
Workgroup: Calendaring Extensions Working Group
Internet-Draft: 01
:
Obsoletes: 3339 (if approved)
Published: 7 January 2021
Intended Status: Standards Track
Expires: 11 July 2021
Author: U. Sharma
 Igalia, S.L.

Date and Time on the Internet: Timestamps with additional information

Abstract

This document defines a date and time format for use in Internet protocols for representation of dates and times using the proleptic Gregorian calendar, with optional extensions representing additional information including a time zone and a calendaring system.

Status of This Memo

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

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

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

This Internet-Draft will expire on 11 July 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

- 1. Introduction
- 2. Definitions
- 3. Two Digit Years
- 4. Local Time
 - 4.1. Coordinated Universal Time (UTC)
 - 4.2. Local Offsets
 - 4.3. Unknown Local Offset Convention
 - 4.4. Unqualified Local Time
- 5. Date and Time format
 - 5.1. Ordering
 - 5.2. Human Readability
 - 5.3. Rarely Used Options
 - 5.4. Redundant Information
 - 5.5. Simplicity
 - 5.6. Internet Date/Time Format
 - 5.7. Restrictions
 - 5.8. Examples
- 6. Security Considerations
- 7. Normative references
- 8. Bibliography
- Appendix A. ISO 8601 Collected ABNF
- Appendix B. Day of the Week
- Appendix C. Leap Years
- Appendix D. Leap Seconds
- Author's Address

1. Introduction

Date and time formats cause a lot of confusion and interoperability problems on the Internet. This document addresses many of the problems encountered and makes recommendations to improve consistency and interoperability when representing and using date and time in Internet protocols.

This document includes an extension to an Internet profile of the [\[ISO8601\]](#) standard for representation of dates and times using the proleptic Gregorian calendar alongside any additional information. Any dates and times that are required to be handled in a different calendar system need to include the ID of the calendar which can be handled by the implementations approximately. In the absence of a calendar ID, the proleptic Gregorian calendar system is assumed.

There are many ways in which date and time values might appear in Internet protocols: this document focuses on just one common usage, viz. timestamps for Internet protocol events. This limited consideration has the following consequences:

- All dates and times are assumed to be in the "current era", somewhere between 0000AD and 9999AD.
- All times expressed have a stated relationship (offset) to Coordinated Universal Time (UTC). Certain applications require the presence of a time zone in order to perform scheduling as well as handle Daylight Savings Time transitions properly. In that case, an optional time zone ID may be included.
- Timestamps can express times that occurred before the introduction of UTC. Such timestamps are expressed relative to universal time, using the best available practice at the stated time.
- Date and time expressions indicate an instant in time. Description of time periods, or intervals, is not covered here.

2. Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

UTC Coordinated Universal Time as maintained by the Bureau International des Poids et Mesures (BIPM).

second A basic unit of measurement of time in the International System of Units. It is defined as the duration of 9,192,631,770 cycles of microwave light absorbed or emitted by the hyperfine transition of cesium-133 atoms in their ground state undisturbed by external fields.

minute A period of time of 60 seconds. However, see also the restrictions in section [Section 5.7](#) and [Appendix D](#) for how leap seconds are denoted within minutes.

hour A period of time of 60 minutes.

day A period of time of 24 hours.

leap year In the proleptic Gregorian calendar, a year which has 366 days. A leap year is a year whose number is divisible by four an integral number of times, except that if it is a centennial year (i.e. divisible by one hundred) it shall also be divisible by four hundred an integral number of times.

ABNF Augmented Backus-Naur Form, a format used to represent permissible strings in a protocol or language, as defined in [[RFC2234](#)].

Email Date/Time Format The date/time format used by Internet Mail as defined by [[RFC2822](#)].

Internet Date/Time Format The date/time format defined in section 5 of this document.

Timestamp This term is used in this document to refer to an unambiguous representation of some instant in time.

Z A suffix which, when applied to a time, denotes a UTC offset of 00:00; often spoken "Zulu" from the ICAO phonetic alphabet representation of the letter "Z".

For more information about time scales, see Appendix E of [[RFC1305](#)], Section 3 of [[ISO8601](#)], and the appropriate ITU documents (ITU-R-TF).

3. Two Digit Years

The following requirements are to address the problems of ambiguity of 2-digit years:

- Internet Protocols **MUST** generate four digit years in dates.
- The use of 2-digit years is deprecated. If a 2-digit year is received, it should be accepted **ONLY** if an incorrect interpretation will not cause a protocol or processing failure (e.g. if used only for logging or tracing purposes).
- It is possible that a program using two digit years will represent years after 1999 as three digits. This occurs if the program simply subtracts 1900 from the year and doesn't check the number of digits. Programs wishing to robustly deal with dates generated by such broken software may add 1900 to three digit years.
- It is possible that a program using two digit years will represent years after 1999 as ":0", ":1", ... ":9", ";0", ... This occurs if the program simply subtracts 1900 from the year and adds the decade to the US-ASCII character zero. Programs wishing to robustly deal with dates generated by such broken software should detect non-numeric decades and interpret appropriately.

The problems with two digit years amply demonstrate why all dates and times used in Internet protocols MUST be fully qualified.

4. Local Time

4.1. Coordinated Universal Time (UTC)

Because the daylight saving rules for local time zones are so convoluted and can change based on local law at unpredictable times, true interoperability is best achieved by using Coordinated Universal Time (UTC). This specification by itself does not cater to local time zone rules. However, certain implementations may be expected to. For these situations, a timestamp may additionally include a local time zone that the implementations can take into account.

4.2. Local Offsets

The offset between local time and UTC is often useful information. For example, in electronic mail (RFC2822, [RFC2822](#)) the local offset provides a useful heuristic to determine the probability of a prompt response. Attempts to label local offsets with alphabetic strings have resulted in poor interoperability in the past [RFC1123](#). As a result, RFC2822 [RFC2822](#) has made numeric offsets mandatory.

Numeric offsets are calculated as "local time minus UTC". So the equivalent time in UTC can be determined by subtracting the offset from the local time. For example, 18:50:00-04:00 is the same time as 22:50:00Z. (This example shows negative offsets handled by adding the absolute value of the offset.)

NOTE: Following ISO 8601, numeric offsets represent only time zones that differ from UTC by an integral number of minutes. However, many historical time zones differ from UTC by a non- integral number of minutes. To represent such historical time stamps exactly, applications must convert them to a representable time zone.

4.3. Unknown Local Offset Convention

If the time in UTC is known, but the offset to local time is unknown, this can be represented with an offset of "-00:00". This differs semantically from an offset of "Z" or "+00:00", which imply that UTC is the preferred reference point for the specified time. RFC2822 [RFC2822](#) describes a similar convention for email.

4.4. Unqualified Local Time

A number of devices currently connected to the Internet run their internal clocks in local time and are unaware of UTC. While the Internet does have a tradition of accepting reality when creating specifications, this should not be done at the expense of interoperability. Since interpretation of an unqualified local time zone will fail in approximately 23/24 of the globe, the interoperability problems of unqualified local time are deemed unacceptable for the Internet.

Systems that are configured with a local time, are unaware of the corresponding UTC offset, and depend on time synchronization with other Internet systems, MUST use a mechanism that ensures correct synchronization with UTC. Some suitable mechanisms are:

- Use Network Time Protocol [[RFC1305](#)] to obtain the time in UTC.
- Use another host in the same local time zone as a gateway to the Internet. This host MUST correct unqualified local times that are transmitted to other hosts.
- Prompt the user for the local time zone and daylight saving rule settings.

5. Date and Time format

This section discusses desirable qualities of date and time formats and defines a format that extends the profile of ISO 8601 for use in Internet protocols.

5.1. Ordering

If date and time components are ordered from least precise to most precise, then a useful property is achieved. Assuming that the time zones of the dates and times are the same (e.g., all in UTC), expressed using the same string (e.g., all "Z" or all "+00:00"), all times have the same number of fractional second digits, and they all have the same suffix (or none), then the date and time strings may be sorted as strings (e.g., using the `strcmp()` function in C) and a time-ordered sequence will result. The presence of optional punctuation would violate this characteristic.

5.2. Human Readability

Human readability has proved to be a valuable feature of Internet protocols. Human readable protocols greatly reduce the costs of debugging since telnet often suffices as a test client and network analyzers need not be modified with knowledge of the protocol. On the other hand, human readability sometimes results in interoperability problems. For example, the date format "10/11/1996" is completely unsuitable for global interchange because it is interpreted differently in different countries. In addition, the date format in (RFC822) has resulted in interoperability problems when people assumed any text string was permitted and translated the three letter abbreviations to other languages or substituted date formats which were easier to generate (e.g. the format used by the C function `ctime`). For this reason, a balance must be struck between human readability and interoperability.

Because no date and time format is readable according to the conventions of all countries, Internet clients SHOULD be prepared to transform dates into a display format suitable for the locality. This may include translating UTC to local time as well as converting from the Gregorian calendar to the viewer's preferred calendar.

5.3. Rarely Used Options

A format which includes rarely used options is likely to cause interoperability problems. This is because rarely used options are less likely to be used in alpha or beta testing, so bugs in parsing are less likely to be discovered. Rarely used options should be made mandatory or omitted for the sake of interoperability whenever possible.

5.4. Redundant Information

If a date/time format includes redundant information, that introduces the possibility that the redundant information will not correlate. For example, including the day of the week in a date/time format introduces the possibility that the day of week is incorrect but the date is correct, or vice versa. Since it is not difficult to compute the day of week from a date (see [Appendix B](#)), the day of week should not be included in a date/time format.

5.5. Simplicity

The complete set of date and time formats specified in ISO 8601 [[ISO8601](#)] is quite complex in an attempt to provide multiple representations and partial representations. [Appendix A](#) contains an attempt to translate the complete syntax of ISO 8601 into [[RFC2234](#)]. Internet protocols have somewhat different requirements and simplicity has proved to be an important characteristic. In addition, Internet protocols usually need complete specification of data in order to achieve true interoperability. Therefore, the complete grammar for ISO 8601 is deemed too complex for most Internet protocols.

The following section defines a format that is an extension of a profile of ISO 8601 for use on the Internet. It is a conformant subset of the ISO 8601 extended format with additional information optionally suffixed. Simplicity is achieved by making most fields and punctuation mandatory.

5.6. Internet Date/Time Format

The following extension of a profile of [\[ISO8601\]](#) dates SHOULD be used in new protocols on the Internet. This is specified using the syntax description notation defined in [\[RFC2234\]](#).

```

date-fullyear   = 4DIGIT
date-month      = 2DIGIT ; 01-12
date-mday       = 2DIGIT ; 01-28, 01-29, 01-30, 01-31 based on month/year
time-hour       = 2DIGIT ; 00-23
time-minute     = 2DIGIT ; 00-59
time-second     = 2DIGIT ; 00-58, 00-59, 00-60 based on leap second rules
time-secfrac    = "." 1*DIGIT
time-numoffset  = ("+" / "-") time-hour ":" time-minute
time-offset     = "Z" / time-numoffset

partial-time    = time-hour ":" time-minute ":" time-second [time-secfrac]
full-date       = date-fullyear "-" date-month "-" date-mday
full-time       = partial-time time-offset

timezone-part   = (ALPHA / "." / "_") *13(ALPHA / "." / "_" / "-") ; but
not "." or ".."
timezone-id     = timezone-part *("/" timezone-part)
timezone        = "[" timezone-id "]"
calendar-part   = 3*8(ALPHA / DIGIT)
calendar-id     = calendar-part *("-" calendar-part)
calendar        = "[c=" calendar-id "]"
suffix-part     = "[" 1*ALPHA "=" 1*(ALPHA / DIGIT) "]"
suffix          = [timezone] [calendar] *suffix-part

date-time       = full-date "T" full-time suffix

```

Figure 1

NOTE 1: Per [\[RFC2234\]](#) and ISO8601, the "T" and "Z" characters in this syntax may alternatively be lower case "t" or "z" respectively.

This date/time format may be used in some environments or contexts that distinguish between the upper- and lower-case letters 'A'-'Z' and 'a'-'z' (e.g. XML). Specifications that use this format in such environments MAY further limit the date/time syntax so that the letters 'T' and 'Z' used in the date/time syntax must always be upper case. Applications that generate this format SHOULD use upper case letters.

NOTE 2: ISO 8601 defines date and time separated by "T". Applications using this syntax may choose, for the sake of readability, to specify a full-date and full-time separated by (say) a space character.

5.7. Restrictions

The grammar element `date-mday` represents the day number within the current month. The maximum value varies based on the month and year as follows:

Month Number	Month/Year	Maximum value of date-mday
01	January	31
02	February, normal	28
02	February, leap year	29
03	March	31
04	April	30
05	May	31
06	June	30
07	July	31
08	August	31
09	September	30
10	October	31
11	November	30
12	December	31

Table 1: Days in each month

[Appendix C](#) contains sample C code to determine if a year is a leap year.

The grammar element `time-second` may have the value "60" at the end of months in which a leap second occurs - to date: June (XXXX-06-30T23:59:60Z) or December (XXXX-12-31T23:59:60Z); see [Appendix D](#) for a table of leap seconds. It is also possible for a leap second to be subtracted, at which times the maximum value of `time-second` is "58". At all other times the maximum value of `time-second` is "59". Further, in time zones other than "Z", the leap second point is shifted by the zone offset (so it happens at the same instant around the globe).

Leap seconds cannot be predicted far into the future. The International Earth Rotation Service publishes bulletins (IERS) that announce leap seconds with a few weeks' warning. Applications should not generate timestamps involving inserted leap seconds until after the leap seconds are announced.

Although ISO 8601 permits the hour to be "24", this extension of a profile of ISO 8601 only allows values between "00" and "23" for the hour in order to reduce confusion.

5.8. Examples

Here are some examples of Internet date/time format.

```
1985-04-12T23:20:50.52Z
```

Figure 2

This represents 20 minutes and 50.52 seconds after the 23rd hour of April 12th, 1985 in UTC.

```
1996-12-19T16:39:57-08:00
```

Figure 3

This represents 39 minutes and 57 seconds after the 16th hour of December 19th, 1996 with an offset of -08:00 from UTC (Pacific Standard Time). Note that this is equivalent to 1996-12-20T00:39:57Z in UTC.

```
1996-12-19T16:39:57-08:00[America/Los_Angeles]
```

Figure 4

This represents the exact same instant as the previous example but additionally specifies the human timezone associated with it for timezone aware implementations to take into account.

```
1990-12-31T23:59:60Z
```

Figure 5

This represents the leap second inserted at the end of 1990.

```
1990-12-31T15:59:60-08:00
```

Figure 6

This represents the same leap second in Pacific Standard Time, 8 hours behind UTC.

```
1937-01-01T12:00:27.87+00:20
```

Figure 7

This represents the same instant of time as noon, January 1, 1937, Netherlands time. Standard time in the Netherlands was exactly 19 minutes and 32.13 seconds ahead of UTC by law from 1909-05-01 through 1937-06-30. This time zone cannot be represented exactly using the HH:MM format, and this timestamp uses the closest representable UTC offset.

```
1937-01-01T12:00:27.87+00:20[c=Japanese]
```

Figure 8

This represents the exact same instant as the previous example but additionally specifies the human calendar associated with it for calendar aware implementations to take into account.

```
1937-01-01T12:00:27.87+00:20[foo=bar][baz=bat]
```

Figure 9

This timestamp utilizes the generalized format to declare two additional pieces of information in the suffix that can be interpreted by any compatible implementations and ignored otherwise.

6. Security Considerations

Since the local time zone of a site may be useful for determining a time when systems are less likely to be monitored and might be more susceptible to a security probe, some sites may wish to emit times in UTC only. Others might consider this to be loss of useful functionality at the hands of paranoia.

7. Normative references

- [RFC2822] Resnick, P., Ed., "Internet Message Format", IETF RFC 2822, IETF RFC 2822, DOI 10.17487/RFC2822, April 2001, <<https://www.rfc-editor.org/info/rfc2822>>.
- [RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", IETF RFC 2234, IETF RFC 2234, DOI 10.17487/RFC2234, November 1997, <<https://www.rfc-editor.org/info/rfc2234>>.
- [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts — Application and Support", IETF RFC 1123, IETF RFC 1123, DOI 10.17487/RFC1123, October 1989, <<https://www.rfc-editor.org/info/rfc1123>>.

- [RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", IETF RFC 1305, DOI 10.17487/RFC1305, March 1992, <<https://www.rfc-editor.org/info/rfc1305>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", IETF RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

8. Bibliography

- [ISO8601] ISO, "Data elements and interchange formats", ISO 8601:1988, June 1988, <<https://www.iso.org/standard/15903.html>>.

Appendix A. ISO 8601 Collected ABNF

This information is based on the 1988 version of ISO 8601. There may be some changes in the 2000 revision.

ISO 8601 does not specify a formal grammar for the date and time formats it defines. The following is an attempt to create a formal grammar from ISO 8601. This is informational only and may contain errors. ISO 8601 remains the authoritative reference.

Note that due to ambiguities in ISO 8601, some interpretations had to be made. First, ISO 8601 is not clear if mixtures of basic and extended format are permissible. This grammar permits mixtures. ISO 8601 is not clear on whether an hour of 24 is permissible only if minutes and seconds are 0. This assumes that an hour of 24 is permissible in any context. Restrictions on date-day in section [Section 5.7](#) apply. ISO 8601 states that the "T" may be omitted under some circumstances. This grammar requires the "T" to avoid ambiguity. ISO 8601 also requires (in

section 5.3.1.3) that a decimal fraction be preceded by a "0" if less than unity. Annex B.2 of ISO 8601 gives examples where the decimal fractions are not preceded by a "0". This grammar assumes section 5.3.1.3 is correct and that Annex B.2 is in error.

```

date-century::    = 2DIGIT    ; 00-99
date-decade::    =  DIGIT    ; 0-9
date-subdecade::  =  DIGIT    ; 0-9
date-year::      = date-decade date-subdecade
date-fullyear::   = date-century date-year
date-month::     = 2DIGIT    ; 01-12
date-wday::      =  DIGIT    ; 1-7 ; 1 is Monday, 7 is Sunday
date-mday::      = 2DIGIT    ; 01-28, 01-29, 01-30, 01-31 based on ; month
/year
date-yday::      = 3DIGIT    ; 001-365, 001-366 based on year
date-week::      = 2DIGIT    ; 01-52, 01-53 based on year

datepart-fullyear:: = [date-century] date-year ["-"]
datepart-ptyear::   = "-" [date-subdecade ["-"]]
datepart-wkyear::   = datepart-ptyear / datepart-fullyear

dateopt-century::   = "-" / date-century
dateopt-fullyear::  = "-" / datepart-fullyear
dateopt-year::      = "-" / (date-year ["-"])
dateopt-month::     = "-" / (date-month ["-"])
dateopt-week::      = "-" / (date-week ["-"])

datespec-full::     = datepart-fullyear date-month ["-"] date-mday
datespec-year::     = date-century / dateopt-century date-year
datespec-month::    = "-" dateopt-year date-month ["-"] date-mday
datespec-mday::     = "--" dateopt-month date-mday
datespec-week::     = datepart-wkyear "W"
                    (date-week / dateopt-week date-wday)
datespec-wday::     = "---" date-wday
datespec-yday::     = dateopt-fullyear date-yday

date:: = datespec-full / datespec-year / datespec-month / datespec-
mday / datespec-week / datespec-wday / datespec-yday

```

Figure 10

Time:

```

time-hour::      = 2DIGIT ; 00-24
time-minute::    = 2DIGIT ; 00-59
time-second::    = 2DIGIT ; 00-58, 00-59, 00-60 based on ; leap-second
rules
time-fraction::  = ("," / ".") 1*DIGIT
time-numoffset:: = ("+" / "-") time-hour [":" time-minute]
time-zone::      = "Z" / time-numoffset

timeopt-hour::   = "-" / (time-hour [":"])
timeopt-minute:: = "-" / (time-minute [":"])

timespec-hour::  = time-hour [":" time-minute [":" time-second]]
timespec-minute:: = timeopt-hour time-minute [":" time-second]
timespec-second:: = "-" timeopt-minute time-second
timespec-base::  = timespec-hour / timespec-minute / timespec-second

time::           = timespec-base [time-fraction] [time-zone]

iso-date-time::  = date "T" time

```

Figure 11

Durations:

```

dur-second::     = 1*DIGIT "S"
dur-minute::     = 1*DIGIT "M" [dur-second]
dur-hour::       = 1*DIGIT "H" [dur-minute]
dur-time::       = "T" (dur-hour / dur-minute / dur-second)
dur-day::        = 1*DIGIT "D"
dur-week::       = 1*DIGIT "W"
dur-month::      = 1*DIGIT "M" [dur-day]
dur-year::       = 1*DIGIT "Y" [dur-month]
dur-date::       = (dur-day / dur-month / dur-year) [dur-time]

duration::       = "P" (dur-date / dur-time / dur-week)

```

Figure 12

Periods:

```

period-explicit:: = iso-date-time "/" iso-date-time
period-start::    = iso-date-time "/" duration
period-end::      = duration "/" iso-date-time

period::          = period-explicit / period-start / period-end

```

Figure 13

Appendix B. Day of the Week

The following is a sample C subroutine loosely based on Zeller's Congruence (ZELLER) which may be used to obtain the day of the week for dates on or after 0000-03-01:

```
char *day_of_week(int day, int month, int year)
{
    int cent;
    char *dayofweek[] = {
        "Sunday", "Monday", "Tuesday", "Wednesday",
        "Thursday", "Friday", "Saturday"
    };

    /* adjust months so February is the last one */
    month -= 2;
    if (month < 1) {
        month += 12;
        --year;
    }
    /* split by century */
    cent = year / 100;
    year %= 100;
    return (dayofweek[((26 * month - 2) / 10 + day + year
                      + year / 4 + cent / 4 + 5 * cent) % 7]);
}
```

Figure 14

Appendix C. Leap Years

Here is a sample C subroutine to calculate if a year is a leap year:

```
/* This returns non-zero if year is a leap year. Must use 4 digit
   year.
*/
int leap_year(int year)
{
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}
```

Figure 15

Appendix D. Leap Seconds

Information about leap seconds can be found at the [US Navy Oceanography Portal](#). In particular, it notes that:

The decision to introduce a leap second in UTC is the responsibility of the International Earth Rotation Service (IERS). According to the CCIR Recommendation, first preference is given to the opportunities at the end of December and June, and second preference to those at the end of March and September.

When required, insertion of a leap second occurs as an extra second at the end of a day in UTC, represented by a timestamp of the form YYYY-MM-DDT23:59:60Z. A leap second occurs simultaneously in all time zones, so that time zone relationships are not affected. See section [Section 5.8](#) for some examples of leap second times.

The following table is an excerpt from the table maintained by the United States Naval Observatory. The source data is located at the [US Navy Oceanography Portal](#).

This table shows the date of the leap second, and the difference between the time standard TAI (which isn't adjusted by leap seconds) and UTC after that leap second.

UTC Date	TAI - UTC After Leap Second
1972-06-30	11
1972-12-31	12
1973-12-31	13
1974-12-31	14
1975-12-31	15
1976-12-31	16
1977-12-31	17
1978-12-31	18
1979-12-31	19
1981-06-30	20
1982-06-30	21
1983-06-30	22
1985-06-30	23
1987-12-31	24

UTC Date	TAI - UTC After Leap Second
1989-12-31	25
1990-12-31	26
1992-06-30	27
1993-06-30	28
1994-06-30	29
1995-12-31	30
1997-06-30	31
1998-12-31	32

Table 2: Historic leap seconds

Author's Address

Ujjwal Sharma

Igalia, S.L.

Email: ryzokuken@igalia.com