portion of a complete UTC accurate local date and time on the date indicated by the YYYY-MM-DD values of the date element, including common-days of 86400-seconds and leap-second-days of 86401 seconds. DST information shall be provided by the DST element, if applicable.

The sequence of YMDhms values shall be governed by the TOD Count Mode CBFTodCountMode_et enumerated values of CBFLocalDate_st::m_eTODMode and DST Count Mode CBFDstCountMode_et CBFDst_st::m_eDSTCountMode (if applicable) in effect.

See Common Calendar Local Timescales,:
4.1 Daylight Saving Time (DST) Count Mode.
4.2 Time-of-day (TOD) Count Mode and leap-second Introduction

Identically to CBF CBFLocalDate_st and CBFDst_st parameters, the values and metadata of the date and DST elements (if applicable) shall represent local time as known to the emitting system in all cases.

Construction of CCF YMDhms from CBF binary data shall employ the algorithms described by the YMDhms API together with logic to apply DST and/or UTC-offset shifts if applicable. The YMDhms values will depend on CBFLocalDate_st::m_eTODMode and CBFDst_st::m_eDSTCountMode if applicable.

CCF delimiter characters and encoding characters are explicitly defined in CCTLib/CCF.h.

The details of the CBF to CCF conversion are shown in:
CCcf.cpp, CCcf::SetCcfFromCCbf()

CCcf::SetCcfFromCCbf_TOD_LEAPSECOND_UTC()
CCcf::SetCcfFromCCbf_TOD_LEAPSECOND_MIDNIGHT()

Parsing of a CCF string and populating a CBF binary is shown in: CCcf.cpp, CCcf::ParseDateSetCbf()

The corresponding CBF components are:
CBFTime_st
CBFLocalDate_st
CBFDst_st (if applicable).
CBFUtcShift_st (if applicable).

CBF CBFLocalDate_st records the date in a compound counter:
CBFLocalDate_st::m_DateTaiUtc_st.m_ui1970DayNumber
CBFLocalDate_st::m_DateTaiUtc_st.m_ui1970TAI_UTC

See CCcf.cpp, CCcf::SetCcfFromCCbf()
CCcf::SetCcfFromCCbf_TOD_LEAPSECOND_MIDNIGHT()
CCcf::SetCcfFromCCbf_TOD_LEAPSECOND_UTC()
CCcf::ParseDateSetCbf()

Examples:

```
D1972-06-30T19:59:60U-04Zamerica/new_yorkAedtV2021aL00*Ss+01cMuX
```

- 1972-06-30T19:59:60 seconds, date and time
- U-04 - UTC-offset
- Zamerica/new_yorkAedtV2021aL0 - time zone
- Aedt - Posix environment time zone abbreviation
- V2021a - Tz Database version
- L0 - Leap-seconds
- * - Is Leap-second indicator
- Ss+01c - DST
  s - DST in effect (summer time)
  +01 - 01:00 DST bias
  c - DST Count Mode DSTCOUNTMODE_CONVENTIONAL
- Mu - TOD Count Mode TOD_LEAPSECOND_UTC_UTC
- X - terminator

```
D1972-07-01T00:59:60U+1Zeurope/berlinAcetV2021aL00*SwcMuX
```

- 1972-07-01T01:59:60 seconds, date and time
- U+1 - UTC offset
- Zeurope/berlin - time zone
- V2021a - Tz Database version
- Acet - Posix environment time zone abbreviation
- L0 - Leap-seconds
- * - Is Leap-second indicator
- Swc - DST
  w - DST not in effect (winter time)
  c - DST Count Mode DSTCOUNTMODE_CONVENTIONAL
- Mu - TOD Count Mode TOD_LEAPSECOND_UTC_UTC
- X - terminator

4.4.5 TOD Count Mode Element

Encodes the TOD Count Mode value.

TOD Count Mode indicates the relation between the DYYY-MM-DD and hh:mm:ss portions of the CCF and the resulting sequence of YMDhms representation.

Required if, and applicable only if, the Date Element "D" is given.

The TOD Count Mode Element shall be a fixed length 2 character string including the M (TOD (M)ode) delimiter and a single lower-case encoding character as show below.

Delimiter	TOD Count Mode Indicator

M	one character encoding as shown in the following table

TOD Count Mode shall be indicated by a single lower-case character as shown in the following table.

Character Encoding	TOD Count Mode CBFTodCountMode_et	Description
a	TOD_NONE	Time has no relation to date or time-of-day
m	TOD_LEAPSECOND_MIDNIGHT	Leap-seconds introduced at midnight on local timescales (Rolling leap-second)
u	TOD_LEAPSECOND_UTC_UTC	Leap-seconds introduced simultaneous with UTC on local timescales. Leap-second label 23:59:60 (UTC specification)
n	TOD_LEAPSECOND_UTC_NTP	Leap-seconds introduced simultaneous with UTC on local timescales. Leap-second label 59:59:59 ("freeze")
p	TOD_LEAPSECOND_UTC_POSIX	Leap-seconds introduced simultaneous with UTC on local timescales. Leap-second label 00:00:00 ("roll over and reset")
g	TOD_24HOUR_DAY_DATE	86400-second-days of calendar (leap-seconds unknown or unavailable)
	TOD_NA	not set or logic error (default)

Example fragments:
`Ma` - TOD_NONE
`Mm` - TOD_LEAPSECOND_MIDNIGHT
`Mu` - TOD_LEAPSECOND_UTC_UTC
`Mn` - TOD_LEAPSECOND_UTC_NTP
`Mp` - TOD_LEAPSECOND_UTC_POSIX
`Mg` - TOD_24HOUR_DAY_DATE

CCF delimiter characters and encoding characters are explicitly defined in CCTLib/CCF.h.

The corresponding CBF member is CBFLocalDate_st::m_eTODMode.

See CBF.h, CBFTodCountMode_et
CBFLocalDate_st::m_eTODMode
See CCcf.cpp, CCcf::SetCcfFromCCbf()
CCcf::ParseTODModeSetCCbf()

4.4.6 UTC-Offset Element

Encodes the local time UTC offset from UTC

The value represents the UTC-offset of local time from UTC. It is the sum of the current UTC-offset of CCbf::CBFLocalDate_st.m_lUTCOffset (called STDOFF in Tz Database) plus DST bias if applied, and any UTC-offset shift if applicable. It is sometimes called "current local offset", and this is called GMTOFF in Tz Database. It is consistent with ISO 8601 representations.

The value is a variable length +-hh:mm:ss representation. A sub-field may appear to encode (rare) UTC-offset shifts. See UTC-offset Shifts.

The local UTC-Offset element shall be delimited with upper-case "U" followed by "+" (East) or "-" (West) followed by a variable length hh:mm:ss representation.

A two digit hour shall always be present.
If minutes is non-zero a two digit minute shall be present separated from hours by a ":" (colon).
If seconds is non-zero a two digit seconds shall be present separated from minutes by a ":" (colon).

Delimiter	UTC-offset sign	UTC-offset	
U	+ or -	2 digit hour	all cases
		":" 2 digit minute	If minutes is non-zero
		":" 2 digit second	If seconds is non-zero

Example fragments:

```
U+00 U-05 U+01 U-10 U+14
U-00:01 U+07:30
U-00:01:15 U+07:30:23
```

The corresponding CBF member is CBFLocalDate_st::m_lUTCOffset

See CBF.h, CBFLocalDate_st::m_lUTCOffset
See CCcf.cpp, CCcf::SetUTCOffsetAndZoneFromCCbf()
CCcf::ParseUTCOffsetSetCCbf()

4.4.6.1 UTC-Offset Shift Sub-fields

Encodes (rare) cases where a time zone has shifted the UTC-offset (STDOFF).

4.4.6.1.1 UTC-offset shift East or West

The UTC-Offset Shift Sub-field encodes the UTC-offset direction (East or West), shift amount and shift time-of-day.

e (TZDUtcShiftDay_et UTCSHIFT_EAST, East is positive shift)
w (TZDUtcShiftDay_et UTCSHIFT_WEST, West is negative shift)
hh:mm:ss variable length UTC-offset shift amount
+hh:mm:ss variable length UTC-offset shift time-of-day (local time)

if UTC-offset shift is East (UTCSHIFT_EAST) a lower-case "e" shall be appended.
if UTC-offset shift is West (UTCSHIFT_WEST) a lower-case "w" shall be appended.

Character Encoding	Description	CBF enumeration TZDUtcShiftDay_et
e	UTC-offset shift East - positive	UTCSHIFT_EAST
w	UTC-offset shift West - negative	UTCSHIFT_WEST

4.4.6.1.2 UTC-offset shift amount

if UTC-offset shift is positive the value shall be preceded by "+".
if UTC-offset shift is negative the value shall be preceded by "-".
A two digit hour shall always be present.
If minutes is non-zero a two digit minute shall be present separated from hours by a ":" (colon).
If seconds is non-zero a two digit seconds shall be present separated from minutes by a ":" (colon).

	UTC-offset shift sign	UTC-offset shift	
	+ or -	two digit hour	all cases
		":" two digit minute	If minutes is non-zero
		":" two digit second	If seconds is non-zero

4.4.6.1.3 UTC-offset shift time-of-day (local time)

if UTC-offset shift time-of-day shall be preceded by "+".
A two digit hour shall always be present.
If minutes is non-zero a two digit minute shall be present separated from hours by a ":" (colon).
If seconds is non-zero a two digit seconds shall be present separated from minutes by a ":" (colon).

UTC-offset shift time-of-day delimiter	UTC-offset shift	
+	two digit hour	all cases
	":" two digit minute	If minutes is non-zero
	":" two digit second	If seconds is non-zero

typedef enum TZDUtcShiftDay_et
See CBF.h, CBFLocalDate_st::m_bUtcShiftExt
See CBF.h, CBFUtcShift_st
See CCcf.cpp, CCcf::SetUTCOffsetAndZoneFromCCbf()
CCcf::ParseUTCOffsetSetCCbf()

See TzDatabaseApi.h

Example fragments:
`U+00w00:01:15+00:01:15` - shift West by 00:01:15 at +00:01:15
`U+02e01+03` - shift East by 01:00:00 at +03:00:00
Example:

```
D2011-03-27T01:59:59U+03e01+02Zeurope/moscowAmskV2021aL24MuX
D2011-03-27T03:00:00U+04e01+02Zeurope/moscowAmskV2021aL24MuX
```

4.4.7 Time Zone Element

Encodes the Tz Database time zone identifier name.

The Time Zone Element shall be delimited with Z ((Z)one) followed by a variable length string of lower-case Tz Database time zone name identifiers including and forward slash "/". For example, "America/New_York" becomes "america/new_york".

Delimiter	Time Zone name
Z	variable length string as lower-case of Tz Database time zone name identifiers

Example fragments:

```
Zetc/utc
Zamerica/new_york
Zamerica/los_angles
Zeurope/london
```

Example

```
D2024-03-10T03:00:00m000U-04Zamerica/new_yorkAedtV2021aL27S01t01a02cMuX
```

CCF delimiter characters and encoding characters are explicitly defined in CCTLib/CCF.h.

The corresponding CBF member is CBFLocalDate_st::m_TZDTimeZone_st.m_uneTZDZoneName_et

See TzDatabaseApi.h, TZDZoneName_et
See CTzZoneTable.cpp, CTzZoneTable::m_CTzZone_stTable[]
See CBF.h, CBFLocalDate_st::m_TZDTimeZone_st
See CCcf.cpp,
    CCcf::SetUTCOffsetAndZoneFromCCbf()
    CCcf::ParseZoneSetCCbf()
See Common Calendar Time Zone API

4.4.8 Posix Abbreviation Name Element

Encodes the Posix-time TZ environment variable abbreviated time zone name as defined by Tz Database.

The Posix Abbreviation Name Element shall be delimited with A ((A)bbriveation) followed by a variable lower case string.

Delimiter	Posix Abbreviated Time Zone name
A	variable length string as lower-case of Posix abbreviated time zone name

Example fragments:

```
Aest
Aedt
```

Example

```
D2024-03-10T03:00:00m000U-04Zamerica/new_yorkAedtV2021aL27S01t01a02cMuX
```

The corresponding CBF member is m_aCBFChar_st16Abbr[16];

See CCcf.cpp,
CCcf::SetUTCOffsetAndZoneFromCCbf()
CCcf::ParseDstSetCCbf()

4.4.9 Posix Abbreviation Name Change Element

Encodes the Posix-time abbreviated name changes.

Most transitions have some change of parameters and the abbreviated name naturally follows. But some examples have *only* a change to the abbreviated name. This type of transition is supported by this element.

The Abbreviation Name Element shall be delimited with Q followed by a variable lower case string.

Delimiter	Posix Abbreviated Time Zone name
Q	variable length string as lower-case of Posix abbreviated time zone name

Example fragments:

```
Qewt_ept+00Aedt
Qewt_ept+00
```

Examples

```
D1945-08-14T18:59:59U-04Zamerica/new_yorkAewtQewt_ept+00V2021aL00S01cMuX
D1945-08-14T19:00:00U-04Zamerica/new_yorkAeptQewt_ept+00V2021aL00S01cMuX
```

The corresponding CBF member is CBFAbbrChange_st;

The presence of the Posix Abbreviation Name Change is indicated by
CBFLocalDate_st::TZDTimeZoneID_st:m_bCBFAbbrChangeExt

4.4.10 IANA Time Zone Database Version Element

Encodes the IANA Time Zone Database tzdata source files release year and release letter.

The IANA Time Zone Database Version Element shall be delimited with V ((V)ersion) followed by a four digit year and single character release letter.

Delimiter	IANA tzdata files release year	IANA tzdata files release letter
V	four digit year (YYYY)	one lower-case character

Example fragments:

```
V2021a
V2022d
```

Example

```
D2024-03-10T03:00:00m000U-04Zamerica/new_yorkAedtV2021aL27S01t01a02cMuX
```

CCF delimiter characters and encoding characters are explicitly defined in CCTLib/CCF.h.

The corresponding CBF members are
CBFLocalDate_st::m_TZDTimeZone_st.m_unTzDataReleaseYear
CBFLocalDate_st::m_TZDTimeZone_st.m_unTzDataReleaseLetter

See TzDatabaseApi.h, TZDDataRelease_st
TZDTimeZone_st
See CBF.h, CBFLocalDate_st::m_TZDTimeZone_st
See CCcf.cpp, CCcf::SetUTCOffsetAndZoneFromCCbf()
CCcf::ParseTzVersionSetCCbf()

4.4.11 Leap-seconds Element

Encodes the leap-second value. The leap-second value is TAI-UTC minus the initial 10s calibration offset between TAI and UTC. *The minimum leap-seconds value is 0 at and before the UTC1972 origin.*

The TOD Count Mode Element describes the method of leap-second introduction resulting in the YMDhms sequence. See TOD Count Mode Element.

If the TOD Count Mode Element value is TOD_NONE or TOD_24HOUR_DAY_DATE the leap-seconds Element shall not appear.

The leap-seconds Element shall be delimited with L ((L)eap) followed by a variable 2-n digit leap-second value. If the current date is a positive leap-second Day a plus sign shall be appended. If the current date is a negative leap-second Day a minus sign shall be appended. If the current time point lies at or within a leap-second an asterisk shall be appended.

Delimiter	Leap-second value	if IsLeapSecond	if IsLeapSecondDay (positive Leap-second)	if IsLeapSecondDay (negative Leap-second)
L	2-n digits	append * (asterisk)	append + (plus)	append - (minus)

Example fragments:
`L25` - TAI-UTC offset minus 10s initial calibration
`L25+` - Is Leap-second Day (positive leap-second)
`L25*` - Is Leap-second
`L25-` - Is Leap-second Day (negative leap-second)

Examples

```
------ 2015 Leap-second Day with TOD_LEAPSECOND_UTC_UTC mode ------
1) second before leap-second day    - not LS day -    L25
2) first second of leap-second day    is  LS day - +  L25+
3) second before leap-second          is  LS day - +  L25+
4) leap-second                        is  LS     - *  L25*
5) second after leap-second           is  LS day - +  L26+
6) last second of leap-second day     is  LS day - +  L26+
7) second after leap-second day       not LS day -    L26

1) D2015-06-29T23:59:59U-04Zamerica/new_yorkAedtV2021aL25Ss+01cMuX
2) D2015-06-30T00:00:00U-04Zamerica/new_yorkAedtV2021aL25+Ss+01cMuX
3) D2015-06-30T19:59:59U-04Zamerica/new_yorkAedtV2021aL25+Ss+01cMuX
4) D2015-06-30T19:59:60U-04Zamerica/new_yorkAedtV2021aL25*Ss+01cMuX
5) D2015-06-30T20:00:00U-04Zamerica/new_yorkAedtV2021aL26+Ss+01cMuX
6) D2015-06-30T23:59:59U-04Zamerica/new_yorkAedtV2021aL26+Ss+01cMuX
7) D2015-07-01T00:00:00U-04Zamerica/new_yorkAedtV2021aL26Ss+01cMuX
```

CCF delimiter characters and encoding characters are explicitly defined in CCTLib/CCF.h.

The corresponding CBF members are

```
CBFLocalDate_st::m_DateTaiUtc_st.m_nLeapsecsHigh;    // Positive leap-seconds
CBFLocalDate_st::m_DateTaiUtc_st.m_nLeapsecsLow;     // Positive leap-seconds
CBFLocalDate_st::m_DateTaiUtc_st.m_nLeapsecsNegHigh; // Negative Leap-seconds
CBFLocalDate_st::m_DateTaiUtc_st.m_lLeapsecsNegLow;  // Negative Leap-seconds
CBFLocalDate_st::m_bIsLeapSecond;         // is Leap-second
CBFLocalDate_st::m_bIsLeapSecondDay;      // is Leap-second day
CBFLocalDate_st::m_bIsLeapSecondNegative; // Leap-second is negative
```

See TaiUtcApi.h, DateTaiUtc_st
See CBF.h, `CBFLocalDate_st::m_DateTaiUtc_st`
`CBFLocalDate_st::m_bIsLeapSecond`
`CBFLocalDate_st::m_bIsLeapSecondDay`
`CBFLocalDate_st::m_bIsLeapSecondNegative`
See CCcf.cpp, `CCcf::SetTAIUTCFromCCbf()`
`CCcf::ParseTAIUTCSetCCbf()`

4.4.12 Daylight Saving Time (DST) Element

Encodes Daylight Saving Time (DST) parameters if applicable.

The DST element shall *not* appear if DST bias equals zero except when DST Transition Day sub-fields may be applicable.

The DST element shall be a variable length string delimited with upper-case "S" (daylight (S)aving) followed by appropriate sub-fields.

Delimiter	Daylight Saving Time
S	DST Bias if applicable and DST Transition Day if applicable

Note that CCT does not have a “tm_isdst” flag, as in traditional Posix-time and TzDb struct tm. Instead, only the DST Bias and DST Transition Day with DST Bias Change and DST Transition Change Time-of-day sub-fields are needed. This then supports unusual situations such as “double summertime” , DST bias and changes not equal to 1-hour. If traditional “tm_isdst” is needed an application may interrogate a CCT timestamp; if DST Bias is not zero, tm_isdst = true.

See CCcf.cpp, CCcf::SetDstMetadataFromCCbf()
CCcf::ParseDstSetCCbf()

4.4.12.1 Daylight Saving Time (DST) Bias Sub-field

Encodes the value of the Daylight Saving Time (DST) offset (called DST Bias) if DST is in effect.

If the DST bias amount is positive the value shall *not* include a sign indicator
If the DST bias amount is negative the value shall be preceded by "-".

A two digit hour shall always be present.
If minutes is non-zero a two digit minute shall be present separated from hours by a ":" (colon).
If seconds is non-zero a two digit seconds shall be present separated from minutes by a ":" (colon).

DST Bias	DST Bias value	
+ or -	two digit hour	all cases
	":" two digit minute	If minutes is non-zero
	":" two digit second	If seconds is non-zero

Corresponding CBF member CBFDstBias_st
See CBF.h, CBFDstBias_st

Example fragments:

```
S01c
S01:15c
S01:30:01c
S-01:30:01c
```

Example

```
D2024-04-00T00:00:00m000U-04Zamerica/new_yorkAedtV2021aL27S01cMuX
```

4.4.12.2 DST Transition Day Sub-fields

If the day has a DST transition the DST Bias element shall appear (its value may be zero bias) together with the DST Transition Day sub-fields

4.4.12.2.1 DST Bias Change Sub-field

Indicates the value of the DST change.
Shall be delimited by lower case “t” followed by and hms indicator. (“t” for “transition”)

A two digit hour shall always be present.
If minutes is non-zero a two digit minute shall be present separated from hours by a ":" (colon).
If seconds is non-zero a two digit seconds shall be present separated from minutes by a ":" (colon).

two digit hour	all cases
":" two digit minute	If minutes is non-zero
":" two digit second	If seconds is non-zero

Corresponding CBF member is
CBFDstTransDay_st:: m_nDstBiasChangeLow:16/ m_nDstBiasChangeHigh:2
See CBF.h, CBFDstTransDay_st

4.4.12.2.2 DST Transition Change Time-of-day Sub-field

Indicates the time-of-day of the DST transition with respect to local time.

Shall be delimited by lower case “a” followed by an hms indicator. (“a” for “at”)

A two digit hour shall always be present.
If minutes is non-zero a two digit minute shall be present separated from hours by a ":" (colon).
If seconds is non-zero a two digit seconds shall be present separated from minutes by a ":" (colon).

two digit hour	all cases
":" two digit minute	If minutes is non-zero
":" two digit second	If seconds is non-zero

Corresponding CBF member is CBFDst_st::m_ulDSTChangeTime

Example fragments

```
S01t01a02c
S01t-01a02c
S00t-01a02c
S-01t01a01c
S00t02a00:01c
S02t-02a00:01c
S02t-01a04:31:19c
S01t-01a24c
S00t00:00:01a23:59:58c
```

Examples

D2024-03-09T23:59:59U-05Zamerica/new_yorkAestV2021aL27MuX
D2024-03-10T01:59:59U-05Zamerica/new_yorkAestV2021aL27S00t01a02cMuX
D2024-03-10T03:00:00U-04Zamerica/new_yorkAedtV2021aL27S01t01a02cMuX
D2024-03-11T00:00:00U-04Zamerica/new_yorkAedtV2021aL27S01cMuX

D2024-11-02T23:59:59U-04Zamerica/new_yorkAedtV2021aL27S01cMuX
D2024-11-03T01:59:59U-04Zamerica/new_yorkAedtV2021aL27S01t-01a02cMu
D2024-11-03T01:00:00U-05Zamerica/new_yorkAestV2021aL27S00t-01a02cMuX
D2024-11-04T00:00:00U-05Zamerica/new_yorkAestV2021aL27MuX

D2024-03-30T23:59:59U+00Zeurope/dublinAgmtV2021aL27S-01cMuX
D2024-03-31T00:59:59U+00Zeurope/dublinAgmtV2021aL27S-01t01a01cMuX
D2024-03-31T02:00:00U+01Zeurope/dublinAistV2021aL27S00t01a01cMuX
D2024-04-01T00:00:00U+01Zeurope/dublinAistV2021aL27MuX

D2024-10-26T23:59:59U+01Zeurope/dublinAistV2021aL27MuX
D2024-10-27T01:59:59U+01Zeurope/dublinAistV2021aL27S00t-01a02cMuX
D2024-10-27T01:00:00U+00Zeurope/dublinAgmtV2021aL27S-01t-01a02cMuX
D2024-10-28T00:00:00U+00Zeurope/dublinAgmtV2021aL27S-01cMuX

4.4.12.3 DST Count Mode Sub-field

Encodes the DST Count Mode in effect.

CCT supports two DST Count Modes, either "conventional", where the DST transition occurs at the typical time-of-day given by tzdb rules, or "uninterrupted", where the DST transition occurs at the end of the day. See Common Calendar Local Timescales, Daylight Saving Time (DST) Count Mode..

If the Count Mode is "conventional" (DSTCOUNTMODE_CONVENTIONAL) a lower-case "c" shall be appended.

If the Count Mode is "uninterrupted" (DSTCOUNTMODE_UNINTERRUPTED) a lower-case "c" shall be appended.

Character Encoding	CBF enumeration CBFDstCountMode_et
c	DSTCOUNTMODE_CONVENTIONAL
u	DSTCOUNTMODE_UNINTERRUPTED

Corresponding CBF member CBFDst_st::m_eDSTCountMode
See CBF.h, CBFDstCountMode_et

Examples:

```
------------------- DST Onset Transition Day --------------------
1) second before Onset day     DST is not in effect - w  Swc
```

```
2) first second of Onset day   DST is not in effect - w  Swo02+01c
3) second before Onset         DST is not in effect - w  Swo02+01c
4) Onset transition            DST is     in effect - s  Sso02+01c
5) last second of Onset day    DST is     in effect - s  Sso02+01c
6) second after Onset Day      DST is     in effect - s  Ss+01c

1) D2015-03-07T23:59:59U-05Zamerica/new_yorkAestV2021aL25SwcMuX
2) D2015-03-08T00:00:00U-05Zamerica/new_yorkAestV2021aL25Swo02+01cMuX
3) D2015-03-08T01:59:59U-05Zamerica/new_yorkAestV2021aL25Swo02+01cMuX
4) D2015-03-08T03:00:00U-04Zamerica/new_yorkAedtV2021aL25Sso02+01cMuX
5) D2015-03-08T23:59:59U-04Zamerica/new_yorkAedtV2021aL25Sso02+01cMuX
6) D2015-03-09T00:00:00U-04Zamerica/new_yorkAedtV2021aL25Ss+01cMuX

------------------- DST Retreat Transition Day -------------------
1) second before Retreat day   DST is     in effect - s  Ss+01c
2) first second of Retreat day DST is     in effect - s  Ssr02+01c
3) second before Retreat       DST is     in effect - s  Ssr02+01c
4) Retreat transition          DST is not in effect - w  Swr02+01c
5) last second of Retreat day  DST is not in effect - w  Swr02+01c
6) second after Retreat Day    DST is not in effect - w  Swc

1) D2015-10-31T23:59:59U-04Zamerica/new_yorkAedtV2021aL26Ss+01cMuX
2) D2015-11-01T00:00:00U-04Zamerica/new_yorkAedtV2021aL26Ssr02+01cMuX
3) D2015-11-01T01:59:59U-04Zamerica/new_yorkAedtV2021aL26Ssr02+01cMuX
4) D2015-11-01T01:00:00U-05Zamerica/new_yorkAestV2021aL26Swr02+01cMuX
5) D2015-11-01T23:59:59U-05Zamerica/new_yorkAestV2021aL26Swr02+01cMuX
6) D2015-11-02T00:00:00U-05Zamerica/new_yorkAestV2021aL26SwcMuX
```

4.5 Assembly and Order

A CCF string shall begin with one of D (Date), T (Time), E (Event), I (Interval), or P (Period).

Required elements and order of presentation for each of the supported meanings are specified in sections below.

4.5.1 UTC accurate local date and time-of-day

To represent a UTC accurate local date and time time point a CCF shall include elements in the order shown in the following table:

Order	Delimiter	Element	
1	D	Date	required
2	T	Time-of-day	required
3	U	UTC offset	required
4	Z	Zone (time zone)	required
5	A	Posix Abbreviation	required
5	V	Version of Tz Database	required
7	L	TAI-UTC (Leap-seconds)	required
8	S	DST	if applicable
9	M	TOD Count Mode shall be TOD_LEAPSECOND_MIDNIGHT or TOD_LEAPSECOND_UTC_UTC or TOD_LEAPSECOND_UTC_NTP or TOD_LEAPSECOND_UTC_POSIX	required
10	X	terminator	required
	E	Event (24 hour period)	excluded
	I	Interval (24 hour period)	excluded
	P	Period (24 hour period)	excluded

D T U Z A V L S M X	required
E I P	excluded

Corresponding CBF variable states

```
   CBFTime_st::m_bIsInterval = false
   CBFTime_st::m_bLocalDateExt == true
   CBFTime_st::m_b24HourPeriodExt == false
   CBFLocalDate_st::m_eTODMode =
      TOD_LEAPSECOND_MIDNIGHT or
      TOD_LEAPSECOND_UTC_UTC or
      TOD_LEAPSECOND_UTC_NTP or
      TOD_LEAPSECOND_UTC_POSIX
   CBFLocalDate_st::m_bDSTExt = true
   CBFDst_st::m_eDSTCountMode =
      DSTCOUNTMODE_CONVENTIONAL or
      DSTCOUNTMODE_UNINTERRUPTED
```

See Common Calendar Local Timescales, 4.2 Time-of-day (TOD) Count Mode and leap-second Introduction, and 4.1 Daylight Saving Time (DST) Count Mode

4.5.2 UTC accurate local date and time with no relation to the date

To represent a UTC accurate local date and a time with no relation to the date a CCF shall include elements in the order shown in the following table:

Order	Delimiter	Element	
1	D	Date	required
2	T	Time-of-day	required
3	U	UTC offset	required
4	Z	Zone (time zone)	required
5	V	Version of Tz Database	required
6	L	TAI-UTC (Leap-seconds)	required
7	M	TOD Count Mode shall be TOD_NONE	required
8	X	terminator	required
	S	DST	excluded
	E	Event (24 hour period)	excluded
	I	Interval (24 hour period)	excluded
	P	Period (24 hour period)	excluded

D T U Z V L M X	required
S E I P	excluded

Corresponding CBF variable states

```
   CBFTime_st::m_bIsInterval == false
   CBFTime_st::m_bLocalDateExt == true
   CBFTime_st::m_b24HourPeriodExt == false
   CBFLocalDate_st::m_eTODMode == TOD_NONE
   CBFLocalDate_st::m_bDSTExt == false
```

See Common Calendar Local Timescales, 4.2 Time-of-day (TOD) Count Mode

4.5.3 Time point less than 86400s

To represent a time point less than 24 hours (< 86400s) a CCF shall include elements in the order shown in the following table:

Order	Delimiter	Element	
1	T	Time	required
2	X	terminator	required
		all others	excluded

T	required
D E I U Z V L S M	excluded

Indicated time shall be less than 86400 seconds.

Corresponding CBF variable states

```
   CBFTime_st::m_bIsInterval == false
   CBFTime_st::m_bLocalDateExt == false
   CBFTime_st::m_b24HourPeriodExt == false
```

4.5.4 Time point equal or greater than 86400s

To represent a time point 24 hours or greater (>= 86400s) a CCF shall include elements in the order shown in the following table:

Order	Delimiter	Element	
1	E	Event	
2	T	Time	required
3	X	terminator	required
		all others	excluded

E and T	required
D I U Z V L S M	excluded

Corresponding CBF variable states

```
   CBFTime_st::m_bIsInterval == false
   CBFTime_st::m_bLocalDateExt == false
   CBFTime_st::m_b24HourPeriodExt == true
```

4.5.5 Interval less than 86400s

To represent an interval less than 24 hours (< 86400s) a CCF shall include elements in the order shown in the following table:

Order	Delimiter	Element	
1	I	Time	required
2	X	terminator	required
		all others	excluded

T	required
D E P U Z V L S M	excluded

Indicated interval shall be less than 86400 seconds.

Corresponding CBF variable states

```
   CBFTime_st::m_bIsInterval == true
   CBFTime_st::m_bLocalDateExt == false
   CBFTime_st::m_b24HourPeriodExt == false
```

4.5.6 Interval equal or greater than 86400s

To represent an interval 24 hours or greater (>= 86400s) a CCF shall include elements in the order shown in the following table:

Order	Delimiter	Element	
1	P	Period	
2	I	Time	required
3	X	terminator	required
		all others	excluded

P and T	required
D E I U Z V L S M	excluded

Corresponding CBF variable states

```
   CBFTime_st::m_bIsInterval == true
   CBFTime_st::m_bLocalDateExt == false
   CBFTime_st::m_b24HourPeriodExt == true
```

4.6 Geostamp

The Common Calendar Timestamp (CCT) specification may be extended to include geographic coordinates to create a Geostamp. The Geostamp specification was developed in collaboration with Son Voba of Sync-n-Scale to support "tractability provenance".

A Geostamp consists of geographic coordinates and a CCT timestamp. Geostamps are technically accurate, making them suitable for general and legal purposes where time recording is used for tracking and auditing and a wide range of spatial-temporal geographic information systems (4D GIS) applications in machine learning, artificial intelligence, data analytics and blockchain distributed ledgers.

Like CCT, Geostamps can be formed in either a binary or character format. The binary format supports efficient machine interoperability while the character format is human readable making their meaning accessible to those less familiar with the intricacies of timekeeping and geographic representations.

Coordinates are carried in the binary CBF CBFLocation_st structure and reflected in CCF character format in the Location Element field.

4.6.1 Location Element

Encodes geographic coordinates.

CCT supports optional inclusion of location, the geographical coordinates of the system at the moment of timestamp generation. This transforms a CCT timestamp into a GeoStamp.

CCT carries coordinates in the form specified by National Marine Electronics Association (NMEA), NMEA 0183 Interface Standard, GPGGA., GGA Global Positioning System Fix Data. Time, Position and fix related data for a GPS receiver. The NMEA data is translated into the CBFLocation_st structure in the CCT binary CBF format. The CBF data is reflected in the CCF character format in the Location Element field.

The Location Element shall be a variable length string delimited with upper-case "C" ((C)oordinates) followed by appropriate sub-fields

Delimiter	Sub-fields			
	External	Latitude	Longitude	Altitude
C	external source or tzdb defaults	degrees, minutes, micro-minutes	degrees, minutes, micro-minutes	meters, centimeters

CCT supports coordinates from two sources, either input from external source such as GPS, or from Tz Database time zone defaults as given in tzdb zone.tab.

if external source a lower-case "e" ((e)xternal) shall be appended.
if not external source a lower-case "z" ((z)one) shall be appended.

The latitude sub-field shall be delimited by lower-case "t". (la(t)itude) followed by "+" or "-", 2 digits of degrees, 2 digits of seconds, a "." (period) separator, and 6 digits microminutes.

The longitude sub-field shall be delimited by lower-case "g". (lon(g)itude) followed by "+" or "-", 2 digits of degrees, 2 digits of seconds, a "." (period) separator, and 6 digits microminutes.

The altitude sub-field shall be delimited by lower-case "t". (lon(g)itude) followed by "+" or "-", 1-n meters, (period) separator, and 6 digits centimeters.

Corresponding CBF member CBFLocation_st
See TzDatabaseAPI.h, CBFLocation_st
See CCct.h, CCct.cpp class CCct
int CCct::SetLocation(char* psLatitude, char* psLongitude, char* psAltitude);

Example fragment:

```
Czt4042.8500000g-7400.3833330
```

Example:

```
D2015-06-30T19:59:60U-04Zamerica/new_yorkAedtV2021aL25*Ss+01cMuCzt4042.8500000g-7400.3833330X
```

Annex A - CCF Character Set

CCF shall be encoded using the ASCII character set excluding control characters and white space.

Disallowed Characters

Character	Dec	Hex	Name
control characters	0 to 31	(0x00) to (0x1F)	control characters
space	32	(0x20)	space
delete	127	(0x7F)	DEL
128 or higher	128 to 255	(0x80) to (0xFF)	128 or higher

Active Characters

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
!"#$%&'()*+,-./:;<=>?@[]^_`{|}~
```

Character	Dec	Hex	Name
!	33	(0x21)	Exclamation
"	34	(0x22h)	Quote
#	35	(0x23)	Number
$	36	(0x24)	Dollar
%	37	(0x25)	Percent
&	38	(0x26)	Ampersand
'	39	(0x27)	Apostrophe
(	40	(0x28)	Open Parenthesis
)	41	(0x29)	Close Parenthesis
*	42	(0x2A)	Asterisk
+	43	(0x2B)	Plus
,	44	(0x2C)	Comma
-	45	(0x2D)	Hyphen
.	46	(0x2E)	Period
/	47	(0x2F)	Forward Slash
0	48	(0x30)	
1	49	(0x31)	
2	50	(0x32)	
3	51	(0x33)	
4	52	(0x34)	
5	53	(0x35)	
6	54	(0x36)	
7	55	(0x37)	
8	56	(0x38)	
9	57	(0x39)	
:	58	(0x3A)	Colon
;	59	(0x3B)	Semicolon
<	60	(0x3C)	Less than
=	61	(0x3D)	Equal
>	62	(0x3E)	Greater than

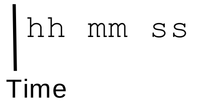
?	63	(0x3F)	Question
@	64	(0x40)	at symbol
A	65	(0x41)	
B	66	(0x42)	
C	67	(0x43)	
D	68	(0x44)	
E	69	(0x45)	
F	70	(0x46)	
G	71	(0x47)	
H	72	(0x48)	
I	73	(0x49)	
J	74	(0x4A)	
K	75	(0x4B)	
L	76	(0x4C)	
M	77	(0x4D)	
N	78	(0x4E)	
O	79	(0x4F)	
P	80	(0x50)	
Q	81	(0x51)	
R	82	(0x52)	
S	83	(0x53)	
T	84	(0x54)	
U	85	(0x55)	
V	86	(0x56)	
W	87	(0x57)	
X	88	(0x58)	
Y	89	(0x59)	
Z	90	(0x5A)	
[	91	(0x5B)	Open Bracket
\	92	(0x5C)	Back Slash
]	93	(0x5D)	Close Bracket
^	94	(0x5E)	Caret
_	95	(0x5F)	Underscore
`	96	(0x60)	Grave Accent
a	97	(0x61)	
b	98	(0x62)	
c	99	(0x63)	
d	100	(0x64)	
e	101	(0x65)	
f	102	(0x66)	
g	103	(0x67)	
h	104	(0x68)	
i	105	(0x69)	
j	106	(0x6A)	
k	107	(0x6B)	
l	108	(0x6C)	
m	109	(0x6D)	
n	110	(0x6E)	
o	111	(0x6F)	

p	112	(0x70)	
q	113	(0x71)	
r	114	(0x72)	
s	115	(0x73)	
t	116	(0x74)	
u	117	(0x75)	
v	118	(0x76)	
w	119	(0x77)	
x	120	(0x78)	
y	121	(0x79)	
z	122	(0x7A)	
{	123	(0x7B)	Open Brace
|	124	(0x7C)	Vertical Bar
}	125	(0x7D)	Close Brace
~	126	(0x7E)	Tilde

Annex B - CCF Example Illustrations

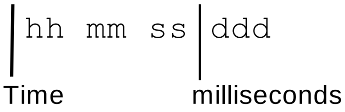
Time point < 86400s in seconds

T01:00:00X



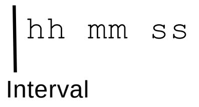
Time point < 86400s in milliseconds

T01:00:00m999X



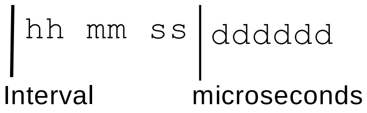
Interval < 86400s in seconds

I00:10:00X



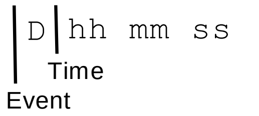
Interval < 86400s in microseconds

I00:10:00u999999X



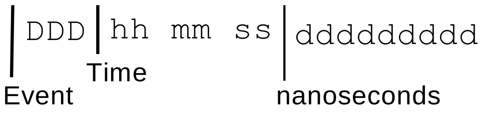
Time point >= 86400s in seconds

E1T12:13:14X



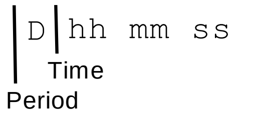
Time point >= 86400s in nanoseconds

E123T01:10:02n999999999X



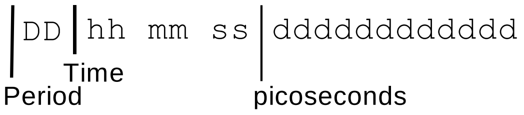
Interval >= 86400s in seconds

P2T22:23:24X



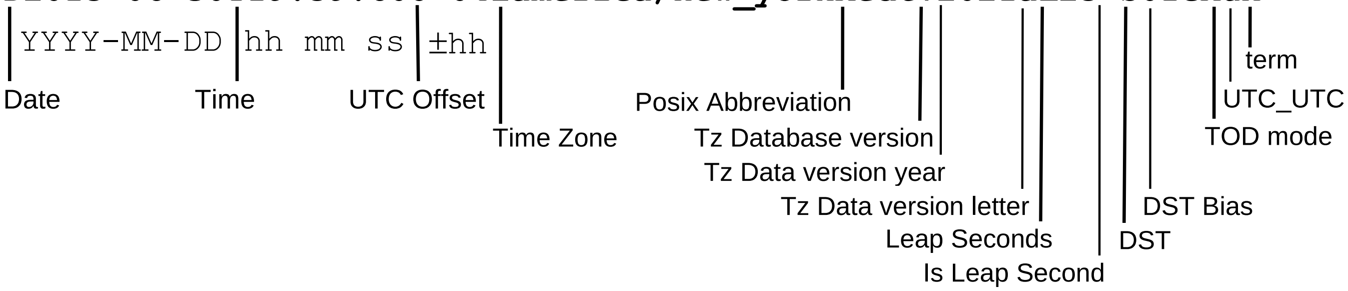
Interval >= 86400s in picoseconds

P12T01:10:02p999999999999X



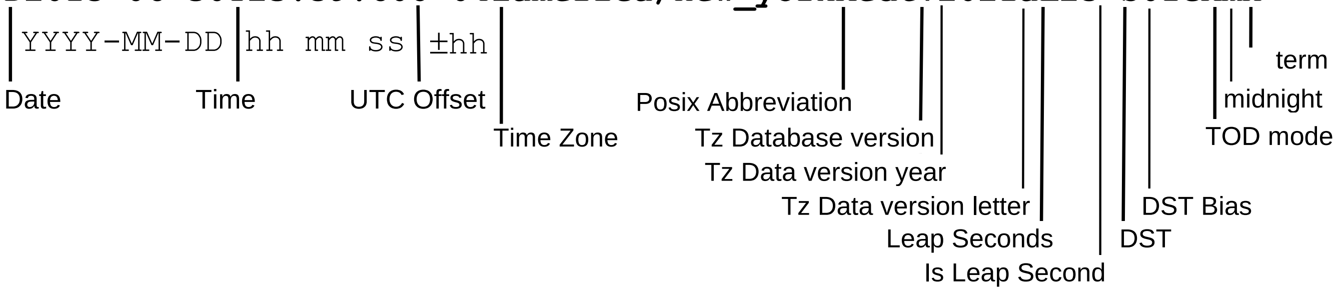
2015 Leap-second in New York with TOD_LEAPSECOND_UTC_UTC Count Mode

D2015-06-30T19:59:60U-04Zamerica/new_yorkAedtV2021aL25*S01cMuX



2015 Leap-second in New York with TOD_LEAPSECOND_MIDNIGHT Count Mode in seconds

D2015-06-30T23:59:60U-04Zamerica/new_yorkAedtV2021aL25*S01cMmX



Annex C - Example Listing from CCT Reference Implementation

```
CCT Version 3.0.0.0 2024-04-23 00:00:00

Time flies when you're having fun!

File out: ..\OutputFiles\CCTOut_BB_TESTSELECTEDCONFIGURATIONSANDSHOWCBFVALUES.txt

======= TestSelectedConfigurationsAndShowCbfValues() =======

------------------------ Common Calendar Character Format (CCF) ----------------
Date       Time     UTC Offset
|          |        |   Zone (Tz time zone name)
|          |        |   |                Abbreviation (Posix name)
|          |        |   |                |   Version (Tz Database release)
|          |        |   |                |   |     Leap-seconds
|          |        |   |                |   |     |   Saving (DST)
|          |        |   |                |   |     |   |DST bias
|          |        |   |                |   |     |   ||  Mode (TOD count mode)
|          |        |   |                |   |     |   ||  | X terminator
D2015-06-30T19:59:60U-04Zamerica/new_yorkAedtV2021aL25*S01cMuX
--------------------------------------------------------------------------------


--------------------------------------
CCF, CCbf member values and CBF binary
--------------------------------------

-- Interval ((I)nterval) < 86400s --
I301:46:38m999X
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    FALSE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   1086398999
CBFTime_st Counter  1086398999
CBF Total size      8 bytes  64 bits
CBF binary interchange bytes as hexidecimal
04 00 17 22 -c1 40 01 00  size 8

-- Interval ((P)eriod) >= 86400s --
P1I00:00:00m000X
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    FALSE
 m_b24HourPeriodExt  TRUE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   0
CBFTime_st Counter  0
CBF24HourPeriod_st::
 m_unDayDuration    1
CBF Total size      12 bytes  96 bits
CBF binary interchange bytes as hexidecimal
04 02 00 00 00 00 01 00 01 00 00 00  size 12

-- Time point ((T)ime) < 86400s --
T301:46:38m999X
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    FALSE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   1086398999
```

```
CBFTime_st Counter  1086398999
CBF Total size      8 bytes  64 bits
CBF binary interchange bytes as hexidecimal
04 00 17 22 -c1 40 00 00  size 8

-- Time point ((E)vent) >= 86400s --
E1T00:00:00m000X
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    FALSE
 m_b24HourPeriodExt  TRUE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   0
CBFTime_st Counter  0
CBF24HourPeriod_st::
 m_unDayDuration    1
CBF Total size      12 bytes  96 bits
CBF binary interchange bytes as hexidecimal
04 02 00 00 00 00 00 00  size 8

-- Second preceding 1972 leap-second in UTC time zone --
D1972-06-30T23:59:59m999U+00Zetc/utcAutcV2021aL00+MuX
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   86399999
CBFTime_st Counter  86399999
CBFLocalDate_st::
 m_l1970DayNumber    911
 m_nLeapsecs         0
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[142] Etc/UTC
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       FALSE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         0
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_NOTAPPLICABLE
 m_bIsLeapSecondDay   TRUE
 m_bIsLeapSecond      FALSE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            FALSE
 m_bDstTransDayExt    FALSE
CBFAbbr::
 utc
CBF Total size      27 bytes  216 bits
CBF binary interchange bytes as hexidecimal
04 01 -ff 5b 26 05 00 00 -8f 03 00 00 00 00 00 -8e 00 31 10 00 00 04 00 02 -f5 -f4 63
size 27

-- 1972 leap-second in UTC time zone --
D1972-06-30T23:59:60m999U+00Zetc/utcAutcV2021aL00*MuX
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   86400999
CBFTime_st Counter  86400999
CBFLocalDate_st::
 m_l1970DayNumber    911
 m_nLeapsecs         0
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[142] Etc/UTC
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       FALSE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         0
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_NOTAPPLICABLE
 m_bIsLeapSecondDay   TRUE
 m_bIsLeapSecond      TRUE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            FALSE
 m_bDstTransDayExt    FALSE
CBFAbbr::
 utc
CBF Total size      27 bytes  216 bits
CBF binary interchange bytes as hexidecimal
04 01 -e7 5f 26 05 00 00 -8f 03 00 00 00 00 00 -8e 00 31 10 00 00 04 00 03 -f5 -f4 63
size 27

-- Second following 1972 leap-second in UTC time zone --
D1972-07-01T00:00:00m000U+00Zetc/utcAutcV2021aL01MuX
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   0
CBFTime_st Counter  0
CBFLocalDate_st::
 m_l1970DayNumber    912
 m_nLeapsecs         1
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[142] Etc/UTC
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       FALSE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         0
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_NOTAPPLICABLE
 m_bIsLeapSecondDay   FALSE
 m_bIsLeapSecond      FALSE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            FALSE
 m_bDstTransDayExt    FALSE
CBFAbbr::
 utc
CBF Total size      27 bytes  216 bits
CBF binary interchange bytes as hexidecimal
04 01 00 00 00 00 00 00 -90 03 00 00 -80 00 00 -8e 00 31 10 00 00 04 00 00 -f5 -f4 63
size 27

-- Second preceding 1972 leap-second in New York time zone --
D1972-06-30T23:19:59m999U-04Zamerica/new_yorkAedtV2021aL01+S01cMuX
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   84000999
CBFTime_st Counter  84000999
CBFLocalDate_st::
 m_l1970DayNumber    911
 m_nLeapsecs         1
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[255] America/New_York
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       FALSE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         -18000
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_CONVENTIONAL
 m_bIsLeapSecondDay   TRUE
 m_bIsLeapSecond      FALSE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            TRUE
 m_bDstTransDayExt    FALSE
CBFAbbr::
 edt
CBFDstBias_st::
 m_eDSTBias         3600
CBF Total size      29 bytes  232 bits
CBF binary interchange bytes as hexidecimal
04 01 -e7 -c0 01 05 00 00 -8f 03 00 00 -80 00 00 -ff 00 31 10 -b0 -b9 55 00 02 10 0e -
e5 -e4 74  size 29

-- 1972 leap-second in New York time zone --
D1972-06-30T19:59:60m999U-04Zamerica/new_yorkAedtV2021aL00*S01cMuX
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   72000999
CBFTime_st Counter  72000999
CBFLocalDate_st::
 m_l1970DayNumber    911
 m_nLeapsecs         0
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[255] America/New_York
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       FALSE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         -18000
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_CONVENTIONAL
 m_bIsLeapSecondDay   TRUE
 m_bIsLeapSecond      TRUE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            TRUE
 m_bDstTransDayExt    FALSE
CBFAbbr::
 edt
CBFDstBias_st::
 m_eDSTBias         3600
CBF Total size      29 bytes  232 bits
CBF binary interchange bytes as hexidecimal
04 01 -e7 -a5 4a 04 00 00 -8f 03 00 00 00 00 00 -ff 00 31 10 -b0 -b9 55 00 03 10 0e -
e5 -e4 74  size 29

-- Second following 1972 leap-second in New York time zone --
```

```
D1972-07-01T00:00:00m000U-04Zamerica/new_yorkAedtV2021aL01S01cMuX
CBFTime_st::
 m_eRateEnumeration CLOCK_3
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_35
 m_ulCounterLow32   0
CBFTime_st Counter  0
CBFLocalDate_st::
 m_l1970DayNumber    912
 m_nLeapsecs         1
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[255] America/New_York
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       FALSE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         -18000
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_CONVENTIONAL
 m_bIsLeapSecondDay   FALSE
 m_bIsLeapSecond      FALSE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            TRUE
 m_bDstTransDayExt    FALSE
CBFAbbr::
 edt
CBFDstBias_st::
 m_eDSTBias         3600
CBF Total size      29 bytes  232 bits
CBF binary interchange bytes as hexidecimal
04 01 00 00 00 00 00 00 -90 03 00 00 -80 00 00 -ff 00 31 10 -b0 -b9 55 00 00 10 0e -e5
-e4 74  size 29

-- Nanosecond preceding 2016 DST Onset in New York time zone --
D2016-03-13T01:59:59n999999999U-05Zamerica/new_yorkAestV2021aL26S00t01a02cMuX
CBFTime_st::
 m_eRateEnumeration CLOCK_9
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_48
 m_ulCounterLow32   1634811903
CBFCounterHigh16_st::
 m_unCounterHigh16  1676
CBFTime_st Counter  7199999999999
CBFLocalDate_st::
 m_l1970DayNumber    16873
 m_nLeapsecs         26
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[255] America/New_York
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       FALSE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         -18000
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_CONVENTIONAL
 m_bIsLeapSecondDay   FALSE
 m_bIsLeapSecond      FALSE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            TRUE
 m_bDstTransDayExt    TRUE
CBFAbbr::
 est
CBFDstBias_st::
 m_eDSTBias         0
CBFDstTransDay_st::
 m_ulDSTTransTime   7200
 m_lDSTBiasChange   3600
CBF Total size      31 bytes  248 bits
CBF binary interchange bytes as hexidecimal
07 05 -ff 3f 71 61 00 00 -8c 06 -e9 41 00 00 00 0d 00 -ff 00 31 10 -b0 -b9 -d5 00 00
00 00 20 1c 10 0e 00 -e5 -f3 74  size 36

-- Nanosecond at 2016 DST Onset in New York time zone with Location--
D2016-03-13T03:00:00n000000000U-
04Zamerica/new_yorkAedtV2021aL26S01t01a02cMuCzt+4042+51g-07400+23X
CBFTime_st::
 m_eRateEnumeration CLOCK_9
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_48
 m_ulCounterLow32   2452217856
CBFCounterHigh16_st::
 m_unCounterHigh16  2514
CBFTime_st Counter  10800000000000
CBFLocalDate_st::
 m_l1970DayNumber    16873
 m_nLeapsecs         26
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[255] America/New_York
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         -18000
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_CONVENTIONAL
 m_bIsLeapSecondDay   FALSE
 m_bIsLeapSecond      FALSE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            TRUE
 m_bDstTransDayExt    TRUE
CBFAbbr::
 edt
CBFDstBias_st::
 m_eDSTBias         3600
CBFDstTransDay_st::
 m_ulDSTTransTime   7200
 m_lDSTBiasChange   3600
CBFLocation_st::
m_i9Lat_Deg       40
m_i7Lat_Min       42
m_i21Lat_uMin     51
m_i9Lgn_Deg       -74
m_i7Lgn_Min       0
m_i21Lng_uMin     23
m_iAlt_cm         16777215
m_bSourceIsExtern 0
m_bIsValidLat     1
m_bIsValidLng     1
m_bIsValidAlt     0
CBF Total size      45 bytes  360 bits
CBF binary interchange bytes as hexidecimal
07 05 00 -e0 29 -92 00 00 -d2 09 -e9 41 00 00 00 0d 00 -ff -80 31 10 -b0 -b9 -d5 00 00
10 0e 20 1c 10 0e 00 -e5 -e4 74 33 00 00 05 17 00 -c0 36 -ff -ff -ff 00 2a 03  size 50

-- Nanosecond following 2016 DST Onset in New York time zone with Location--
```

```
D2016-03-13T03:00:00n000000001U-
04Zamerica/new_yorkAedtV2021aL26S01t01a02cMuCzt+4042+51g-07400+23X
CBFTime_st::
 m_eRateEnumeration CLOCK_9
 m_bLocalDateExt    TRUE
 m_b24HourPeriodExt  FALSE
 m_bCounterSign     positive
 m_eCounterSize     COUNTERSIZE_48
 m_ulCounterLow32   2452217857
CBFCounterHigh16_st::
 m_unCounterHigh16  2514
CBFTime_st Counter  10800000000001
CBFLocalDate_st::
 m_l1970DayNumber    16873
 m_nLeapsecs         26
 m_TZDTimeZoneID_st.m_unZoneIdx    idx[255] America/New_York
 m_TZDTimeZoneID_st.m_unTzDataReleaseYear   49
 m_TZDTimeZoneID_st.m_unTzDataReleaseLetter 0
 m_TZDTimeZoneID_st.m_bCBFLocationExt       TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrExt           TRUE
 m_TZDTimeZoneID_st.m_bCBFAbbrChangeExt     FALSE
 m_lUTCOffset         -18000
 m_eTODMode           TOD_LEAPSECOND_UTC_UTC
 m_eDstMode      DSTCOUNTMODE_CONVENTIONAL
 m_bIsLeapSecondDay   FALSE
 m_bIsLeapSecond      FALSE
 m_bIsLeapSecondNegative   FALSE
 m_bUtcShiftExt       FALSE
 m_bDstBiasExt            TRUE
 m_bDstTransDayExt    TRUE
CBFAbbr::
 edt
CBFDstBias_st::
 m_eDSTBias         3600
CBFDstTransDay_st::
 m_ulDSTTransTime   7200
 m_lDSTBiasChange   3600
CBFLocation_st::
m_i9Lat_Deg       40
m_i7Lat_Min       42
m_i21Lat_uMin     51
m_i9Lgn_Deg       -74
m_i7Lgn_Min       0
m_i21Lng_uMin     23
m_iAlt_cm         16777215
m_bSourceIsExtern 0
m_bIsValidLat     1
m_bIsValidLng     1
m_bIsValidAlt     0
CBF Total size      45 bytes  360 bits
CBF binary interchange bytes as hexidecimal
07 05 01 -e0 29 -92 00 00 -d2 09 -e9 41 00 00 00 0d 00 -ff -80 31 10 -b0 -b9 -d5 00 00
10 0e 20 1c 10 0e 00 -e5 -e4 74 33 00 00 05 17 00 -c0 36 -ff -ff -ff 00 2a 03  size 50


Your time is up.
```