Independent Submission
Internet-Draft
Intended status: Standards Track
Expires: March 14, 2019

A. Olson

P. Eggert
UCLA
K. Murchison
FastMail

September 10, 2018

The Time Zone Information Format (TZif)
draft-murchison-tzdist-tzif-13

Abstract

This document defines the Time Zone Information Format (TZif) for representing and exchanging time zone information, independent of any particular service or protocol. Two MIME media types for this format are also defined.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Time zone data typically consists of offsets from Universal Time (UT), daylight saving transition rules, one or more local time designations (acronyms or abbreviations), and optional leap second adjustments. One such format for conveying this information is iCalendar [RFC5545]. It is a text-based format used by calendaring and scheduling systems.

This document defines the Time Zone Information Format (TZif). It is a binary format used by most UNIX systems to calculate local time. There is a wide variety of interoperable software [tz-link] capable of generating and reading files in this format.

This specification does not define the source of leap second information, nor does it define the source the time zone data, metadata, identifiers, aliases, localized names, or versions as defined in Section 3 of [RFC7808]. One such source is the IANA-hosted time zone database [RFC6557].

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [1] [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used in this document:

Coordinated Universal Time (UTC): The basis for civil time since 1960. It is approximately equal to mean solar time at the prime meridian (0 degrees longitude).

Daylight Saving Time (DST): The time according to a location's law or practice, adjusted as necessary from standard time. The adjustment may be positive, negative, or zero.

International Atomic Time (TAI): The time standard based on atomic clocks since 1972. It is equal to UTC except without leap second adjustments.

Leap Second Correction (LEAPCORR): The value of TAI - UTC - 10 for timestamps after the first leap second, and zero for timestamps before that. The expression "TAI - UTC - 10" comes from the fact that TAI - UTC was defined to be 10 just prior to the first leap second in 1972, so clocks with leap seconds have a zero LEAPCORR before the first leap second.

Local Time: The time according to a location's current time zone offset from Universal Time.

Standard Time: The time according to a location's law or practice, unadjusted for Daylight Saving Time.

Time Change: A change to civil timekeeping practice. It occurs when one or more of the following happen simultaneously:

1. a change in UT offset

2. a change in whether standard or daylight saving time is in use

3. a change in time zone abbreviation

4. a leap second (i.e., a change in LEAPCORR)

Time Zone Data: The Time Zone Data Distribution Service (TZDIST) [RFC7808] defines "Time zone data" as "data that defines a single

time zone, including an identifier, UTC offset values, DST rules, and other information such as time zone abbreviations." The interchange format defined in this document is one such form of time zone data.

Universal Time (UT): The basis of civil time. This is the principal form of the mean solar time at the prime meridian (0 degrees longitude) for timestamps before UTC was introduced in 1960, and is UTC for timestamps thereafter. Although UT is sometimes called "UTC" or "GMT" in other sources, this specification uses the term "UT" to avoid confusion with UTC or with GMT.

UNIX Time: The time as returned by the C time() function (see Section 3 of the "System Interfaces" Volume [2] of [POSIX]). This is an integer number of seconds since the POSIX Epoch (1970-01-01 00:00:00 UTC) not counting leap seconds. As an extension to POSIX, negative values represent times before the POSIX Epoch, using UT.

UNIX Leap Time: UNIX time plus all preceding leap second corrections. For example, if the first leap second record in a TZif file occurs at 1972-06-30 23:59:60 UTC, the UNIX leap time for the timestamp 1972-07-01 00:00:00 UTC would be 78796801, one greater than the UNIX time for the same timestamp. Similarly, if the second leap second record occurs at 1972-12-31 23:59:60 UTC, its UNIX leap time would be 94694401; the second occurrence accounts for the first leap second. If a TZif file specifies no leap second records, UNIX leap time is equivalent to UNIX time.

Wall Time: The time as shown on a clock set according to a location's law or practice.

3. The Time Zone Information Format (TZif)

The time zone information format begins with a fixed 44-octet header (Section 3.1) followed by a variable-length data block (Section 3.2) using four-octet (32-bit) transition times and leap second occurrences. These 32-bit values are limited to representing times from 1901-12-13 20:45:52 through 2038-01-19 03:14:07 UT.

The TZif header contains a field which specifies the version of the file's format. Version 1 files terminate after the 32-bit data block.

Version 2 and 3 files extend the format by appending a second 44-octet header, another variable-length data block using eight-octet (64-bit) transition times and leap second occurrences, and a variable

length footer (Section 3.3). These 64-bit values can represent times approximately 292 billion years into the past or future.

A TZif file is structured as follows:

```
                    Version 1       Versions 2 & 3
                 +-------------+   +-------------+
                 | Header for  |   | Header for  |
                 |   32-bit    |   |   32-bit    |
                 | Transitions |   | Transitions |
                 +-------------+   +-------------+
                 |  Data with  |   |  Data with  |
                 |   32-bit    |   |   32-bit    |
                 | Transitions |   | Transitions |
                 +-------------+   +-------------+
                                   | Header for  |
                                   |   64-bit    |
                                   | Transitions |
                                   +-------------+
                                   |  Data with  |
                                   |   64-bit    |
                                   | Transitions |
                                   +-------------+
                                   |   Footer    |
                                   +-------------+
```

General Format of TZif Files

NOTE: All multi-octet integer values MUST be stored in network octet order format (high-order octet first, otherwise known as big-endian), with all bits significant. Signed integer values MUST be represented using two's complement.

3.1. TZif Header

The TZif header is structured as follows (the number of octets occupied by a field is shown in parenthesis):

```
       +---------------+---+
       |  magic    (4) |ver|
       +---------------+---+---------------------------------------+
       |           [unused - reserved for future use] (15)         |
       +---------------+---------------+---------------+-----------+
       |  isutcnt  (4) |  isstdcnt (4) |  leapcnt  (4) |
       +---------------+---------------+---------------+
       |  timecnt  (4) |  typecnt  (4) |  charcnt  (4) |
       +---------------+---------------+---------------+
```

TZif Header

The fields of the header are defined as follows:

magic: The four-octet ASCII [RFC0020] sequence "TZif" (0x54 0x5A 0x69 0x66) which identifies the file as utilizing the Time Zone Information Format.

ver(sion): An octet identifying the version of the file's format. The value MUST be one of the following:

NUL (0x00) Version 1 - The file contains only the 32-bit header and data block. Version 1 files MUST NOT contain a 64-bit header, data block, or footer.

'2' (0x32) Version 2 - The file MUST contain the 32-bit header and data block, a 64-bit header and data block, and a footer. The TZ string in the footer (Section 3.3), if nonempty, MUST strictly adhere to the requirements for the TZ environment variable as defined in Section 8.3 of the "Base Definitions" Volume [3] of [POSIX], and MUST encode the POSIX portable character set as ASCII.

'3' (0x33) Version 3 - The file MUST contain the 32-bit header and data block, a 64-bit header and data block, and a footer. The TZ string in the footer (Section 3.3), if nonempty, MUST conform to POSIX requirements with ASCII encoding, except that it MAY use the TZ string extensions described below (Section 3.3.1).

isutcnt: A four-octet unsigned integer specifying the number of UT/ local indicators contained in the data block - MUST either be zero or equal to 'typecnt'.

isstdcnt: A four-octet unsigned integer specifying the number of standard/wall indicators contained in the data block - MUST either be zero or equal to 'typecnt'.

leapcnt: A four-octet unsigned integer specifying the number of leap second records contained in the data block.

timecnt: A four-octet unsigned integer specifying the number of transition times contained in the data block.

typecnt: A four-octet unsigned integer specifying the number of local time type records contained in the data block - MUST NOT be zero. (Although time type 0 is not used in files that have nonempty TZ strings but no transitions, it is nevertheless required because many TZif readers reject files that lack time types.)

charcnt: A four-octet unsigned integer specifying the total number of octets used by the set of time zone designations contained in the data block.

3.2. TZif Data Block

The TZif data block consists of seven variable-length elements, each of which is series of zero or more items. The number of items in each series is determined by the corresponding count field in the header. The total length of each element is calculated by multiplying the number of items by the size of each item. Therefore, implementations that do not wish to parse or use the 32-bit data block can calculate its total length and skip directly to the header of the 64-bit data block.

In the initial data block, time values are 32-bit (TIME_SIZE = 4 octets). In the second data block, present only in version 2 and 3 files, time values are 64-bit (TIME_SIZE = 8 octets).

The data block is structured as follows (the number of octets occupied by a field is shown in parenthesis):

```
        +---------------------------------------------------------+
        |  transition times          (timecnt x TIME_SIZE)        |
        +---------------------------------------------------------+
        |  transition types          (timecnt)                    |
        +---------------------------------------------------------+
        |  local time type records   (typecnt x 6)                |
        +---------------------------------------------------------+
        |  time zone designations    (charcnt)                    |
        +---------------------------------------------------------+
        |  leap second records       (leapcnt x (TIME_SIZE + 4))  |
        +---------------------------------------------------------+
        |  standard/wall indicators  (isstdcnt)                   |
        +---------------------------------------------------------+
        |  UT/local indicators       (isutcnt)                    |
        +---------------------------------------------------------+
```

TZif Data Block

The elements of the data block are defined as follows:

transition times: A series of four- or eight-octet UNIX leap time values sorted in strictly ascending order. Each value is used as a transition time at which the rules for computing local time may change. The number of time values is specified by the 'timecnt' field in the header. Each time value SHOULD be at least -2**59. (-2**59 is the greatest negated power of 2 that predates the Big Bang, and avoiding earlier timestamps works around known TZif reader bugs relating to outlandlishly negative timestamps.)

transition types: A series of one-octet unsigned integers specifying the type of local time of the corresponding transition time. These values serve as indices into the array of local time type records. The number of type indices is specified by the 'timecnt' field in the header. Each type index MUST be in the range [0, 'typecnt').

local time type records: A series of six-octet records specifying a local time type. The number of records is specified by the 'typecnt' field in the header. Each record has the following format:

```
   +---------------+-+-+---+
   |  utoff (4)    |dst|idx|
   +---------------+---+---+
```

utoff: A four-octet signed integer specifying the number of seconds to be added to UT in order to determine local time. The value MUST NOT be -2**31, and SHOULD be in the range

[-89999, 93599] (i.e., its value SHOULD be more than -25 hours and less than 26 hours). (Avoiding -2**31 allows 32-bit clients to negate the value without overflow. Restricting it to [-89999, 93599] allows easy support by implementations that already support the the POSIX-required range [-24:59:59, 25:59:59].)

(is)dst: A one-octet value indicating whether local time should be considered Daylight Savings Time (DST). The value MUST be 0 or 1. A value of one (1) indicates that DST is in effect. A value of zero (0) indicates that standard time in effect.

(desig)idx: A one-octet unsigned integer specifying an index into the series of time zone designation octets, thereby selecting a particular designation string. Each index MUST be in the range [0, 'charcnt'), and MUST index a NUL-terminated (0x00) sequence of octets.

time zone designations: A series of octets constituting an array of NUL-terminated (0x00) time zone designation strings. The total number of octets is specified by the 'charcnt' field in the header. Note that two designations MAY overlap if one is a suffix of the other.

leap second records: A series of eight- or twelve-octet records specifying the corrections that need to be applied to UTC in order to determine TAI. The records are sorted by the occurrence time in strictly ascending order. The number of records is specified by the 'leapcnt' field in the header. Each record has one of the following structures:

32-bit Data Block:

```
+---------------+---------------+
|  occur (4)    |  corr (4)     |
+---------------+---------------+
```

64-bit Data Block:

```
+---------------+---------------+---------------+
|  occur (8)                    |  corr (4)     |
+---------------+---------------+---------------+
```

occur(rence): A four- or eight-octet UNIX leap time value specifying the time at which a leap second correction occurs. The first value, if present, MUST be nonnegative, and each later value MUST be at least 2419199 greater than the previous

value. (This is 28 days' worth of seconds, minus a potential negative leap second.)

corr(ection): A four-octet signed integer specifying the value of LEAPCORR on or after the occurrence. The correction value in the first leap second record, if present, MUST be either one (1) or minus one (-1). The correction values in adjacent leap second records MUST differ by exactly one (1). The value of LEAPCORR is zero for timestamps that occur before the occurrence time in the first leap second record (or for all timestamps if there are no leap second records).

standard/wall indicators: A series of one-octet values indicating whether the transition times associated with local time types were specified as standard time or wall clock time. Each value MUST be 0 or 1. A value of one (1) indicates standard time, and MUST be set to one (1) if the corresponding UT/local indicator is set to one (1). A value of zero (0) indicates wall time. The number of values is specified by the 'isstdcnt' field in the header. If 'isstdcnt' is zero (0), all transition times associated with local time types are assumed to be specified as wall time.

UT/local indicators: A series of one-octet values indicating whether the transition times associated with local time types were specified as UT or local time. Each value MUST be 0 or 1. A value of one (1) indicates UT, and the corresponding standard/wall indicator MUST also be set to one (1). A value of zero (0) indicates local time. The number of values is specified by the 'isutcnt' field in the header. If 'isutcnt' is zero (0), all transition times associated with local time types are assumed to be specified as local time.

The type corresponding to a transition time specifies local time for timestamps starting at the given transition time and continuing up to, but not including, the next transition time. Local time for timestamps before the first transition is specified by the first time type (time type 0). Local time for timestamps on or after the last transition is specified by the TZ string in the footer (Section 3.3) if present and nonempty, and is unspecified otherwise. If there are no transitions, local time for all timestamps is specified by the TZ string in the footer if present and nonempty, and is specified by time type 0 otherwise.

A given pair of standard/wall and UT/local indicators is used to designate whether the corresponding transition time was specified as UT, standard time, or wall clock time. Note that there are only three combinations of the two indicators given that the standard/wall value MUST be one (1) if the UT/local value is one (1). This

information can be useful if the transition times in a TZif file need to be transformed into transitions appropriate for another time zone (e.g. when calculating transition times for a simple POSIX TZ string such as "AKST9AKDT").

In order to eliminate unused space in a TZif file, every nonzero local time type index SHOULD appear at least once in the transition type array. Likewise, every octet in the time zone designations array SHOULD be used by at least one time type record.

3.3. TZif Footer

The TZif footer is structured as follows (the number of octets occupied by a field is shown in parenthesis):

```
                   +---+--------------------+---+
                   | NL|  TZ string (0...)  |NL |
                   +---+--------------------+---+
```

TZif Footer

The elements of the footer are defined as follows:

NL: An ASCII new line character (0x0A).

TZ string: A rule for computing local time changes after the last transition time stored in the 64-bit data block. The string is either empty or uses the expanded format of the "TZ" environment variable as defined in Section 8.3 of the "Base Definitions" Volume [4] of [POSIX] with ASCII encoding, possibly utilizing extensions described below (Section 3.3.1) in version 3 files. If the string is empty, the corresponding information is not available. If the string is nonempty and one or more transitions appear in the 64-bit data, the string MUST be consistent with the last 64-bit transition - i.e., evaluating the TZ string at the time of the last transition should yield the same time type as the time type specified in the last transition. The string MUST NOT contain NUL octets or be NUL-terminated, and SHOULD NOT begin with the ':' (colon) character.

3.3.1. TZ String Extensions

The TZ string in a Version 3 TZif file MAY use the following extensions to POSIX TZ strings. These extensions are described using the terminology of Section 8.3 of the "Base Definitions" Volume [5] of [POSIX].

- The hours part of the transition times may be signed and range from -167 through 167 (-167 <= hh <= 167) instead of the POSIX-required unsigned values from 0 through 24.

  Example: <-03>3<-02>,M3.5.0/-2,M10.5.0/-1
  This represents a time zone that observes daylight saving time from 22:00 on the day before March's last Sunday until 23:00 on the day before October's last Sunday. Standard time is 3 hours west of UT and is abbreviated "-03"; daylight saving time is 2 hours west of UT and is abbreviated "-02".

- DST is considered to be in effect all year if it starts January 1 at 00:00 and ends December 31 at 24:00 plus the difference between daylight saving and standard time, leaving no room for standard time in the calendar.

  Example: EST5EDT,0/0,J365/25
  This represents a time zone that observes daylight saving time all year. It is 4 hours west of UT and is abbreviated "EDT".

4. Interoperability Considerations

These recommended practices should be followed in order to assure interoperability of TZif applications.

- Version 1 files are considered a legacy format and SHOULD NOT be generated, as they do not support transition times after the year 2038.

- Implementations SHOULD generate a version 3 file if TZ string extensions are necessary to accurately model transition times. Otherwise, version 2 files SHOULD be generated.

- The sequence of time changes defined by the 32-bit header and data block SHOULD be a contiguous subsequence of the time changes defined by the 64-bit header and data block, and by the footer. This guideline helps obsolescent version 1 readers agree with current readers about timestamps within the contiguous subsequence. It also lets writers not catering to obsolescent readers use a 'timecnt' of zero in the 32-bit data to save space.

- Time zone designations SHOULD consist of at least three (3) and no more than six (6) ASCII characters from the set of alphanumerics, '-', and '+'. This is for compatibility with POSIX requirements for time zone abbreviations.

- When reading a version 2 or 3 file, implementations SHOULD ignore the 32-bit header and data block except for the purpose of skipping over them.

- Implementations SHOULD calculate the total lengths of the 32- and 64-bit headers and data blocks and compare them against the actual file size, as part of a validity check for the file.

- When a TZif file is used in a MIME message entity it SHOULD be indicated by one of the following media types:

  * "application/tzif-leap" (Section 8.2) to indicate that leap second records are included in the TZif data.

  * "application/tzif" (Section 8.1) to indicate that leap second records are not included in the TZif data; 'leapcnt' in the header(s) MUST be zero (0).

- Common interoperability issues and possible workarounds are described in Appendix A.

5. Use with the Time Zone Data Distribution Service

The Time Zone Data Distribution Service (TZDIST) [RFC7808] is a service that allows reliable, secure, and fast delivery of time zone data and leap second rules to client systems such as calendaring and scheduling applications or operating systems.

A TZDIST service MAY supply time zone data to clients in the Time Zone Information Format. Such a service MUST indicate that it supports this format by including the MIME media type "application/tzif" (Section 8.1) in its "capabilities" response (see Section 5.1 of [RFC7808]). A TZDIST service MAY also include the MIME media type "application/tzif-leap" (Section 8.2) in its "capabilities" response if it is able to generate TZif files containing leap second records. A TZDIST service MUST NOT advertise the "application/tzif-leap" MIME media type without also advertising "application/tzif".

TZDIST clients MUST use the HTTP "Accept" [RFC7231] header field to indicate their preference to receive data in the "application/tzif" and/or "application/tzif-leap" formats.

5.1. Truncating TZif Files

As described in Section 3.9 of [RFC7808], a TZDIST service MAY truncate time zone transition data. A truncated TZif file is valid from its first and up to, but not including, its last 64-bit transition time, if present.

When truncating the start of a TZif file, the service MUST supply in the 64-bit data a first transition time that is the start point of the truncation range. As with untruncated TZif files, time type 0 indicates local time immediately before the start point, and the time type of the first transition indicates local time thereafter.

When truncating the end of a TZif file, the service MUST supply in the 64-bit data a last transition time that is the end point of the truncation range, and MUST supply an empty TZ string. As with untruncated TZif files with empty TZ strings, a truncated TZif file does not indicate local time after the last transition.

All represented information that falls inside the truncation range MUST be the same as that represented by a corresponding untruncated TZif file.

TZDIST clients SHOULD NOT use a truncated TZif file (as described above) to interpret timestamps outside the truncation time range.

5.2. Example

In this example, the client checks the server for the available formats and then requests that the time zone with a specific time zone identifier be returned in Time Zone Information Format.

Note that this example presumes that the time zone context path has been discovered (see [RFC7808] Section 4.2.1) to be "/tzdist".

```
>> Request <<

GET /tzdist/capabilities HTTP/1.1
Host: tz.example.com

>> Response <<

HTTP/1.1 200 OK
Date: Fri, 01 Jun 2018 14:52:23 GMT
Content-Type: application/json; charset="utf-8"
Content-Length: xxxx

{
  "version": 1,

  "info": {
    "primary-source": "IANA:2018e",
    "formats": [
      "text/calendar",
      "application/tzif",
      "application/tzif-leap"
    ],
...
  },
...
}


>> Request <<

GET /tzdist/zones/America%2FNew_York HTTP/1.1
Host: tz.example.com
Accept: application/tzif

>> Response <<

HTTP/1.1 200 OK
Date: Fri, 01 Jun 2018 14:52:24 GMT
Content-Type: application/tzif
Content-Length: xxxx
ETag: "123456789-000-111"

TZif2...[binary data without leap second records]...
EST5EDT,M3.2.0,M11.1.0
```

6. Security Considerations

The Time Zone Information Format contains no executable code and the format does not define any extensibility areas that could be used to store such code.

TZif contains counted arrays of data elements. All counts should be checked when processing TZif objects to guard against references past the end of the object.

TZif provides no confidentiality or integrity protection. Time zone information is normally public and does not call for confidentiality protection. Since time zone information is used in many critical applications, integrity protection may be required, and must be provided externally.

7. Privacy Considerations

None.

8. IANA Considerations

This document defines two MIME [RFC6838] media types for the exchange of data utilizing the Time Zone Information Format.

8.1. application/tzif

Type name: application

Subtype name: tzif

Required parameters: none

Optional parameters: none

Encoding considerations: binary

Security considerations: See Section 6 of this document.

Interoperability considerations: See Section 4 of this document.

Published specification: This specification.

Applications that use this media type: This media type is designed for widespread use by applications that need to use or exchange time zone information, such as the Time Zone Information Compiler (zic) [6] and the GNU C Library [7]. The Time Zone Distribution Service [RFC7808] can directly use this media type.

Fragment identifier considerations: N/A

Additional information:

Magic number(s): The first 4 octets are 0x54, 0x5A, 0x69, 0x66

File extensions(s): N/A

Macintosh file type code(s): N/A

Person & email address to contact for further
information:
Time Zone Database mailing list <tz@iana.org>

Intended usage: COMMON

Restrictions on usage: N/A

Author: See the "Author's Address" section of this document.

Change controller: IETF

8.2. application/tzif-leap

Type name: application

Subtype name: tzif-leap

Required parameters: none

Optional parameters: none

Encoding considerations: binary

Security considerations: See Section 6 of this document.

Interoperability considerations: See Section 4 of this document.

Published specification: This specification.

Applications that use this media type: This media type is designed for widespread use by applications that need to use or exchange time zone information, such as the Time Zone Information Compiler (zic) [8] and the GNU C Library [9]. The Time Zone Distribution Service [RFC7808] can directly use this media type.

Fragment identifier considerations: N/A

Additional information:

Magic number(s): The first 4 octets are 0x54, 0x5A, 0x69, 0x66

File extensions(s): N/A

Macintosh file type code(s): N/A

Person & email address to contact for further information:
Time Zone Database mailing list <tz@iana.org>

Intended usage: COMMON

Restrictions on usage: N/A

Author: See the "Author's Address" section of this document.

Change controller: IETF

9. Acknowledgments

The authors would like to thank the following individuals for contributing their ideas and support for writing this specification: Michael Douglass, Ned Freed, Guy Harris, Eliot Lear, and Alexey Melnikov.

10. References

10.1. Normative References

[POSIX] IEEE, "Standard for Information Technology--Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7", IEEE 1003.1-2017, DOI 10.1109/IEEESTD.2018.8277153, January 2018, <http://pubs.opengroup.org/onlinepubs/9699919799/>.

This standard is also <https://ieeexplore.ieee.org/servlet/opac?punumber=8277151> published by ieeexplore.ieee.org.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7808] Douglass, M. and C. Daboo, "Time Zone Data Distribution Service", RFC 7808, DOI 10.17487/RFC7808, March 2016, <https://www.rfc-editor.org/info/rfc7808>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <https://www.rfc-editor.org/info/rfc5545>.

[RFC6557] Lear, E. and P. Eggert, "Procedures for Maintaining the Time Zone Database", BCP 175, RFC 6557, DOI 10.17487/RFC6557, February 2012, <https://www.rfc-editor.org/info/rfc6557>.

[tz-link] Eggert, P. and A. Olson, "Sources for Time Zone and Daylight Saving Time Data", 2018, <https://www.iana.org/time-zones/repository/tz-link.html>.

10.3. URIs

[1] https://tools.ietf.org/html/bcp14

[2] http://pubs.opengroup.org/onlinepubs/9699919799/functions/time.html

[3] http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap08.html#tag_08_03

[4] http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap08.html#tag_08_03

[5] http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap08.html#tag_08_03

[6] http://man7.org/linux/man-pages/man8/zic.8.html

[7] https://www.gnu.org/software/libc/

[8] http://man7.org/linux/man-pages/man8/zic.8.html

[9] https://www.gnu.org/software/libc/

Appendix A. Common Interoperability Issues

This section documents common problems in implementing this specification. Most of these are problems in generating TZif files for use by readers conforming to predecessors of this specification. The goals of this section are:

1. to help TZif writers output files that avoid common pitfalls in older or buggy TZif readers,

2. to help TZif readers avoid common pitfalls when reading files generated by future TZif writers, and

3. to help any future specification authors see what sort of problems arise when the TZif format is changed.

When new versions of the TZif format have been defined, a design goal has been that a reader can successfully use a TZif file even if the file is of a later TZif version than what the reader was designed for. When complete compatibility was not achieved, an attempt was made to limit glitches to rarely-used timestamps, and to allow simple partial workarounds in writers designed to generate new-version data useful even for older-version readers. This section attempts to document these compatibility issues and workarounds, as well as to document other common bugs in readers.

Interoperability problems with TZif include the following:

o Some readers examine only 32-bit data. As a partial workaround, a writer can output as much 32-bit data as possible. However, a reader should ignore 32-bit data, and should use 64-bit data even if the reader's native timestamps have only 32 bits.

- Some readers designed for version 2 might mishandle timestamps after a version 3 file's last transition, because they cannot parse extensions to POSIX in the TZ-like string. As a partial workaround, a writer can output more transitions than necessary, so that only far-future timestamps are mishandled by version 2 readers.

- Some readers designed for version 2 do not support permanent daylight saving time, e.g., a TZ string "EST5EDT,0/0,J365/25" denoting permanent Eastern Daylight Time (-04). As a partial workaround, a writer can substitute standard time for the next time zone east, e.g., "AST4" for permanent Atlantic Standard Time (-04).

- Some readers ignore the footer, and instead predict future timestamps from the time type of the last transition. As a partial workaround, a writer can output more transitions than necessary.

- Some readers do not use time type 0 for timestamps before the first transition, in that they infer a time type using a heuristic that does not always select time type 0. As a partial workaround, a writer can output a dummy (no-op) first transition at an early time.

- Some readers mishandle timestamps before the first transition that has a timestamp not less than -2**31. Readers that support only 32-bit timestamps are likely to be more prone to this problem, for example, when they process 64-bit transitions only some of which are representable in 32 bits. As a partial workaround, a writer can output a dummy transition at timestamp -2**31.

- Some readers mishandle a transition if its timestamp has the minimum possible signed 64-bit value. Timestamps less than -2**59 are not recommended.

- Some readers mishandle POSIX-style TZ strings that contain '<' or '>'. As a partial workaround, a writer can avoid using '<' or '>' for time zone abbreviations containing only alphabetic characters.

- Many readers mishandle time zone abbreviations that contain non-ASCII characters. These characters are not recommended.

- Some readers may mishandle time zone abbreviations that contain fewer than 3 or more than 6 characters, or that contain ASCII characters other than alphanumerics, '-', and '+'. These abbreviations are not recommended.

- o Some readers mishandle TZif files that specify daylight-saving time UT offsets that are less than the UT offsets for the corresponding standard time. These readers do not support locations like Ireland, which uses the equivalent of the POSIX TZ string "IST-1GMT0,M10.5.0,M3.5.0/1", observing standard time (IST, +01) in summer and daylight saving time (GMT, +00) in winter. As a partial workaround, a writer can output data for the equivalent of the POSIX TZ string "GMT0IST,M3.5.0/1,M10.5.0", thus swapping standard and daylight saving time. Although this workaround misidentifies which part of the year uses daylight saving time, it records UT offsets and time zone abbreviations correctly.

Some interoperability problems are reader bugs that are listed here mostly as warnings to developers of readers.

- o Some readers do not support negative timestamps. Developers of distributed applications should keep this in mind if they need to deal with pre-1970 data.

- o Some readers mishandle timestamps before the first transition that has a nonnegative timestamp. Readers that do not support negative timestamps are likely to be more prone to this problem.

- o Some readers mishandle time zone abbreviations like "-08" that contain '+', '-', or digits.

- o Some readers mishandle UT offsets that are out of the traditional range of -12 through +12 hours, and so do not support locations like Kiritimati that are outside this range.

- o Some readers mishandle UT offsets in the range [-3599, -1] seconds from UT, because they integer-divide the offset by 3600 to get 0 and then display the hour part as "+00".

- o Some readers mishandle UT offsets that are not a multiple of one hour, or of 15 minutes, or of 1 minute.

Appendix B. Change History (To be removed by RFC Editor before publication)

Changes since -12:

- o Added reference to RFC0020.

- o Fully fleshed out application/tzif-leap declaration rather than referencing application/tzif.

- o Added definition for "Leap Second Correction".

- Added external references directly to the relevant sections of POSIX.

Changes since -11:

- Removed text requiring leapcnt by non-zero for application/tzif-leap files.

Changes since -10:

- Removed text mandating all TZDIST features be supported.

- Minor editorial changes.

Changes since -09:

- Removed "Update 7808" from header as this spec doesn't introduce new TZDIST functionality.

- Added text regarding truncation of TZif via TZDIST.

- Expanded what this spec DOESN'T define.

- Added reasons for some of the recommended practices.

- Added common interoperability issues appendix.

- Minor editorial changes.

Changes since -08:

- Clarifying text regarding MIME types.

- Consolidated/referenced duplicate security and interoperability text.

- Switched to 'octets' instead of 'characters' when describing time zone designations.

- Minor editorial changes.

Changes since -07:

- Clarifying text regarding TZ string.

- Added "application/tzif-leap" MIME media type.

- New reference for zic(8) man page.

- Minor editorial changes.

Changes since -06:

- Added definition of UNIX Leap Time and used it to describe transition times and leap second occurrences.
- Moved TZif generation recommendations into discussion of version field.
- Repeated TZif generation recommendations in TZDIST section.
- Rewrote part of the TZ string text.
- Minor editorial changes.

Changes since -05:

- Clarify TAI, leap seconds, some descriptions, and some field values/ranges with text from Paul Eggert.
- Refined MIME declaration based on feedback from Ned Freed.

Changes since -04:

- Edited text discussing timestamps before first and after last transition.
- Specified legal range of time type indices and time zone designation indices.
- Notes that corrections in adjacent leap second records must differ by one.
- Added recommendations to eliminate unused space.
- Minor editorial changes.

Changes since -03:

- Removed definition of GMT.
- Updated definitions of UTC, TAI, and UT
- Switched to using UT rather than UTC.
- Added more text about the use of standard/wall and UT/local indicators.

- o Added Acknowledgments.

- o Minor editorial changes.

Changes since -02:

- o Updated definitions of Standard Time and DST.

- o Added definitions of GMT and UT.

- o Added a definition of Time Zone Data from RFC7808.

- o Removed sentence stating that TZDB is accurate.

- o Added more text for standard/wall and UTC/local indicators and counts.

- o Added text discussing timestamps before first and after last transition.

- o Added more guidance text regarding 32-bit and 64-bit data consistency.

- o Minor editorial changes.

Changes since -01:

- o Renamed "POSIX Time" to "UNIX Time" and noted that values can be negative.

- o Noted that signed values MUST be represented using two's complement.

- o Renamed "POSIX TZ string" to "TZ string" and noted that it can be empty.

- o Moved TZ string extensions into its own subsection with examples.

- o Renamed leap second "epoch" to "occurrence".

- o Editorial changes from Paul Eggert.

Changes since -00:

- o Split TZif format description into a general overview and 3 subsections.

- o Updated Keywords boilerplate.

- Updated POSIX reference.

- Editorial changes from Eliot Lear.

Authors' Addresses

Arthur David Olson

Email: arthurdavidolson@gmail.com

Paul Eggert
University of California, Los Angeles

Email: eggert@cs.ucla.edu

Kenneth Murchison
FastMail US LLC

Email: murch@fastmailteam.com