Common Calendar TAI-UTC API

Leap-second Metadata Interface Specification

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

TAI-UTC, the integral second difference between TAI and UTC, is at the foundation of modern civil timekeeping. Obtaining and maintaining this information is critical to accurate timekeeping but no formal mechanism for automatic access to this metadata has been adopted. The TAI-UTC API addresses this obvious missing link in timekeeping technologies.

The IERS issues Bulletin C "*either to announce a time step in UTC or to confirm that there will be no time step at the next possible date*". Bulletin C was originally intended as a human-readable publication requiring human operator intervention in a timekeeping system to make appropriate UTC adjustments.

Today the IERS publishes Bulletin C in several forms including text and XML versions (http://www.iers.org/IERS/EN/DataProducts/EarthOrientationData/eop.html). They also provide the Leap-second WebService (http://hpiers.obspm.fr/eop-pc/index.php?index=webservice). One IERS publication, Leap_Second_History.dat (https://hpiers.obspm.fr/eoppc/bul/bulc/Leap_Second_History.dat) provides a concise tabulation of the previous Leap-second introductions including the state of the current Bulletin C. See Supplementary Material, Annex A - IERS Leap-second History File

All these IERS resources are helpful but none provide a comprehensive and uniformly formatted version of the leap-second information suitable for use in an automated dissemination mechanism.

The TAI-UTC API provides methods for automatic dissemination and acquisition of TAI-UTC metadata to fill the obvious missing link between UTC time dissemination and the TAI timescale. TAI-UTC API defines data types and required operations for uniform treatment of the TAI-UTC (Leap-seconds) history, announcement, and the expiration date of valid leap-second information. The API's goal is to provide uniform timekeeping results through any technology employed for leap-second metadata dissemination and consistent application of the data to timekeeping systems. It is applicable to existing timekeeping technologies and timescales and crucial to any implementation of Common Calendar.

The Common Calendar specifications are intended to be used as generic APIs suitable for implementation on many platforms. The specifications are written largely in terms of c/c++, adopting the c/c++ language syntax for data types and methods to avoid potential ambiguity. CCT "(or "Common Calendar Timescales") is a c/c++ reference implementation of the Common Calendar specifications. The specifications reference the CCT code directly to show unambiguous logic and facilitate implementation.

2 Scope

Specifies a data type for common representation of MJD and TAI-UTC (Leap-second) values as provided by IERS Leap_Second_History.dat, construction of a leap-seconds table, and TAI-UTC Server and Client for communicating the leap-second table to client applications.

3 Normative References

BIPM The International System of Units (SI) 8th edition 2006 (commonly called the SI Brochure)
http://www.bipm.org/en/publications/si-brochure/

BIPM JCGM 200:2012, International vocabulary of metrology – Basic and general concepts and associated terms (VIM)
http://www.bipm.org/en/publications/guides/vim.html

IERS Conventions (2010), (IERS Technical Note No. 36)
https://www.iers.org/IERS/EN/Publications/TechnicalNotes/tn36.html?nn=94912

Recommendation ITU-R TF.457-2, Use Of The Modified Julian Date By The Standard-Frequency And Time-Signal Services
http://www.itu.int/rec/R-REC-TF.457-2-199710-I

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

Common Calendar Date and Time Terms and Definitions
https://common-calendar.org/Common_Calendar_Specification/Common_Calendar_Date_and_Time_Terms_and_Definitions_V38_2023_02_24.pdf

4 TAI-UTC API - Leap-second Metadata Interface Specification

TAI-UTC data format is based on a packed binary pair consisting of a 23-bit day number, a 12-bit positive leap-second value and a 12 bit negative leap-second value. This provides a day and leap-second counter in a 6 bytes, 48-bit data structure with a range of approximately -22967 to 22967 years[1]. With this

[1] TAI-UTC data format is said to support a range of "3000 years". A rough approximation of the range of a 24-bit day counter (MAX 8388607) is 8388607 / 365.24 = ~22967.38 years. The number of Leap Seconds that may occur is unknown and there are many methods of estimates in the literature. The DateTaiUtc_st uses a large bit size of 15-bits to provide a large range for for positive leap-seconds, `16383 MAX,` and negative leap-seconds, -16384 MIN.

compact binary data representation each TAI-UTC value is small and it is then possible to construct a list of the TAI-UTC history (a leap-seconds Table) of reasonable size.

4.1 Origin

The TAI-UTC API origin shall be day zero = UTC1972 (1972-01-01 00:00:10 (TAI) = 1972-01-01T00:00:00 (UTC)). This corresponds with MJD 41317 as given by the IERS in https://hpiers.obspm.fr/iers/bul/bulc/Leap_Second_History.dat.

See *Annex A IERS Leap-second History File*.

4.2 Use of MJD

IERS Leap_Second_History.dat supplies the date of leap-second introductions in the form of Modified Julian Date (MJD). The IERS Leap_Second_History.dat data shall be interpreted by TAI-UTC API as:

- MJD 41317 = UTC1972 (1972-01-01 00:00:10 (TAI) = 1972-01-01T00:00:00 (UTC)).
- MJD 41317 time-of-day seconds = 0 (zero) = midnight.
- Leap-second values = TAI-UTC minus the 10s initial calibration offset.

Note that the origin of CCT timestamp formats and processes is UTC1970, coincident with the Posix time "the Epoch" as used by Time Zone Database.
UTC1970 day 0 = MJD 40587 = UTC1970 (1970-01-01 00:00:10 (TAI) = 1970-01-01T00:00:00 (UTC))
This is exactly 2 years (730 days or 63072000 seconds) before UTC1972

4.3 Leap-second Introduction

There is some uncertainty in the UTC specifications regarding exactly when the TAI-UTC value is to update with respect to the UTC date boundary. ITU-R Recommendation TF.460-6[2] makes it clear the leap-second occurs as the last second of a day. BIPM Circular T[3] and IERS Bulletin C[4] indicate the TAI-UTC value increments at the midnight rollover, that is, immediately *after* the leap-second. However, there is no clear statement regarding this fact; the relationship is only implied by the values in BIPM Circular T and IERS Bulletin C. Common Calendar TAI-UTC API shall treat this implied relationship as fact:

A leap-second shall be introduced in the YMDhms representation of UTC as the last second of the day preceding the UTC date indicated by IERS Bulletin C. A positive leap-second shall immediately follow the second labeled YYYY-MM-DDT23:29:59(UTC) and be labeled YYYY-MM-DDT23:29:60(UTC). A negative leap-second shall immediately follow the second labeled YYYY-MM-DDT23:29:58(UTC) and the label YYYY-MM-DDT23:29:59(UTC) is omitted. The leap-second value shall change immediately following the leap-second, coincident with the UTC YMDhms midnight following the leap-second

See *Annex B Leap-second Introduction.*

4.4 TAI-UTC API Data Structure

The TAI-UTC API data element shall be:

```
typedef struct DateTaiUtc_st // 7 bytes, 52-bit
{
signed long m_l1970DayNumber:24;  // signed zero-based 86400-second days
                                  // since 1970-01-01T00:00:00 (UTC)
                                  // ((2^24)/2)-1 = 8388607 MAX
                                  // 8388607 / 365.24 = ~22967.38 years
                                  // (2^24)/2)* -1 = -8388608 MIN
                                  // -8388608 / 365.24 = ~-22967.38 years
signed long m_lLeapsecsNegLow:8;  // Negative Leap-seconds low word
signed short m_nLeapsecsNegHigh:7; // Negative leap-seconds high word
                                   // Negative TAI-UTC minus initial 10s
                                   // ((2^15)/2)* -1 = -16384 MIN
signed short m_nLeapsecs:9;        // Positive leap-seconds
signed char m_nLeapsecsLow:6;      // Positive leap-seconds
                                   // TAI-UTC minus initial 10s
                                   // ((2^15)/2)-1 = 16383 MAX
signed char m_nReserved:2;
```

[2] Recommendation ITU-R TF.460-6 (02/02), Standard-Frequency and Time-Signal Emissions
[3] BUREAU INTERNATIONAL DES POIDS ET MESURES (BIPM) Circular T
[4] INTERNATIONAL EARTH ROTATION AND REFERENCE SYSTEMS SERVICE (IERS), Bulletin C

```
} DateTaiUtc_st;
```

TAI-UTC API data element

Day Number Days since UTC1970	**Positive Leap-seconds TAI-UTC minus Initial 10s**	**Negative Leap-seconds TAI-UTC minus Initial 10s**

See *Annex E CCT Reference Implementation struct DateTaiUtc_st Declaration*.

4.5 Leap-second Table

An list of TAI-UTC API data elements is used to construct a table of TAI-UTC history corresponding to the IERS Leap_Second_History.dat information. An additional TAI-UTC API data element, called EXPIRATION, shall be appended as the last element of the list to signal the expiration date of the validity of the last TAI-UTC value.

The next-to-last element shall hold the most recently announced leap-second and the TAI-UTC value of EXPIRATION is the same as the TAI-UTC value in that next to last element. When the IERS announces a leap-second the announced values are inserted as a new element in the next-to-last position and the values of the EXPIRATION element are updated.

The expiration date can be obtained from the IERS Leap_Second_History.dat file. It includes a line in the comment section stating the file's "expiration" date, for example: `#  File expires on 28 June 2018`. It is not an official obligation of IERS to provide this expiration date. For purposes of the TAI-UTC API the expiration date is taken to be a declaration by IERS that no leap-second will be declared before that date.

See *Annex D TAI-UTC API Leap-second Table Listing*

5 TAI-UTC Server and Client

TAI-UTC API defines operations and methods for a TAI-UTC Server and Client to provide a uniform method of automatically obtaining leap-second metadata.

A TAI-UTC API metadata server would provide connection and RPC calls to functions defined by the TAI-UTC API by whatever technology and mechanism in use, for examples: SOAP, REST, Service provided by an NTPd-type server, DNS[5], Block Chain.

A client makes connection to the server and make the defined RPC calls to this server.

The CCT reference implementation demonstrates a rudimentary TAI-UTC API metadata server and client This demo code does not attempt to replicate the client/server connection and RPC mechanisms for simplicity and because they are outside the scope of the TAI-UTC API itself. As a c++ demo it simply implements the TAI-UTC API functions as methods of class CTaiUtcServer and CTaiUtcClient. The pointer to class CTaiUtcServer is simply passed to the class CTaiUtcClient which makes calls to its methods.

See *Annex F CCT Use of TAI-UTC Sever and Client*.

5.1 TAI-UTC Server

The TAI-UTC Server constructs a TAI-UTC leap-second table from data acquired from IERS LeapSecondHistory.dat and provides methods for a TAI-UTC Client to obtain the leap-second metadata.

See `CCT\CCTTaiUtcApiLib, CTaiUtcServer.n, CTaiUtcServer.cpp,`
```
   class CTaiUtcServer
```

5.1.1 ParseAndPopulateFromIERSLeapSecondHistoryDat

Parse IERS Leap_Second_History.dat file and populate a TAI-UTC leap-second table.

[5] See Poul-Henning Kamp's DNS implementation: https://github.com/bsdphk/DUT1_DNS

This is not an official method of the API, but encapsulates the operations necessary for the server to construct its copy of the TAI-UTC API leap-second table. [The CCT reference implementation currently fakes parsing the IERS (hard coded).]

```
int CTaiUtcServer::ParseAndPopulateFromIERSLeapSecondHistoryDat();
```

5.1.2 GetTaiUtcTableSize
Get the number of elements in the table.

int CTaiUtcServer::GetTaiUtcTableSize(int* piTaiUtcTableSize);

5.1.3 GetDateTaiUtcAtPosition
Get the TAI-UTC data element at index position

```
int CTaiUtcServer::GetDateTaiUtcAtPosition(int iIdx, DateTaiUtc_st*
                                                     pDateTaiUtc_st);
```

5.1.4 GetFirstDateTaiUtc
Get the first (zeroth) TAI-UTC data element

```
int CTaiUtcServer::GetFirstDateTaiUtc(DateTaiUtc_st* pDateTaiUtc_st);
```

5.1.5 GetNextDateTaiUtc
Get next TAI-UTC data element

```
int CTaiUtcServer::GetNextDateTaiUtc(DateTaiUtc_st* pDateTaiUtc_st);
```

5.1.6 GetMostRecentLeapSecond
Get most recent leap-second TAI-UTC data element (the next-to-last element of the table)

```
int CTaiUtcServer::GetMostRecentLeapSecond(DateTaiUtc_st* pDateTaiUtc_st);
```

5.1.7 GetExpiration
Get the EXPIRATION TAI-UTC data element (the last element of the table)

```
int CTaiUtcServer::GetExpiration(DateTaiUtc_st* pDateTaiUtc_st);
```

5.2 TAI-UTC Client
A TAI-UTC Client implementation makes RPC calls across some connection mechanism to acquire the leap-second metadata from a TaiUtc Server.

See `CCT\CCTTaiUtcApiLib, CTaiUtcClient.h, CTaiUtcClient.cpp,`
`class CTaiUtcClient`

5.2.1 GetTaiUtcTableSize
Get the number of elements in the table

```
int CTaiUtcClient::GetTaiUtcTableSize(int* piTaiUtcTableSize);
```

5.2.2 GetDateTaiUtcAtPosition
Get the TAI-UTC data element at index position

```
int CTaiUtcClient::GetDateTaiUtcAtPosition(int iIdx, DateTaiUtc_st*
                                                     pDateTaiUtc_st);
```

5.2.3 GetFirstDateTaiUtc
Get the first (zeroth) TAI-UTC data element

```
int CTaiUtcClient::GetFirstDateTaiUtc(DateTaiUtc_st* pDateTaiUtc_st);
```

5.2.4 GetNextDateTaiUtc
Get next TAI-UTC data element

```
int CTaiUtcClient::GetNextDateTaiUtc(DateTaiUtc_st* pDateTaiUtc_st);
```

5.2.5 GetMostRecentLeapSecond
Get most recent leap-second TAI-UTC data element (the next-to-last element of the table)

```
int CTaiUtcClient::GetMostRecentLeapSecond(DateTaiUtc_st* pDateTaiUtc_st);
```

5.2.6 GetExpiration
Get the EXPIRATION TAI-UTC data element (the last element of the table)

```
int CTaiUtcClient::GetExpiration(DateTaiUtc_st* pDateTaiUtc_st);
```

5.2.7 GetTableFromServer
Get a copy of the TAI-UTC leap-second table from the TAI-UTC API Server.

```
int CTaiUtcClient::GetTableFromServer(CTaiUtcServer* pCTaiUtcServer)
```

Annex A IERS Leap-second History File

The IERS makes the leap-second history available as text files at:

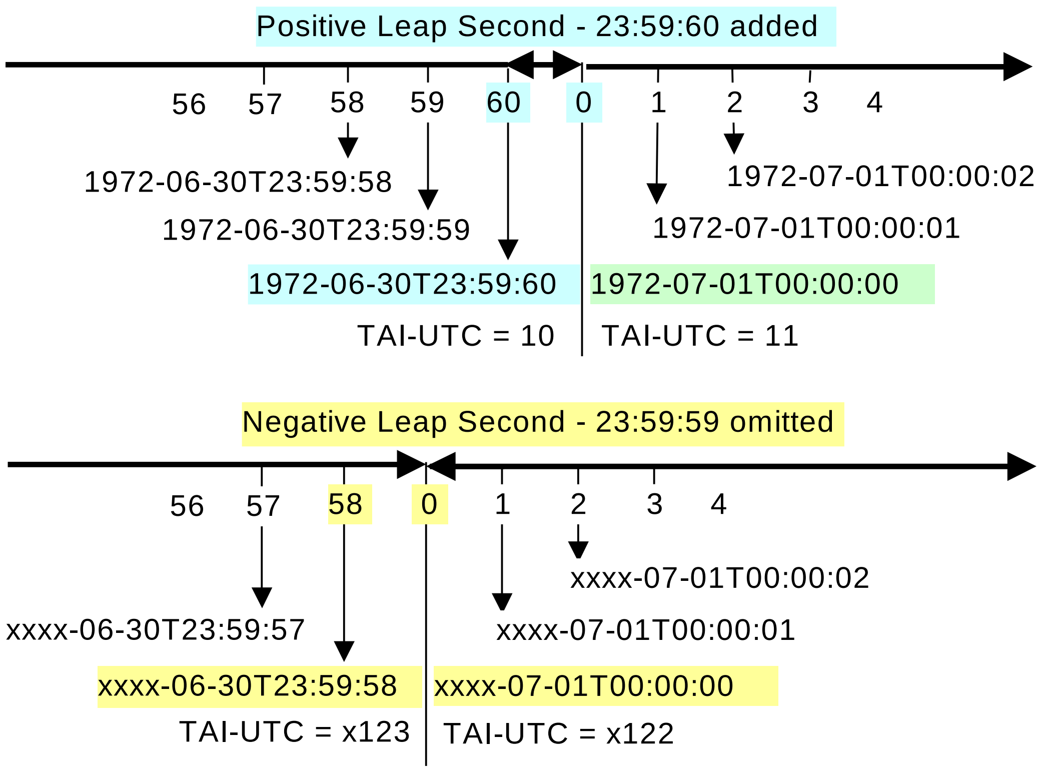
https://hpiers.obspm.fr/iers/bul/bulc/Leap_Second.dat
https://hpiers.obspm.fr/eoppc/bul/bulc/Leap_Second_History.dat

Sample retrieved 2017-08-30:

```
#  Value of TAI-UTC in second valid beetween the initial value until
#  the epoch given on the next line. The last line reads that NO
#  leap-second was introduced since the corresponding date
#  Updated through IERS Bulletin C52 issued in July 2016
#
#
#  File expires on 28 June 2018
#
#
#    MJD        Date        TAI-UTC (s)
#           day month year
#    ---    --------------   ------
#
    41317.0    1  1 1972       10
    41499.0    1  7 1972       11
    41683.0    1  1 1973       12
    42048.0    1  1 1974       13
    42413.0    1  1 1975       14
    42778.0    1  1 1976       15
    43144.0    1  1 1977       16
    43509.0    1  1 1978       17
    43874.0    1  1 1979       18
    44239.0    1  1 1980       19
    44786.0    1  7 1981       20
    45151.0    1  7 1982       21
    45516.0    1  7 1983       22
    46247.0    1  7 1985       23
    47161.0    1  1 1988       24
    47892.0    1  1 1990       25
    48257.0    1  1 1991       26
    48804.0    1  7 1992       27
    49169.0    1  7 1993       28
    49534.0    1  7 1994       29
    50083.0    1  1 1996       30
    50630.0    1  7 1997       31
    51179.0    1  1 1999       32
    53736.0    1  1 2006       33
    54832.0    1  1 2009       34
    56109.0    1  7 2012       35
    57204.0    1  7 2015       36
    57754.0    1  1 2017       37
```

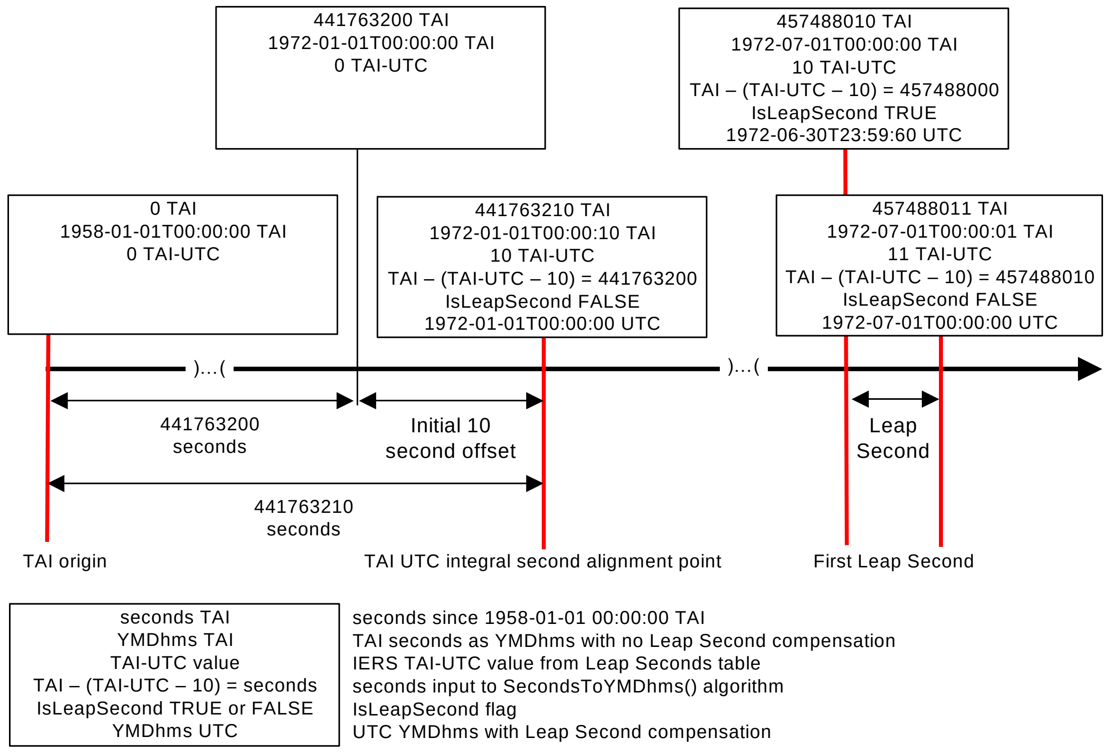
Annex B Leap-second Introduction

Illustration - Leap-second Introduction and TAI-UTC Value Update



Annex C TAI and UTC Time Points

Illustration: Informative TAI and UTC Time Points



Annex D TAI-UTC API Leap-second Table Listing

Listing of TAI-UTC API leap-second table shows the values in effect on the date of this document's publication including the EXPIRATION entry in the last position

```
1970day     0 Leap-seconds  0 // 1970-01-01 00:00:00 MJD 40587
1970day   730 Leap-seconds  0 // 1972-01-01 00:00:00 MJD 41317
1970day   912 Leap-seconds  1 // 1972-07-01 00:00:01 MJD 41499
1970day  1096 Leap-seconds  2 // 1973-01-01 00:00:02 MJD 41683
1970day  1461 Leap-seconds  3 // 1974-01-01 00:00:03 MJD 42048
1970day  1826 Leap-seconds  4 // 1975-01-01 00:00:04 MJD 42413
1970day  2191 Leap-seconds  5 // 1976-01-01 00:00:05 MJD 42778
1970day  2557 Leap-seconds  6 // 1977-01-01 00:00:06 MJD 43144
1970day  2922 Leap-seconds  7 // 1978-01-01 00:00:07 MJD 43509
1970day  3287 Leap-seconds  8 // 1979-01-01 00:00:08 MJD 43874
1970day  3652 Leap-seconds  9 // 1980-01-01 00:00:09 MJD 44239
1970day  4199 Leap-seconds 10 // 1981-07-01 00:00:10 MJD 44786
1970day  4564 Leap-seconds 11 // 1982-07-01 00:00:11 MJD 45151
1970day  4929 Leap-seconds 12 // 1983-07-01 00:00:12 MJD 45516
1970day  5660 Leap-seconds 13 // 1985-07-01 00:00:13 MJD 46247
1970day  6574 Leap-seconds 14 // 1988-01-01 00:00:14 MJD 47161
1970day  7305 Leap-seconds 15 // 1990-01-01 00:00:15 MJD 47892
1970day  7670 Leap-seconds 16 // 1991-01-01 00:00:16 MJD 48257
1970day  8217 Leap-seconds 17 // 1992-07-01 00:00:17 MJD 48804
1970day  8582 Leap-seconds 18 // 1993-07-01 00:00:18 MJD 49169
1970day  8947 Leap-seconds 19 // 1994-07-01 00:00:19 MJD 49534
1970day  9496 Leap-seconds 20 // 1996-01-01 00:00:20 MJD 50083
1970day 10043 Leap-seconds 21 // 1997-07-01 00:00:21 MJD 50630
1970day 10592 Leap-seconds 22 // 1999-01-01 00:00:22 MJD 51179
1970day 13149 Leap-seconds 23 // 2006-01-01 00:00:23 MJD 53736
1970day 14245 Leap-seconds 24 // 2009-01-01 00:00:24 MJD 54832
1970day 15522 Leap-seconds 25 // 2012-07-01 00:00:25 MJD 56109
1970day 16617 Leap-seconds 26 // 2015-07-01 00:00:26 MJD 57204
1970day 17167 Leap-seconds 27 // 2017-01-01 00:00:27 MJD 57754
1970day 18989 Leap-seconds 27 // 2021-12-28 00:00:27 MJD 59576
```

Annex E CCT Implementation struct DateTaiUtc_st Declaration

Extract from CCT\CCTTaiUtcApiLib\TaiUtcApi.h

```
#define OFFSET_MJD_TO_1970_DAYS__40587 40587

typedef struct DateTaiUtc_st // 7 bytes, 52-bit
{
signed long m_l1970DayNumber:24;   // signed zero-based 86400-second days
                                   // since 1970-01-01T00:00:00 (UTC)
                                   // ((2^24)/2)-1 = 8388607 MAX
                                   // 8388607 / 365.24 = ~22967.38 years
                                   // (2^24)/2)* -1 = -8388608 MIN
                                   // -8388608 / 365.24 = ~-22967.38 years
signed long m_lLeapsecsNegLow:8;   // Negative Leap-seconds low word
signed short m_nLeapsecsNegHigh:7; // Negative leap-seconds high word
                                   // Negative TAI-UTC minus initial 10s
                                   // ((2^15)/2)* -1 = -16384 MIN
signed short m_nLeapsecs:9;        // Positive leap-seconds
signed char m_nLeapsecsLow:6;      // Positive leap-seconds
                                   // TAI-UTC minus initial 10s
                                   // ((2^15)/2)-1 = 16383 MAX
signed char m_nReserved:2;
} DateTaiUtc_st;
```

Annex F CCT Implementation Use of TAI-UTC Sever and Client

An implementation of the TAI-UTC Sever and Client is illustrated near the top of `CCT\CCTDemoConsole\CCTDemoTests.cpp, main()`. It constructs a server and a client which obtains the leap-seconds table from the sever and the client's copy of the leap-second table is used throughout the CCT demo tests.

See `CCT\CCTDemoConsole\CCTDemoConsole.cpp`

```
       main(..)
// TAI-UTC metadata server implementing TAI-UTC API
CTaiUtcServer CTaiUtcServerX;
// obtain TAI-UTC data from IERS to populate CTaiUtcServer
// in this example, fake parsing the IERS Leap_Second_History.Dat file
// see CCTTaiUtcApiLib, CTaiUtcServer.h
CTaiUtcServerX.ParseAndPopulateFromIERSLeapSecondHistoryDat();

// TAI-UTC metadata client implementing TAI-UTC API
CTaiUtcClient CTaiUtcClientX;
// CTaiUtcClient supplies Leap-second information
// TaiUtcClient connects to server and obtains data via TAI-UTC API calls
// see CCTTaiUtcApiLib, CTaiUtcClient.h
CTaiUtcClientX.GetTableFromServer(&CTaiUtcServerX);
```

See `CCT\CCTDemoConsole\CCTDemoTests.cpp`

```
   TestCTaiUtcTablePrintf(&Test_stX);
```