Internet Engineering Task Force (IETF)
Request for Comments: 8536
Category: Standards Track
ISSN: 2070-1721

A. Olson

P. Eggert
UCLA

K. Murchison
FastMail

February 2019

The Time Zone Information Format (TZif)

Abstract

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

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8536.

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 specifies the widely deployed Time Zone Information Format (TZif). It is a binary format used by most UNIX systems to calculate local time. This format was introduced in the 1980s and has evolved since then into multiple upward-compatible versions. There is a wide variety of interoperable software capable of generating and reading files in this format [tz-link].

This specification does not define the source of the data assembled into a TZif file. 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 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used in this document (see "Sources for Time Zone and Daylight Saving Time Data" [tz-link] for more detailed information about civil timekeeping data and practice):

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, when adjusted as necessary from standard time. The adjustment may be positive or negative, and the amount of adjustment may vary depending on the date and time; the TZif format even allows the adjustment to be zero, although this is not common practice.

International Atomic Time (TAI): The time standard based on atomic clocks since 1972. It is equal to UTC but 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: Civil time for a particular location. Its offset from universal time can depend on the date and time of day.

POSIX Epoch: 1970-01-01 00:00:00 UTC, the basis for absolute timestamps in this document.

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 daylight saving time is in effect
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.

Transition Time: The moment of occurrence of a time change that is not a leap second. It is identified with a signed integer count of UNIX leap time seconds since the POSIX epoch.

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 time() function provided by the C programming language (see Section 3 of the "System Interfaces" volume of [POSIX]). This is an integer number of seconds since the POSIX epoch, 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, it accounts for the first leap second, so the UNIX leap time of 1972-12-31 23:59:60 UTC would be 94694401, and the UNIX leap time of 1973-01-01 00:00:00 UTC would be 94694402. If a TZif file specifies no leap-second records, UNIX leap time is equal to UNIX time.

Wall Time: Another name for local time; short for "wall-clock time".

3. The Time Zone Information Format (TZif)

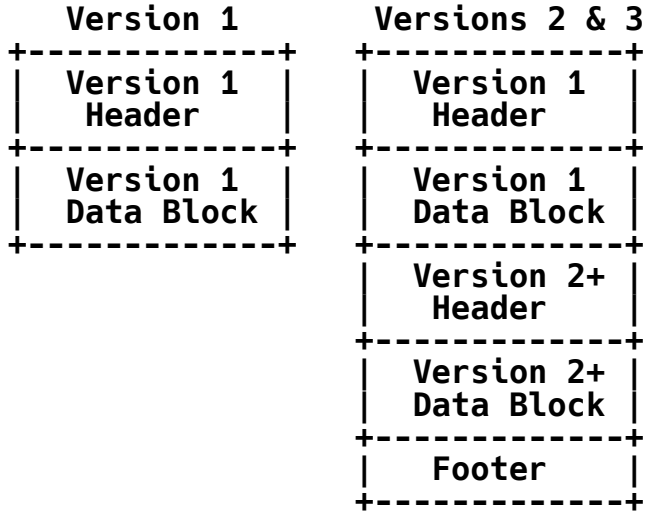
The Time Zone Information Format begins with a fixed 44-octet version 1 header (Section 3.1) containing a field that specifies the version of the file's format. Readers designed for version N can read version N+1 files without too much trouble; data specific to version N+1 either appears after version N data so that earlier-version readers can easily ignore later-version data they are not designed for, or it appears as a minor extension to version N that version N readers are likely to tolerate well.

The version 1 header is followed by a variable-length version 1 data block (Section 3.2) containing four-octet (32-bit) transition times and leap-second occurrences. These 32-bit values are limited to representing time changes from 1901-12-13 20:45:52 through 2038-01-19 03:14:07 UT, and the version 1 header and data block are present only for backward compatibility with obsolescent readers, as discussed in Common Interoperability Issues (Appendix A).

Version 1 files terminate after the version 1 data block. Files from versions 2 and 3 extend the format by appending a second 44-octet version 2+ header, a variable-length version 2+ data block containing 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.

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.

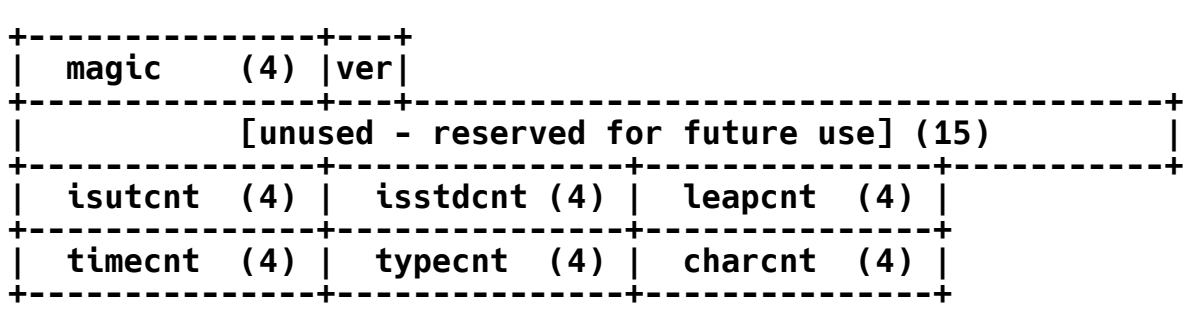
A TZif file is structured as follows:



General Format of TZif Files

3.1. TZif Header

A TZif header is structured as follows (the lengths of multi-octet fields are shown in parentheses):



TZif Header

The fields of the header are defined as follows:

magic: The four-octet ASCII [RFC20] 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 version 1 header and data block. Version 1 files MUST NOT contain a version 2+ header, data block, or footer.

'2' (0x32) Version 2 - The file MUST contain the version 1 header and data block, a version 2+ 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 of [POSIX] and MUST encode the POSIX portable character set as ASCII.

'3' (0x33) Version 3 - The file MUST contain the version 1 header and data block, a version 2+ 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 local time type records convey no useful information in files that have nonempty TZ strings but no transitions, at least one such record is nevertheless required because many TZif readers reject files that have zero 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 - MUST NOT be zero. The count includes the trailing NUL (0x00) octet at the end of the last time zone designation.

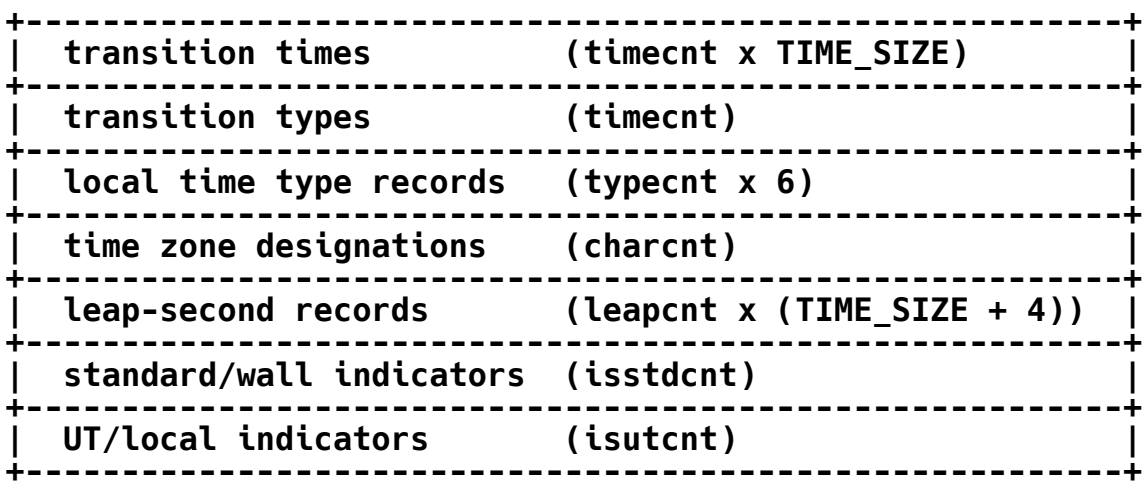
Although the version 1 and 2+ headers have the same format, magic number, and version fields, their count fields may differ, because the version 1 data can be a subset of the version 2+ data.

3.2. TZif Data Block

A TZif data block consists of seven variable-length elements, each of which is a series of 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 version 1 data block can calculate its total length and skip directly to the header of the version 2+ data block.

In the version 1 data block, time values are 32 bits (TIME_SIZE = 4 octets). In the version 2+ data block, present only in version 2 and 3 files, time values are 64 bits (TIME_SIZE = 8 octets).

The data block is structured as follows (the lengths of multi-octet fields are shown in parentheses):



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 outlandishly 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 zero-based 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" - 1].

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 (the lengths of multi-octet fields are shown in parentheses):

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 POSIX-required range [-24:59:59, 25:59:59].

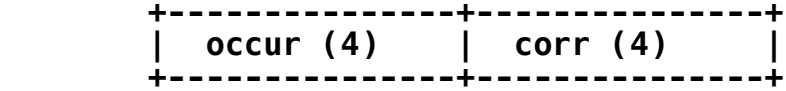
(is)dst: A one-octet value indicating whether local time should be considered Daylight Saving Time (DST). The value MUST be 0 or 1. A value of one (1) indicates that this type of time is DST. A value of zero (0) indicates that this time type is standard time.

(desig)idx: A one-octet unsigned integer specifying a zero-based index into the series of time zone designation octets, thereby selecting a particular designation string. Each index MUST be in the range [0, "charcnt" - 1]; it designates the NUL-terminated string of octets starting at position "idx" in the time zone designations. (This string MAY be empty.) A NUL octet MUST exist in the time zone designations at or after position "idx".

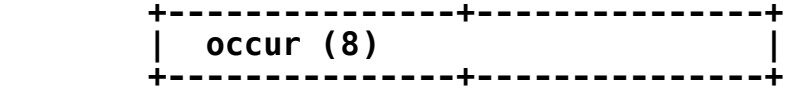
time zone designations: A series of octets constituting an array of NUL-terminated (0x00) time zone designation strings. The total number of octets is specified by the "charcnt" field in the header. Note that two designations MAY overlap if one is a suffix of the other. The character encoding of time zone designation strings is not specified; however, see Section 4 of this document.

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 (the lengths of multi-octet fields are shown in parentheses):

Version 1 Data Block:



version 2+ Data Block:



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. The value 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.

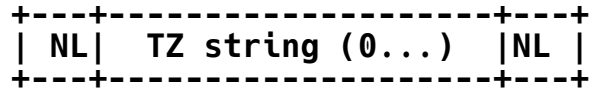
The type corresponding to a transition time specifies local time for timestamps starting at the given transition time and continuing up to, but not including, the next transition time. Local time for timestamps before the first transition is specified by the first time type (time type 0). Local time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

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 lengths of multi-octet fields are shown in parentheses):



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 version 2+ 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 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 version 2+ data, the string MUST be consistent with the last version 2+ transition. In other words, evaluating the TZ string at the time of the last transition should yield the same time type as was specified in the last transition. The string MUST NOT contain NUL octets or be NUL-terminated, and it SHOULD NOT begin with the ':' (colon) character.

The TZif footer is present only in version 2 and 3 files, as the obsolescent version 1 format was designed before the need for a footer was apparent.

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 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

The following practices help ensure the 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 that only understand version 1 MUST ignore any data that extends beyond the calculated end of the version 1 data block.

- 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 version 1 header and data block SHOULD be a contiguous sub-sequence of the time changes defined by the version 2+ 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 supporting obsolescent readers use a "timecnt" of zero in the version 1 data block 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 version 1 header and data block except for the purpose of skipping over them.

- Implementations SHOULD calculate the total lengths of the headers and data blocks and check that they all fit within 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 as necessary (none are necessary if the file is truncated to a range that precedes the first leap second).

  * "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 media type "application/tzif" (Section 8.1) in its "capabilities" response (see Section 5.1 of [RFC7808]). A TZDIST service MAY also include the 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" 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 version 2+ transition time, if present.

When truncating the start of a TZif file, the service MUST supply in the version 2+ 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 version 2+ 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 TZDIST Request for TZif Data

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 it does not define any extensible 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

The Time Zone Information Format contains publicly available data, and it does not define any extensible areas that could be used to store private data.

As discussed in Section 9 of [RFC7808], transmission of time zone data over an insecure communications channel could leak the past, current, or future location of a device or user. As such, TZif data transmitted over a public communications channel MUST be protected with a confidentiality layer such as that provided by Transport Layer Security (TLS) [RFC8446].

8. IANA Considerations

This document defines two media types [RFC6838] 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 RFC 8536.

Interoperability considerations: See Section 4 of RFC 8536.

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) [ZIC] and the GNU C Library [GNU-C]. 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 "Authors' Addresses" section of RFC 8536.

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 RFC 8536.

Interoperability considerations: See Section 4 of RFC 8536.

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) [ZIC] and the GNU C Library [GNU-C]. 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 "Authors' Addresses" section of RFC 8536.

Change controller: IETF

9. References

9.1. Normative References

[GNU-C] "The GNU C Library (glibc)", <https://www.gnu.org/software/libc/>.

[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/>.

[RFC20] 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>.

[ZIC] Kerrisk, M., "ZIC(8)", man-pages release 4.16, February 2010, <http://man7.org/linux/man-pages/man8/zic.8.html>.

9.2. Informative References

[EGGERT-TZ] "History for tz", October 2018, <https://github.com/eggert/tz/commits/master/tzfile.5>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[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>.

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 [EGGERT-TZ]. 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 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 documenting other common bugs in readers.

Interoperability problems with TZif include the following:

- Some readers examine only version 1 data. As a partial workaround, a writer can output as much version 1 data as possible. However, a reader should ignore version 1 data and use version 2+ 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.

- 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.

- 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.
- 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.
- Some readers mishandle time zone abbreviations like "-08" that contain '+', '-', or digits.
- 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.
- 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".
- Some readers mishandle UT offsets that are not a multiple of one hour, 15 minutes, or 1 minute.

Appendix B. Example TZif Files

The following sections contain annotated hexadecimal dumps of example TZif files.

Note that these examples should only be considered informative. Although the example data entries are current as of the publication date of this document, the data will likely change in the future as leap seconds are added and changes are made to civil time.

B.1. Version 1 File Representing UTC (with Leap Seconds)

File Offset	Data Octets (hexadecimal)	Record Name / Field Name	Field Value
000	54 5a 69 66	magic	"TZif"
004	00	version	0 (1)
005	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
020	00 00 00 01	isutccnt	1
024	00 00 00 01	isstdcnt	1
028	00 00 00 1b	isleapcnt	27
032	00 00 00 00	timecnt	0
036	00 00 00 01	typecnt	1
040	00 00 00 04	charcnt	4
		localtimetype[0]	
044	00 00 00 00	utcoff	00:00
048	00	isdst	0 (no)
049	00	desigidx	0
050	55 54 43 00	designations[0]	"UTC"
		leapsecond[0]	
054	04 b2 58 00	occurrence	78796800 (1972-06-30T23:59:60Z)
058	00 00 00 01	correction	1
		leapsecond[1]	
062	05 a4 ec 01	occurrence	94694401 (1972-12-31T23:59:60Z)
066	00 00 00 02	correction	2
		leapsecond[2]	
070	07 86 1f 82	occurrence	126230402 (1973-12-31T23:59:60Z)
074	00 00 00 03	correction	3
		leapsecond[3]	
078	09 67 53 03	occurrence	157766403 (1974-12-31T23:59:60Z)
082	00 00 00 04	correction	4

		leapsecond[4]	
086	0b 48 86 84	occurrence	189302404 (1975-12-31T23:59:60Z)
090	00 00 00 05	correction	5
		leapsecond[5]	
094	0d 2b 0b 85	occurrence	220924805 (1976-12-31T23:59:60Z)
098	00 00 00 06	correction	6
		leapsecond[6]	
102	0f 0c 3f 06	occurrence	252460806 (1977-12-31T23:59:60Z)
106	00 00 00 07	correction	7
		leapsecond[7]	
110	10 ed 72 87	occurrence	283996807 (1978-12-31T23:59:60Z)
114	00 00 00 08	correction	8
		leapsecond[8]	
118	12 ce a6 08	occurrence	315532808 (1979-12-31T23:59:60Z)
122	00 00 00 09	correction	9
		leapsecond[9]	
126	15 9f ca 89	occurrence	362793609 (1981-06-30T23:59:60Z)
130	00 00 00 0a	correction	10
		leapsecond[10]	
134	17 80 fe 0a	occurrence	394329610 (1982-06-30T23:59:60Z)
138	00 00 00 0b	correction	11
		leapsecond[11]	
142	19 62 31 8b	occurrence	425865611 (1983-06-30T23:59:60Z)
146	00 00 00 0c	correction	12
		leapsecond[12]	
150	1d 25 ea 0c	occurrence	489024012 (1985-06-30T23:59:60Z)
154	00 00 00 0d	correction	13
		leapsecond[13]	
158	21 da e5 0d	occurrence	567993613 (1987-12-31T23:59:60Z)

162	00 00 00 0e	correction	14
166	25 9e 9d 8e	leapsecond[14] occurrence	631152014 (1989-12-31T23:59:60Z)
170	00 00 00 0f	correction	15
174	27 7f d1 0f	leapsecond[15] occurrence	662688015 (1990-12-31T23:59:60Z)
178	00 00 00 10	correction	16
182	2a 50 f5 90	leapsecond[16] occurrence	709948816 (1992-06-30T23:59:60Z)
186	00 00 00 11	correction	17
190	2c 32 29 11	leapsecond[17] occurrence	741484817 (1993-06-30T23:59:60Z)
194	00 00 00 12	correction	18
198	2e 13 5c 92	leapsecond[18] occurrence	773020818 (1994-06-30T23:59:60Z)
202	00 00 00 13	correction	19
206	30 e7 24 13	leapsecond[19] occurrence	820454419 (1995-12-31T23:59:60Z)
210	00 00 00 14	correction	20
214	33 b8 48 94	leapsecond[20] occurrence	867715220 (1997-06-30T23:59:60Z)
218	00 00 00 15	correction	21
222	36 8c 10 15	leapsecond[21] occurrence	915148821 (1998-12-31T23:59:60Z)
226	00 00 00 16	correction	22
230	43 b7 1b 96	leapsecond[22] occurrence	1136073622 (2005-12-31T23:59:60Z)
234	00 00 00 17	correction	23

		leapsecond[23]	
238	49 5c 07 97	occurrence	1230768023 (2008-12-31T23:59:60Z)
242	00 00 00 18	correction	24
		leapsecond[24]	
246	4f ef 93 18	occurrence	1341100824 (2012-06-30T23:59:60Z)
250	00 00 00 19	correction	25
		leapsecond[25]	
254	55 93 2d 99	occurrence	1435708825 (2015-06-30T23:59:60Z)
258	00 00 00 1a	correction	26
		leapsecond[26]	
262	58 68 46 9a	occurrence	1483228826 (2016-12-31T23:59:60Z)
266	00 00 00 1b	correction	27
270	00	UT/local[0]	0 (local)
271	00	standard/wall[0]	0 (wall)

To determine TAI corresponding to 2000-01-01T00:00:00Z (UNIX time = 946684800), the following procedure would be followed:

1. Find the latest leap-second occurrence prior to the time of interest (leapsecond[21]) and note the correction value (LEAPCORR = 22).

2. Add LEAPCORR + 10 to the time of interest to yield TAI of 2000-01-01T00:00:32.

B.2. Version 2 File Representing Pacific/Honolulu

File Offset	Hexadecimal Octets	Record Name / Field Name	Field Value
000	54 5a 69 66	magic	"TZif"
004	32	version	'2' (2)
005	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
020	00 00 00 06	isutccnt	6
024	00 00 00 06	isstdcnt	6
028	00 00 00 00	isleapcnt	0
032	00 00 00 07	timecnt	7
036	00 00 00 06	typecnt	6
040	00 00 00 14	charcnt	20
044	80 00 00 00	trans time[0]	-2147483648 (1901-12-13T20:45:52Z)
048	bb 05 43 48	trans time[1]	-1157283000 (1933-04-30T12:30:00Z)
052	bb 21 71 58	trans time[2]	-1155436200 (1933-05-21T21:30:00Z)
056	cb 89 3d c8	trans time[3]	-880198200 (1942-02-09T12:30:00Z)
060	d2 23 f4 70	trans time[4]	-769395600 (1945-08-14T23:00:00Z)
064	d2 61 49 38	trans time[5]	-765376200 (1945-09-30T11:30:00Z)
068	d5 8d 73 48	trans time[6]	-712150200 (1947-06-08T12:30:00Z)
072	01	trans type[0]	1
073	02	trans type[1]	2
074	01	trans type[2]	1
075	03	trans type[3]	3
076	04	trans type[4]	4
077	01	trans type[5]	1
078	05	trans type[6]	5
079	ff ff 6c 02	localtimetype[0] utcoff	-37886 (-10:21:26)
083	00	isdst	0 (no)
084	00	desigidx	0

		localtimetype[1]	
085	ff ff 6c 58	utcoff	-37800 (-10:30)
089	00	isdst	0 (no)
090	04	desigidx	4
		localtimetype[2]	
091	ff ff 7a 68	utcoff	-34200 (-09:30)
095	01	isdst	1 (yes)
096	08	desigidx	8
		localtimetype[3]	
097	ff ff 7a 68	utcoff	-34200 (-09:30)
101	01	isdst	1 (yes)
102	0c	desigidx	12
		localtimetype[4]	
103	ff ff 7a 68	utcoff	-34200 (-09:30)
107	01	isdst	1 (yes)
108	10	desigidx	16
		localtimetype[5]	
109	ff ff 73 60	utcoff	-36000 (-10:00)
113	00	isdst	0 (no)
114	04	desigidx	4
115	4c 4d 54 00	designations[0]	"LMT"
119	48 53 54 00	designations[4]	"HST"
123	48 44 54 00	designations[8]	"HDT"
127	48 57 54 00	designations[12]	"HWT"
131	48 50 54 00	designations[16]	"HPT"
135	00	UT/local[0]	1 (UT)
136	00	UT/local[1]	0 (local)
137	00	UT/local[2]	0 (local)
138	00	UT/local[3]	0 (local)
139	01	UT/local[4]	1 (UT)
140	00	UT/local[5]	0 (local)
141	00	standard/wall[0]	1 (standard)
142	00	standard/wall[1]	0 (wall)
143	00	standard/wall[2]	0 (wall)
144	00	standard/wall[3]	0 (wall)
145	01	standard/wall[4]	1 (standard)
146	00	standard/wall[5]	0 (wall)
147	54 5a 69 66	magic	"TZif"
151	32	version	'2' (2)

152	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
167	00 00 00 06	isutccnt	6
171	00 00 00 06	isstdcnt	6
175	00 00 00 00	isleapcnt	0
179	00 00 00 07	timecnt	7
183	00 00 00 06	typecnt	6
187	00 00 00 14	charcnt	20
191	ff ff ff ff 74 e0 70 be	trans time[0]	-2334101314 (1896-01-13T22:31:26Z)
199	ff ff ff ff bb 05 43 48	trans time[1]	-1157283000 (1933-04-30T12:30:00Z)
207	ff ff ff ff bb 21 71 58	trans time[2]	-1155436200 (1933-05-21T21:30:00Z)
215	ff ff ff ff cb 89 3d c8	trans time[3]	-880198200 (1942-02-09T12:30:00Z)
223	ff ff ff ff d2 23 f4 70	trans time[4]	-769395600 (1945-08-14T23:00:00Z)
231	ff ff ff ff d2 61 49 38	trans time[5]	-765376200 (1945-09-30T11:30:00Z)
239	ff ff ff ff d5 8d 73 48	trans time[6]	-712150200 (1947-06-08T12:30:00Z)
247	01	trans type[0]	1
248	02	trans type[1]	2
249	01	trans type[2]	1
250	03	trans type[3]	3
251	04	trans type[4]	4
252	01	trans type[5]	1
253	05	trans type[6]	5
		localtimetype[0]	
254	ff ff 6c 02	utcoff	-37886 (-10:21:26)
258	00	isdst	0 (no)
259	00	desigidx	0
		localtimetype[1]	
260	ff ff 6c 58	utcoff	-37800 (-10:30)
264	00	isdst	0 (no)
265	04	desigidx	4
		localtimetype[2]	
266	ff ff 7a 68	utcoff	-34200 (-09:30)
270	01	isdst	1 (yes)
271	08	desigidx	8

		localtimetype[3]	
272	ff ff 7a 68	utcoff	-34200 (-09:30)
276	01	isdst	1 (yes)
277	0c	desigidx	12
		localtimetype[4]	
278	ff ff 7a 68	utcoff	-34200 (-09:30)
282	01	isdst	1 (yes)
283	10	desigidx	16
		localtimetype[5]	
284	ff ff 73 60	utcoff	-36000 (-10:00)
288	00	isdst	0 (no)
289	04	desigidx	4
290	4c 4d 54 00	designations[0]	"LMT"
294	48 53 54 00	designations[4]	"HST"
298	48 44 54 00	designations[8]	"HDT"
302	48 57 54 00	designations[12]	"HWT"
306	48 50 54 00	designations[16]	"HPT"
310	00	UT/local[0]	0 (local)
311	00	UT/local[1]	0 (local)
312	00	UT/local[2]	0 (local)
313	00	UT/local[3]	0 (local)
314	01	UT/local[4]	1 (UT)
315	00	UT/local[5]	0 (local)
316	00	standard/wall[0]	0 (wall)
317	00	standard/wall[1]	0 (wall)
318	00	standard/wall[2]	0 (wall)
319	00	standard/wall[3]	0 (wall)
320	01	standard/wall[4]	1 (standard)
321	00	standard/wall[5]	0 (wall)
322	0a	NL	'\n'
323	48 53 54 31 30	TZ string	"HST10"
328	0a	NL	'\n'

To determine the local time in this time zone corresponding to 1933-05-04T12:00:00Z (UNIX time = -1156939200), the following procedure would be followed:

1. Find the latest time transition prior to the time of interest (trans time[1]).

2. Reference the corresponding transition type (trans type[1]) to determine the local time type index (2).

3. Reference the corresponding local time type (localtimetype[2]) to determine the offset from UTC (-09:30), the daylight saving indicator (1 = yes), and the index into the time zone designation strings (8).

4. Look up the corresponding time zone designation string (designations[8] = "HDT").

5. Add the UTC offset to the time of interest to yield a local daylight saving time of 1933-05-04T02:30:00-09:30 (HDT).

To determine the local time in this time zone corresponding to 2019-01-01T00:00:00Z (UNIX time = 1546300800), the following procedure would be followed:

1. Find the latest time transition prior to the time of interest (there is no such transition).

2. Look up the TZ string in the footer ("HST10"), which indicates that the time zone designation is "HST" year-round, and the offset to UTC is 10:00.

3. Subtract the UTC offset from the time of interest to yield a standard local time of 2018-12-31T14:00:00-10:00 (HST).

B.3. Truncated Version 3 File Representing Asia/Jerusalem

The following TZif file has been truncated to start on 2038-01-01T00:00:00Z.

File Offset	Hexadecimal Octets	Record Name / Field Name	Field Value
000	54 5a 69 66	magic	"TZif"
004	33	version	'3' (3)
005	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
020	00 00 00 00	isutccnt	0
024	00 00 00 00	isstdcnt	0
028	00 00 00 00	isleapcnt	0
032	00 00 00 00	timecnt	0
036	00 00 00 00	typecnt	0
040	00 00 00 00	charcnt	0
044	54 5a 69 66	magic	"TZif"
048	33	version	'3' (3)
049	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
064	00 00 00 03	isutccnt	1
068	00 00 00 03	isstdcnt	1
072	00 00 00 00	isleapcnt	0
076	00 00 00 03	timecnt	1
080	00 00 00 03	typecnt	1
084	00 00 00 08	charcnt	4
088	00 00 00 00 7f e8 17 80	trans time[0]	2145916800 (2038-01-01T00:00:00Z)
096	00	trans type[0]	0
		localtimetype[0]	
097	00 00 1c 20	utcoff	7200 (+02:00)
101	00	isdst	0 (no)
102	00	desigidx	0
103	49 53 54 00	designations[0]	"IST"
107	01	UT/local[0]	1 (UT)

108	01	standard/wall[0]	1 (standard)
109	0a	NL	'\n'
110	49 53 54 2d 32 49 44 54 2c 4d 33 2e 34 2e 34 2f 32 36 2c 4d 31 30 2e 35 2e 30	TZ string	"IST-2IDT, M3.4.4/26,M10.5.0"
136	0a	NL	'\n'

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.

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