

Internet Engineering Task Force (IETF)
Request for Comments: 7808
Category: Standards Track
ISSN: 2070-1721

M. Douglass
Spherical Cow Group
C. Daboo
Apple
March 2016

Time Zone Data Distribution Service

Abstract

This document defines a time zone data distribution 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.

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 5741](#).

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc7808>.

Copyright Notice

Copyright (c) 2016 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 (<http://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	4
1.1.	Conventions	4
2.	Architectural Overview	5
3.	General Considerations	7
3.1.	Time Zone	7
3.2.	Time Zone Data	7
3.3.	Time Zone Metadata	7
3.4.	Time Zone Data Server	7
3.5.	Observance	7
3.6.	Time Zone Identifiers	7
3.7.	Time Zone Aliases	8
3.8.	Time Zone Localized Names	8
3.9.	Truncating Time Zones	9
3.10.	Time Zone Versions	10
4.	Time Zone Data Distribution Service Protocol	10
4.1.	Server Protocol	10
4.1.1.	Time Zone Queries	11
4.1.2.	Time Zone Formats	11
4.1.3.	Time Zone Localization	12
4.1.4.	Conditional Time Zone Requests	12
4.1.5.	Expanded Time Zone Data	14
4.1.6.	Server Requirements	14
4.1.7.	Error Responses	14
4.1.8.	Extensions	14
4.2.	Client Guidelines	14
4.2.1.	Discovery	14
4.2.1.1.	SRV Service Labels for the Time Zone Data Distribution Service	15
4.2.1.2.	TXT Records for a Time Zone Data Distribution Service	15
4.2.1.3.	Well-Known URI for a Time Zone Data Distribution Service	16
4.2.1.3.1.	Example: Well-Known URI Redirects to Actual Context Path	17
4.2.2.	Synchronization of Time Zones	17
4.2.2.1.	Initial Synchronization of All Time Zones	17
4.2.2.2.	Subsequent Synchronization of All Time Zones	17
4.2.2.3.	Synchronization with Preexisting Time Zone Data	18
5.	Actions	18
5.1.	"capabilities" Action	18
5.1.1.	Example: get capabilities	19

5.2.	"list" Action	21
5.2.1.	Example: List Time Zone Identifiers	22
5.3.	"get" Action	23
5.3.1.	Example: Get Time Zone Data	24
5.3.2.	Example: Conditional Get Time Zone Data	25

5.3.3.	Example: Get Time Zone Data Using a Time Zone Alias	25
5.3.4.	Example: Get Truncated Time Zone Data	26
5.3.5.	Example: Request for a Nonexistent Time Zone	27
5.4.	"expand" Action	27
5.4.1.	Example: Expanded JSON Data Format	29
5.5.	"find" Action	30
5.5.1.	Example: find action	31
5.6.	"leapseconds" Action	32
5.6.1.	Example: Get Leap-Second Information	33
6.	JSON Definitions	34
6.1.	capabilities Action Response	34
6.2.	list/find Action Response	37
6.3.	expand Action Response	38
6.4.	leapseconds Action Response	39
7.	New iCalendar Properties	40
7.1.	Time Zone Upper Bound	40
7.2.	Time Zone Identifier Alias Property	41
8.	Security Considerations	42
9.	Privacy Considerations	43
10.	IANA Considerations	44
10.1.	Service Actions Registration	45
10.1.1.	Service Actions Registration Procedure	45
10.1.2.	Registration Template for Actions	46
10.1.3.	Actions Registry	47
10.2.	timezone Well-Known URI Registration	47
10.3.	Service Name Registrations	47
10.3.1.	timezone Service Name Registration	47
10.3.2.	timezones Service Name Registration	48
10.4.	TZDIST Identifiers Registry	48
10.4.1.	Registration of invalid-action Error URN	49
10.4.2.	Registration of invalid-changedsince Error URN	49
10.4.3.	Registration of tzid-not-found Error URN	50
10.4.4.	Registration of invalid-format Error URN	50
10.4.5.	Registration of invalid-start Error URN	50
10.4.6.	Registration of invalid-end Error URN	51
10.4.7.	Registration of invalid-pattern Error URN	51

10.5.	iCalendar Property Registrations	52
11.	References	52
11.1.	Normative References	52
11.2.	Informative References	55
	Acknowledgements	55
	Authors' Addresses	56

[1.](#) Introduction

Time zone data typically combines a coordinated universal time (UTC) offset with daylight saving time (DST) rules. Time zones are typically tied to specific geographic and geopolitical regions. Whilst the UTC offset for particular regions changes infrequently, DST rules can change frequently and sometimes with very little notice (maybe hours before a change comes into effect).

Calendaring and scheduling systems, such as those that use iCalendar [[RFC5545](#)], as well as operating systems, critically rely on time zone data to determine the correct local time. As such, they need to be kept up to date with changes to time zone data. To date, there has been no fast and easy way to do that. Time zone data is often supplied in the form of a set of data files that have to be "compiled" into a suitable database format for use by the client application or operating system. In the case of operating systems, often those changes only get propagated to client machines when there is an operating system update, which can be infrequent, resulting in inaccurate time zone data being present for significant amounts of time. In some cases, old versions of operating systems stop being supported, but are still in use and thus require users to manually "patch" their system to keep up to date with time zone changes.

Along with time zone data, it is also important to track the use of leap seconds to allow a mapping between International Atomic Time (TAI) and UTC. Leap seconds can be added (or possibly removed) at various times of year in an irregular pattern typically determined by precise astronomical observations. The insertion of leap seconds

into UTC is currently the responsibility of the International Earth Rotation Service.

This specification defines a time zone data distribution service protocol that allows for fast, reliable, and accurate delivery of time zone data and leap-second information to client systems. This protocol is based on HTTP [[RFC7230](#)] using a simple JSON-based API [[RFC7159](#)].

This specification does not define the source of the time zone data or leap-second information. It is assumed that a reliable and accurate source is available. One such source is the IANA-hosted time zone database [[RFC6557](#)].

[1.1](#). Conventions

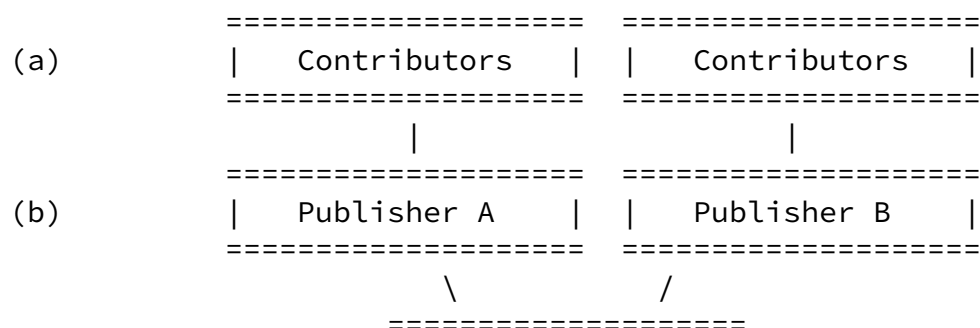
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

Unless otherwise indicated, UTC date-time values as specified in [[RFC3339](#)] use a "Z" suffix, and not fixed numeric offsets.

This specification contains examples of HTTP requests and responses. In some cases, additional line breaks have been introduced into the request or response data to match maximum line-length limits of this document.

[2](#). Architectural Overview

The overall process for the delivery of time zone data can be visualized via the diagram below.



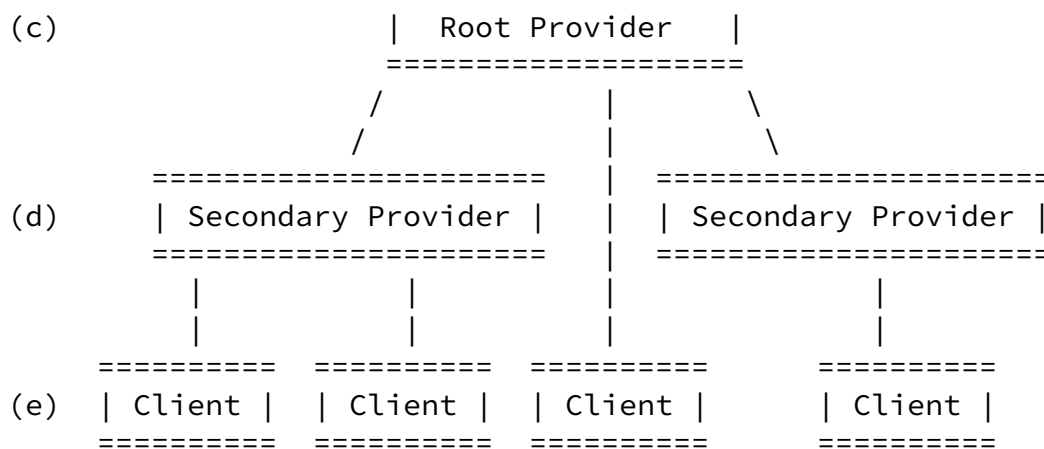


Figure 1: Time Zone Data Distribution Service Architecture

The overall service is made up of several layers:

- (a) Contributors: Individuals, governments, or organizations that provide information about time zones to the publishing process. There can be many contributors. Note this specification does not address how contributions are made.

- (b) Publishers: Publishers aggregate information from contributors, determine the reliability of the information and, based on that, generate time zone data. There can be many publishers, each getting information from many different contributors. In some cases, a publisher may choose to "republish" data from another publisher.
- (c) Root Providers: Servers that obtain and then provide the time zone data from publishers and make that available to other servers or clients. There can be many root providers. Root providers can choose to supply time zone data from one or more publishers.
- (d) Secondary Providers: Servers that handle the bulk of the requests and reduce the load on root servers. These will

typically be simple, caches of the root server, located closer to clients. For example a large Internet Service Provider (ISP) may choose to set up their own secondary provider to allow clients within their network to make requests of that server rather than make requests of servers outside their network. Secondary servers will cache and periodically refresh data from the root servers.

- (e) Clients: Applications, operating systems, etc., that make use of time zone data and retrieve that from either root or secondary providers.

Some of those layers may be coalesced by implementors. For example, a vendor may choose to implement the entire service as a single monolithic virtual server with the address embedded in distributed systems. Others may choose to provide a service consisting of multiple layers of providers, many secondary servers, and a small number of root servers.

This specification is concerned only with the protocol used to exchange data between providers and from provider to client. This specification does not define how contributors pass their information to publishers, nor how those publishers vet that information to obtain trustworthy data, nor the format of the data produced by the publishers.

[3.](#) General Considerations

This section defines several terms and explains some key concepts used in this specification.

[3.1.](#) Time Zone

A time zone is a description of the past and predicted future

timekeeping practices of a collection of clocks that are intended to agree.

Note that the term "time zone" does not have the common meaning of a region of the world at a specific UTC offset, possibly modified by daylight saving time. For example, the "Central European Time" zone can correspond to several time zones "Europe/Berlin", "Europe/Paris", etc., because subregions have kept time differently in the past.

[3.2.](#) Time Zone Data

Time zone data is data that defines a single time zone, including an identifier, UTC offset values, DST rules, and other information such as time zone abbreviations.

[3.3.](#) Time Zone Metadata

Time zone metadata is data that describes additional properties of a time zone that is not itself included in the time zone data. This can include such things as the publisher name, version identifier, aliases, and localized names (see below).

[3.4.](#) Time Zone Data Server

A time zone data server is a server implementing the Time Zone Data Distribution Service Protocol defined by this specification.

[3.5.](#) Observance

A time zone with varying rules for the UTC offset will have adjacent periods of time that use different UTC offsets. Each period of time with a constant UTC offset is called an observance.

[3.6.](#) Time Zone Identifiers

Time zone identifiers are unique names associated with each time zone, as defined by publishers. The iCalendar [[RFC5545](#)] specification has a "TZID" property and parameter whose value is set to the corresponding time zone identifier and used to identify time zone data and relate time zones to start and end dates in events,

etc. This specification does not define what format of time zone

identifiers should be used. It is possible that time zone identifiers from different publishers overlap, and there might be a need for a provider to distinguish those with some form of "namespace" prefix identifying the publisher. However, development of a standard (global) naming scheme for time zone identifiers is out of scope for this specification.

[3.7.](#) Time Zone Aliases

Time zone aliases map a name onto a time zone identifier. For example, "US/Eastern" is usually mapped on to "America/New_York". Time zone aliases are typically used interchangeably with time zone identifiers when presenting information to users.

A time zone data distribution service needs to maintain time zone alias mapping information and expose that data to clients as well as allow clients to query for time zone data using aliases. When returning time zone data to a client, the server returns the data with an identifier matching the query, but it can include one or more additional identifiers in the data to provide a hint to the client that alternative identifiers are available. For example, a query for "US/Eastern" could include additional identifiers for "America/New_York" or "America/Montreal".

The set of aliases may vary depending on whether time zone data is truncated (see [Section 3.9](#)). For example, a client located in the US state of Michigan may see "US/Eastern" as an alias for "America/Detroit", whereas a client in the US state of New Jersey may see it as an alias for "America/New_York", and all three names may be aliases if time zones are truncated to post-2013 data.

[3.8.](#) Time Zone Localized Names

Localized names are names for time zones that can be presented to a user in their own language. Each time zone may have one or more localized names associated with it. Names would typically be unique in their own locale as they might be presented to the user in a list. Localized names are distinct from abbreviations commonly used for UTC offsets within a time zone. For example, the time zone "America/New_York" may have the localized name "Nueva York" in a Spanish locale, as distinct from the abbreviations "EST" and "EDT", which may or may not have their own localizations.

A time zone data distribution service might need to maintain localized name information, for one or more chosen languages, as well as allow clients to query for time zone data using localized names.

3.9. Truncating Time Zones

Time zone data can contain information about past and future UTC offsets that may not be relevant for a particular server's intended clients. For example, calendaring and scheduling clients are likely most concerned with time zone data that covers a period for one or two years in the past on into the future, as users typically create new events only for the present and future. Similarly, time zone data might contain a large amount of "future" information about transitions occurring many decades into the future. Again, clients might be concerned only with a smaller range into the future, and data past that point might be unnecessary.

To avoid having to send unnecessary data, servers can choose to truncate time zone data to a range determined by start- and end-point date-time values, and to provide only offsets and rules between those points. If such truncation is done, the server **MUST** include the ranges it is using in the "capabilities" action response (see [Section 6.1](#)), so that clients can take appropriate action if they need time zone data for times outside of those ranges.

The truncation points at the start and end of a range are always a UTC date-time value, with the start point being "inclusive" to the overall range, and the end point being "exclusive" to the overall range (i.e., the end value is just past the end of the last valid value in the range). A server will advertise a truncation range for the truncated data it can supply or will provide an indicator that it can truncate at any start or end point to produce arbitrary ranges. In addition, the server can advertise that it supplies untruncated data -- that is, data that covers the full range of times available from the source publisher. In the absence of any indication of truncated data available on the server, the server will supply only untruncated data.

When truncating the start of a "VTIMEZONE" component, the server **MUST** include exactly one "STANDARD" or "DAYLIGHT" subcomponent with a "DTSTART" property value that matches the start point of the truncation range, and appropriate "TZOFFSETFROM" and "TZOFFSETTO" properties to indicate the correct offset in effect right before and after the start point of the truncation range. This subcomponent, which is the first observance defined by the time zone data, represents the earliest valid date-time covered by the time zone data in the truncated "VTIMEZONE" component.

When truncating the end of a "VTIMEZONE" component, the server **MUST** include a "TZUNTIL" iCalendar property ([Section 7.1](#)) in the

"VTIMEZONE" component to indicate the end point of the truncation range.

[3.10.](#) Time Zone Versions

Time zone data changes over time, and it is important for consumers of that data to stay up to date with the latest versions. As a result, it is useful to identify individual time zones with a specific version number or version identifier as supplied by the time zone data publisher. There are two common models that time zone data publishers might use to publish updates to time zone data:

- a. with the "monolithic" model, the data for all time zones is published in one go, with a single version number or identifier applied to the entire data set. For example, a publisher producing data several times a year might use version identifiers "2015a", "2015b", etc.
- b. with the "incremental" model, each time zone has its own version identifier, so that each time zone can be independently updated without impacting any others. For example, if the initial data has version "A.1" for time zone "A", and "B.1" for time zone "B", and then time zone "B" changes; when the data is next published, time zone "A" will still have version "A.1", but time zone "B" will now have "B.2".

A time zone data distribution service needs to ensure that the version identifiers used by the time zone data publisher are available to any client, along with the actual publisher name on a per-time-zone basis. This allows clients to compare publisher/version details on any server, with existing locally cached client data, and only fetch those time zones that have actually changed (see [Section 4.2.2](#) for more details on how clients synchronize data from the server).

[4.](#) Time Zone Data Distribution Service Protocol

[4.1.](#) Server Protocol

The time zone data distribution service protocol uses HTTP [[RFC7230](#)] for query and delivery of time zone data, metadata, and leap-second information. The interactions with the HTTP server can be broken

down into a set of "actions" that define the overall function being requested (see [Section 5](#)). Each action targets a specific HTTP resource using the GET method, with various request-URI parameters altering the behavior as needed.

The HTTP resources used for requests will be identified via URI templates [[RFC6570](#)]. The overall time zone data distribution service has a "context path" request-URI template defined as "{/service-prefix}". This "root" prefix is discovered by the client as per

[Section 4.2.1](#). Request-URIs that target time zone data directly use the prefix template "{/service-prefix,data-prefix}". The second component of the prefix template can be used to introduce additional path segments in the request-URI to allow for alternative ways to "partition" the time zone data. For example, time zone data might be partitioned by publisher release dates or version identifiers. This specification does not define any partitions; that is left for future extensions. When the "data-prefix" variable is empty, the server is expected to return the current version of time zone data it has for all publishers it supports.

All URI template variable values, and URI request parameters that contain text values, MUST be encoded using the UTF-8 [[RFC3629](#)] character set. All responses MUST return data using the UTF-8 [[RFC3629](#)] character set. It is important to note that any "/" characters, which are frequently found in time zone identifiers, are percent-encoded when used in the value of a path segment expansion variable in a URI template (as per [Section 3.2.6 of \[RFC6570\]](#)). Thus, the time zone identifier "America/New_York" would appear as "America%2FNew_York" when used as the value for the "{/tzid}" URI template variable defined later in this specification.

The server provides time zone metadata in the form of a JSON [[RFC7159](#)] object. Clients can directly request the time zone metadata or issue queries for subsets of metadata that match specific criteria.

Security and privacy considerations for this protocol are discussed in detail in [Sections 8](#) and [9](#), respectively.

[4.1.1](#). Time Zone Queries

Time zone identifiers, aliases, or localized names can be used to query for time zone data or metadata. This will be more explicitly defined below for each action. In general, however, if a "tzid" URI template variable is used, then the value may be an identifier or an alias. When the "pattern" URI query parameter is used, it may be an identifier, an alias, or a localized name.

[4.1.2.](#) Time Zone Formats

The default media type [[RFC2046](#)] format for returning time zone data is the iCalendar [[RFC5545](#)] data format. In addition, the iCalendar-in-XML [[RFC6321](#)] and iCalendar-in-JSON [[RFC7265](#)] representations are available. Clients use the HTTP Accept header field (see [Section 5.3.2 of \[RFC7231\]](#)) to indicate their preference for the returned data format. Servers indicate the available formats that they support via the "capabilities" action response ([Section 5.1](#)).

[4.1.3.](#) Time Zone Localization

As per [Section 3.8](#), time zone data can support localized names. Clients use the HTTP Accept-Language header field (see [Section 5.3.5 of \[RFC7231\]](#)) to indicate their preference for the language used for localized names in the response data.

[4.1.4.](#) Conditional Time Zone Requests

When time zone data or metadata changes, it needs to be distributed in a timely manner because changes to local time offsets might occur within a few days of the publication of the time zone data changes. Typically, the number of time zones that change is small, whilst the overall number of time zones can be large. Thus, when a client is using more than a few time zones, it is more efficient for the client to be able to download only those time zones that have changed (an incremental update).

Clients initially request a full list of time zones from the server using a "list" action request (see [Section 5.2](#)). The response to that request includes two items the client caches for use with subsequent "conditional" (incremental update) requests:

1. An opaque synchronization token in the "synctoken" JSON member. This token changes whenever there is a change to any metadata

associated with one or more time zones (where the metadata is the information reported in the "list" action response for each time zone).

2. The HTTP ETag header field value for each time zone returned in the response. The ETag header field value is returned in the "etag" JSON member, and it corresponds to the ETag header field value that would be returned when executing a "get" action request (see [Section 5.3](#)) against the corresponding time zone data resource.

For subsequent updates to cached data, clients can use the following procedure:

- a. Send a "list" action request with a "changedsince" URI query parameter with its value set to the last opaque synchronization token returned by the server. The server will return time zone metadata for only those time zones that have changed since the last request.
- b. The client will cache the new opaque synchronization token returned in the response for the next incremental update, along with the returned time zone metadata information.

- c. The client will check each time zone metadata to see if the "etag" value is different from that of any cached time zone data it has.
- d. The client will use a "get" action request to update any cached time zone data for those time zones whose ETag header field value has changed.

Note that time zone metadata will always change when the corresponding time zone data changes. However, the converse is not true: it is possible for some piece of the time zone metadata to change without the corresponding time zone data changing. e.g., for the case of a "monolithic" publisher (see [Section 3.10](#)), the version identifier in every time zone metadata element will change with each new published revision; however, only a small subset of time zone data will actually change.

If a client needs data for only one or a small set of time zones

(e.g., a clock in a fixed location), then it can use a conditional HTTP request to determine if the time zone data has changed and retrieve the new data. The full details of HTTP conditional requests are described in [[RFC7232](#)]; what follows is a brief summary of what a client typically does.

- a. When the client retrieves the time zone data from the server using a "get" action (see [Section 5.3](#)), the server will include an HTTP ETag header field in the response.
- b. The client will store the value of that header field along with the request-URI used for the request.
- c. When the client wants to check for an update, it issues another "get" action HTTP request on the original request-URI, but this time it includes an If-None-Match HTTP request header field, with a value set to the ETag header field value from the previous response. If the data for the time zone has not changed, the server will return a 304 (Not Modified) HTTP response. If the data has changed, the server will return a normal HTTP success response that will include the changed data, as well as a new value for the ETag header field.

Clients SHOULD poll for changes, using an appropriate conditional request, at least once a day. A server acting as a secondary provider, caching time zone data from another server, SHOULD poll for changes once per hour. See [Section 8](#) on expected client and server behavior regarding high request rates.

[4.1.5](#). Expanded Time Zone Data

Determining time zone offsets at a particular point in time is often a complicated process, as the rules for daylight saving time can be complex. To help with this, the time zone data distribution service provides an action that allows clients to request the server to expand a time zone into a set of "observances" over a fixed period of time (see [Section 5.4](#)). Each of these observances describes a UTC onset time and UTC offsets for the prior time and the observance time. Together, these provide a quick way for "thin" clients to determine an appropriate UTC offset for an arbitrary date without

having to do full time zone expansion themselves.

[4.1.6.](#) Server Requirements

To enable a simple client implementation, servers SHOULD ensure that they provide or cache data for all commonly used time zones, from various publishers. That allows client implementations to configure a single server to get all time zone data. In turn, any server can refresh any of the data from any other server -- though the root servers may provide the most up-to-date copy of the data.

[4.1.7.](#) Error Responses

When an HTTP error response is returned to the client, the server SHOULD return a JSON "problem details" object in the response body, as per [[RFC7807](#)]. Every JSON "problem details" object MUST include a "type" member with a URI value matching the applicable error code (defined for each action in [Section 5](#)).

[4.1.8.](#) Extensions

This protocol is designed to be extensible through a standards-based registration mechanism (see [Section 10](#)). It is anticipated that other useful time zone actions will be added in the future (e.g., mapping a geographical location to time zone identifiers, getting change history for time zones), and so, servers MUST return a description of their capabilities. This will allow clients to determine if new features have been installed and, if not, fall back on earlier features or disable some client capabilities.

[4.2.](#) Client Guidelines

[4.2.1.](#) Discovery

Client implementations need to either know where the time zone data distribution service is located or discover it through some mechanism. To use a time zone data distribution service, a client

needs a Fully Qualified Domain Name (FQDN), port, and HTTP request-URI path. The request-URI path found via discovery is the "context path" for the service itself. The "context path" is used as the value of the "service-prefix" URI template variable when executing

actions (see [Section 5](#)).

The following subsections describe two methods of service discovery using DNS SRV records [[RFC2782](#)] and an HTTP "well-known" [[RFC5785](#)] resource. However, alternative mechanisms could also be used (e.g., a DHCP server option [[RFC2131](#)]).

[4.2.1.1](#). SRV Service Labels for the Time Zone Data Distribution Service

[[RFC2782](#)] defines a DNS-based service discovery protocol that has been widely adopted as a means of locating particular services within a local area network and beyond, using SRV RR records. This can be used to discover a service's FQDN and port.

This specification adds two service types for use with SRV records:

timezone: Identifies a time zone data distribution server that uses HTTP without Transport Layer Security ([[RFC2818](#)]).

timezones: Identifies a time zone data distribution server that uses HTTP with Transport Layer Security ([[RFC2818](#)]).

Clients MUST honor "TTL", "Priority", and "Weight" values in the SRV records, as described by [[RFC2782](#)].

Example: service record for server without Transport Layer Security.

```
_timezone._tcp SRV 0 1 80 tz.example.com.
```

Example: service record for server with transport layer security.

```
_timezones._tcp SRV 0 1 443 tz.example.com.
```

[4.2.1.2](#). TXT Records for a Time Zone Data Distribution Service

When SRV RRs are used to advertise a time zone data distribution service, it is also convenient to be able to specify a "context path" in the DNS to be retrieved at the same time. To enable that, this specification uses a TXT RR that follows the syntax defined in [Section 6 of \[RFC6763\]](#) and defines a "path" key for use in that record. The value of the key MUST be the actual "context path" to the corresponding service on the server.

A site might provide TXT records in addition to SRV records for each service. When present, clients MUST use the "path" value as the "context path" for the service in HTTP requests. When not present, clients use the ".well-known" URI approach described in [Section 4.2.1.3](#).

As per [Section 8](#), the server MAY require authentication when a client tries to access the path URI specified by the TXT RR (i.e., the server would return a 401 status response to the unauthenticated request from the client, then return a redirect response after a successful authentication by the client).

Example: text record for service with Transport Layer Security.

_timezones._tcp TXT path=/timezones

[4.2.1.3](#). Well-Known URI for a Time Zone Data Distribution Service

A "well-known" URI [[RFC5785](#)] is registered by this specification for the Time Zone Data Distribution service, "timezone" (see [Section 10](#)). This URI points to a resource that the client can use as the initial "context path" for the service they are trying to connect to. The server MUST redirect HTTP requests for that resource to the actual "context path" using one of the available mechanisms provided by HTTP (e.g., using an appropriate 3xx status response). Clients MUST handle HTTP redirects on the ".well-known" URI, taking into account security restrictions on redirects described in [Section 8](#). Servers MUST NOT locate the actual time zone data distribution service endpoint at the ".well-known" URI as per [Section 1.1 of \[RFC5785\]](#). The "well-known" URI MUST be present on the server, even when a TXT RR ([Section 4.2.1.2](#)) is used in the DNS to specify a "context path".

Servers SHOULD set an appropriate Cache-Control header field value (as per [Section 5.2 of \[RFC7234\]](#)) in the redirect response to ensure caching occurs as needed, or as required by the type of response generated. For example, if it is anticipated that the location of the redirect might change over time, then an appropriate "max-age" value would be used.

As per [Section 8](#), the server MAY require authentication when a client tries to access the ".well-known" URI (i.e., the server would return a 401 status response to the unauthenticated request from the client, then return the redirect response after a successful authentication by the client).

[4.2.1.3.1.](#) Example: Well-Known URI Redirects to Actual Context Path

A time zone data distribution server has a "context path" that is `/servlet/timezone`. The client will use `/.well-known/timezone` as the path for the service after it has first found the FQDN and port number via an SRV lookup or via manual entry of information by the user. When the client makes its initial HTTP request against `/.well-known/timezone`, the server would issue an HTTP 301 redirect response with a Location response header field using the path `/servlet/timezone`. The client would then "follow" this redirect to the new resource and continue making HTTP requests there. The client would also cache the redirect information, subject to any Cache-Control directive, for use in subsequent requests.

[4.2.2.](#) Synchronization of Time Zones

This section discusses possible client synchronization strategies using the various protocol elements provided by the server for that purpose.

[4.2.2.1.](#) Initial Synchronization of All Time Zones

When a secondary service or a client wishing to cache all time zone data first starts, or wishes to do a full refresh, it synchronizes with another server by issuing a "list" action to retrieve all the time zone metadata. The client preserves the returned opaque token for subsequent use (see "synctoken" in [Section 5.2.1](#)). The client stores the metadata for each time zone returned in the response. Time zone data for each corresponding time zone can then be fetched and stored locally. In addition, a mapping of aliases to time zones can be built from the metadata. A typical "list" action response size is about 50-100 KB of "pretty printed" JSON data, for a service using the IANA time zone database [[RFC6557](#)], as of the time of publication of this specification.

[4.2.2.2.](#) Subsequent Synchronization of All Time Zones

A secondary service or a client caching all time zones needs to periodically synchronize with a server. To do so, it issues a "list" action with the "changesince" URI query parameter set to the value

of the opaque token returned by the last synchronization. The client again preserves the returned opaque token for subsequent use. The client updates its stored time zone metadata using the new values returned in the response, which contains just the time zone metadata for those time zones changed since the last synchronization. In addition, it compares the "etag" value in each time zone metadata to the ETag header field value for the corresponding time zone data resource it has previously cached; if they are different, it fetches

the new time zone data. Note that if the client presents the server with a "changesince" value that the server does not support, all time zone data is returned, as it would for the case where the request did not include a "changesince" value.

Publishers should take into account the fact that the "outright" deletion of time zone names will cause problems to simple clients, and so aliasing a deleted time zone identifier to a suitable alternate one is preferable.

[4.2.2.3](#). Synchronization with Preexisting Time Zone Data

A client might be pre-provisioned with time zone data from a source other than the time zone data distribution service it is configured to use. In such cases, the client might want to minimize the amount of time zone data it synchronizes by doing an initial "list" action to retrieve all the time zone metadata, but then only fetch time zone data for those time zones that do not match the publisher and version details for the pre-provisioned data.

[5](#). Actions

Servers MUST support the following actions. The information below shows details about each action: the request-URI the client targets (in the form of a URI template [[RFC6570](#)]), a description, the set of allowed query parameters, the nature of the response, and a set of possible error codes for the response (see [Section 4.1.7](#)).

For any error not covered by the specific error codes defined below, the "urn:ietf:params:tzdist:error:invalid-action" error code is returned to the client in the JSON "problem details" object.

The examples in the following subsections presume that the timezone

context path has been discovered to be `"/servlet/timezone"` (as in the example in [Section 4.2.1.3.1](#)).

[5.1](#). "capabilities" Action

Name: capabilities

Request-URI Template:
`{/service-prefix}/capabilities`

Description: This action returns the capabilities of the server, allowing clients to determine if a specific feature has been deployed and/or enabled.

Parameters: None

Response: A JSON object containing a "version" member, an "info" member, and an "actions" member; see [Section 6.1](#).

Possible Error Codes: No specific code.

[5.1.1](#). Example: get capabilities

>> Request <<

GET /servlet/timezone/capabilities HTTP/1.1
Host: tz.example.com

>> Response <<

HTTP/1.1 200 OK
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: application/json; charset="utf-8"
Content-Length: xxxx

```
{
  "version": 1,

  "info": {
    "primary-source": "Olson:2011m",
    "formats": [
      "text/calendar",
```

```

    "application/calendar+xml",
    "application/calendar+json"
  ],
  "truncated" : {
    "any": false,
    "ranges": [
      {
        "start": "1970-01-01T00:00:00Z",
        "end": "*"
      },
      {
        "start": "2010-01-01T00:00:00Z",
        "end": "2020-01-01T00:00:00Z"
      }
    ]
  },
  "untruncated": true
},
"provider-details": "http://tz.example.com/about.html",
"contacts": ["mailto:tzs@example.org"]
},

```

```

"actions": [
  {
    "name": "capabilities",
    "uri-template": "/servlet/timezone/capabilities",
    "parameters": []
  },
  {
    "name": "list",
    "uri-template": "/servlet/timezone/zones{?changedsince}",
    "parameters": [
      {
        "name": "changedsince",
        "required": false,
        "multi": false
      }
    ]
  }
],

```

```

{
  "name": "get",
  "uri-template": "/servlet/timezone/zones{/tzid}{?start,end}",
  "parameters": [
    {
      "name": "start",
      "required": false,
      "multi": false
    },
    {
      "name": "end",
      "required": false,
      "multi": false
    }
  ]
},

{
  "name": "expand",
  "uri-template":
    "/servlet/timezone/zones{/tzid}/observances{?start,end}",
  "parameters": [
    {
      "name": "start",
      "required": true,
      "multi": false
    },
    {
      "name": "end",

```

```

      "required": true,
      "multi": false
    }
  ]
},

{
  "name": "find",
  "uri-template": "/servlet/timezone/zones{?pattern}",
  "parameters": [
    {
      "name": "pattern",

```

```

        "required": true,
        "multi": false
    }
]
},

{
    "name": "leapseconds",
    "uri-template": "/servlet/timezone/leapseconds",
    "parameters": []
}
]
}

```

5.2. "list" Action

Name: list

Request-URI Template:

```
{/service-prefix,data-prefix}/zones{?changedsince}
```

Description: This action lists all time zone identifiers in summary format, with publisher, version, aliases, and optional localized data. In addition, it returns an opaque synchronization token for the entire response. If the "changedsince" URI query parameter is present, its value MUST correspond to a previously returned synchronization token value. When "changedsince" is used, the server MUST return only those time zones that have changed since the specified synchronization token. If the "changedsince" value is not supported by the server, the server MUST return all time zones, treating the request as if it had no "changedsince".

Parameters:

changedsince

OPTIONAL, and MUST NOT occur more than once.

Response: A JSON object containing a "synctoken" member and a "timezones" member; see [Section 6.2](#).

Possible Error Codes:

urn:ietf:params:tzdist:error:invalid-changedsince

The "changedsince" URI query parameter appears more than once.

[5.2.1.](#) Example: List Time Zone Identifiers

In this example the client requests the full set of time zone identifiers.

>> Request <<

```
GET /servlet/timezone/zones HTTP/1.1
Host: tz.example.com
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: application/json; charset="utf-8"
Content-Length: xxxx
```

```
{
  "synctoken": "2009-10-11T09:32:11Z",
  "timezones": [
    {
      "tzid": "America/New_York",
      "etag": "123456789-000-111",
      "last-modified": "2009-09-17T01:39:34Z",
      "publisher": "Example.com",
      "version": "2015a",
      "aliases": ["US/Eastern"],
      "local-names": [
        {
          "name": "America/New_York",
          "lang": "en_US"
        }
      ]
    },
    ...other time zones...
  ]
}
```

[5.3.](#) "get" Action

Name: get

Request-URI Template:

`{/service-prefix,data-prefix}/zones{/tzid}{?start,end}`

The "tzid" variable value is REQUIRED in order to distinguish this action from the "list" action.

Description: This action returns a time zone. The response MUST contain an ETag response header field indicating the current value of the strong entity tag of the time zone resource.

In the absence of any Accept HTTP request header field, the server MUST return time zone data with the "text/calendar" media type.

If the "tzid" variable value is actually a time zone alias, the server will return the matching time zone data with the alias as the identifier in the time zone data. The server MAY include one or more "TZID-ALIAS-OF" properties (see [Section 7.2](#)) in the time zone data to indicate additional identifiers that have the matching time zone identifier as an alias.

Parameters:

start=<date-time>

OPTIONAL, and MUST NOT occur more than once. Specifies the inclusive UTC date-time value at which the returned time zone data is truncated at its start.

end=<date-time>

OPTIONAL, and MUST NOT occur more than once. Specifies the exclusive UTC date-time value at which the returned time zone data is truncated at its end.

Response: A document containing all the requested time zone data in the format specified.

Possible Error Codes:

urn:ietf:params:tzdist:error:tzid-not-found

No time zone associated with the specified "tzid" path segment value was found.

urn:ietf:params:tzdist:error:invalid-format

The Accept request header field supplied by the client did not contain a media type for time zone data supported by the server.

urn:ietf:params:tzdist:error:invalid-start

The "start" URI query parameter has an incorrect value, or appears more than once, or does not match one of the fixed truncation range start values advertised in the "capabilities" action response.

urn:ietf:params:tzdist:error:invalid-end

The "end" URI query parameter has an incorrect value, or appears more than once, or has a value less than or equal to the "start" URI query parameter, or does not match one of the fixed truncation range end values advertised in the "capabilities" action response.

[5.3.1.](#) Example: Get Time Zone Data

In this example, the client requests that the time zone with a specific time zone identifier be returned.

>> Request <<

```
GET /servlet/timezone/zones/America%2FNew_York HTTP/1.1
Host: tz.example.com
Accept:text/calendar
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: text/calendar; charset="utf-8"
Content-Length: xxxx
ETag: "123456789-000-111"
```

BEGIN:VCALENDAR

...

BEGIN:VTIMEZONE

TZID:America/New_York

...

END:VTIMEZONE
END:VCALENDAR

[5.3.2.](#) Example: Conditional Get Time Zone Data

In this example the client requests that the time zone with a specific time zone identifier be returned, but uses an If-None-Match header field in the request, set to the value of a previously returned ETag header field, or the value of the "etag" member in a JSON "timezone" object returned from a "list" action response. In this example, the data on the server has not changed, so a 304 response is returned.

>> Request <<

```
GET /servlet/timezone/zones/America%2FNew_York HTTP/1.1
Host: tz.example.com
Accept:text/calendar
If-None-Match: "123456789-000-111"
```

>> Response <<

```
HTTP/1.1 304 Not Modified
Date: Wed, 4 Jun 2008 09:32:12 GMT
```

[5.3.3.](#) Example: Get Time Zone Data Using a Time Zone Alias

In this example, the client requests that the time zone with an aliased time zone identifier be returned, and the server returns the time zone data with that identifier and two aliases.

>> Request <<

```
GET /servlet/timezone/zones/US%2FEastern HTTP/1.1
Host: tz.example.com
Accept:text/calendar
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: text/calendar; charset="utf-8"
Content-Length: xxxx
ETag: "123456789-000-111"
```

```
BEGIN:VCALENDAR
...
BEGIN:VTIMEZONE
TZID:US/Eastern
TZID-ALIAS-OF:America/New_York
TZID-ALIAS-OF:America/Montreal
```

```
...
END:VTIMEZONE
END:VCALENDAR
```

[5.3.4.](#) Example: Get Truncated Time Zone Data

Assume the server advertises a "truncated" object in its "capabilities" response that appears as:

```
"truncated": {
  "any": false,
  "ranges": [
    {"start": "1970-01-01T00:00:00Z", "end": "*"},
    {"start": "2010-01-01T00:00:00Z", "end": "2020-01-01T00:00:00Z"}
  ],
  "untruncated": false
}
```

In this example, the client requests that the time zone with a specific time zone identifier truncated at one of the ranges specified by the server be returned. Note the presence of a "STANDARD" component that matches the start point of the truncation range (converted to the local time for the UTC offset in effect at the matching UTC time). Also, note the presence of the "TZUNTIL" ([Section 7.1](#)) iCalendar property in the "VTIMEZONE" component, indicating the upper bound on the validity period of the time zone data.

>> Request <<

```
GET /servlet/timezone/zones/America%2FNew_York
?start=2010-01-01T00:00:00Z&end=2020-01-01T00:00:00Z HTTP/1.1
Host: tz.example.com
Accept:text/calendar
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: text/calendar; charset="utf-8"
Content-Length: xxxx
ETag: "123456789-000-111"
```

```
BEGIN:VCALENDAR
...
BEGIN:VTIMEZONE
TZID:America/New_York
TZUNTIL:20200101T000000Z
```

```
BEGIN:STANDARD
DTSTART:20101231T190000
TZNAME:EST
TZOFFSETFROM:-0500
TZOFFSETTO:-0500
END:STANDARD
...
END:VTIMEZONE
END:VCALENDAR
```

[5.3.5.](#) Example: Request for a Nonexistent Time Zone

In this example, the client requests that the time zone with a specific time zone identifier be returned. As it turns out, no time zone exists with that identifier.

>> Request <<

```
GET /servlet/timezone/zones/America%2FPittsburgh HTTP/1.1
Host: tz.example.com
Accept:application/calendar+json
```

>> Response <<

```
HTTP/1.1 404 Not Found
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: application/problem+json; charset="utf-8"
Content-Language: en
Content-Length: xxxx
```

```
{
  "type": "urn:ietf:params:tzdist:error:tzid-not-found",
  "title": "Time zone identifier was not found on this server",
  "status": 404
}
```

[5.4.](#) "expand" Action

Name: expand

Request-URI Template:

```
{/service-prefix,data-prefix}/zones{/tzid}/observances{?start,end}
```

The "tzid" variable value is REQUIRED.

Description: This action expands the specified time zone into a list of onset start date/time values (in UTC) and UTC offsets. The response MUST contain an ETag response header field indicating the current value of the strong entity tag of the time zone being expanded.

Parameters:

start=<date-time>: REQUIRED, and MUST occur only once. Specifies the inclusive UTC date-time value for the start of the period of interest.

end=<date-time>: REQUIRED, and MUST occur only once. Specifies the exclusive UTC date-time value for the end of the period of

interest. Note that this is the exclusive end value, i.e., it represents the date just after the range of interest. For if a client wants the expanded date just for the year 2014, it would use a start value of "2014-01-01T00:00:00Z" and an end value of "2015-01-01T00:00:00Z". An error occurs if the end value is less than or equal to the start value.

Response: A JSON object containing a "tzid" member and an "observances" member; see [Section 6.3](#). If the time zone being expanded is not fully defined over the requested time range (e.g., because of truncation), then the server MUST include "start" and/or "end" members in the JSON response to indicate the actual start and end points for the observances being returned. The server MUST include an expanded observance representing the time zone information in effect at the start of the returned observance period.

Possible Error Codes

urn:ietf:params:tzdist:error:tzid-not-found

No time zone associated with the specified "tzid" path segment value was found.

urn:ietf:params:tzdist:error:invalid-start

The "start" URI query parameter has an incorrect value, or appears more than once, or is missing, or has a value outside any fixed truncation ranges advertised in the "capabilities" action response.

urn:ietf:params:tzdist:error:invalid-end

The "end" URI query parameter has an incorrect value, or appears more than once, or has a value less than or equal to

the "start" URI query parameter, or has a value outside any fixed truncation ranges advertised in the "capabilities" action response.

[5.4.1](#). Example: Expanded JSON Data Format

In this example, the client requests a time zone in the expanded

form.

>> Request <<

```
GET /servlet/timezone/zones/America%2FNew_York/observances
?start=2008-01-01T00:00:00Z&end=2009-01-01T00:00:00Z HTTP/1.1
Host: tz.example.com
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Mon, 11 Oct 2009 09:32:12 GMT
Content-Type: application/json; charset="utf-8"
Content-Length: xxxx
ETag: "123456789-000-111"
```

```
{
  "tzid": "America/New_York",
  "observances": [
    {
      "name": "Standard",
      "onset": "2008-01-01T00:00:00Z",
      "utc-offset-from": -18000,
      "utc-offset-to": -18000
    },
    {
      "name": "Daylight",
      "onset": "2008-03-09T07:00:00Z",
      "utc-offset-from": -18000,
      "utc-offset-to": -14400
    },
    {
      "name": "Standard",
      "onset": "2008-11-02T06:00:00Z",
      "utc-offset-from": -14400,
      "utc-offset-to": -18000
    },
  ]
}
```

[5.5.](#) "find" Action

Name: find

Request-URI Template:

```
{/service-prefix,data-prefix}/zones{?pattern}
```

Description: This action allows a client to query the time zone data distribution service for a matching identifier, alias, or localized name, using a simple "glob" style pattern match against the names known to the server (with an asterisk (*) as the wildcard character). Pattern-match strings (which have to be percent-encoded and then decoded when used in the URI query parameter) have the following options:

- * not present: An exact text match is done, e.g., "xyz"
- * first character only: An ends-with text match is done, e.g., "xyz"
- * last character only: A starts-with text match is done, e.g., "xyz"
- * first and last characters only: A substring text match is done, e.g., "xyz"

Escaping \ and *: To match 0x2A ("*") and 0x5C ("\") characters in a time zone identifier, those characters have to be "escaped" in the pattern by prepending a single 0x5C ("\") character. For example, a pattern "*Test\\Time*Zone\\" is used for an exact match against the time zone identifier "*Test\Time*Zone*". An unescaped "*" character MUST NOT appear in the middle of the string and MUST result in an error. An unescaped "\" character MUST NOT appear anywhere in the string and MUST result in an error.

In addition, when matching:

Underscores: Underscore characters (0x5F) in time zone identifiers MUST be mapped to a single space character (0x20) prior to string comparison in both the pattern and time zone identifiers being matched. This allows time zone identifiers such as "America/New_York" to match a query for "*New York*".

Case mapping: ASCII characters in the range 0x41 ("A") through 0x5A ("Z") MUST be mapped to their lowercase equivalents in both the pattern and time zone identifiers being matched.

Parameters:

pattern=<text>
REQUIRED, and MUST occur only once.

Response: The response has the same format as the "list" action, with one result object per successful match; see [Section 6.2](#).

Possible Error Codes

urn:ietf:params:tzdist:error:invalid-pattern
The "pattern" URI query parameter has an incorrect value or appears more than once.

[5.5.1](#). Example: find action

In this example, the client asks for data about the time zone "US/Eastern".

>> Request <<

```
GET /servlet/timezone/zones?pattern=US/Eastern HTTP/1.1
Host: tz.example.com
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: application/json; charset="utf-8"
Content-Length: xxxx
```

```
{
  "synctoken": "2009-10-11T09:32:11Z",
  "timezones": [
    {
      "tzid": "America/New_York",
      "etag": "123456789-000-111",
      "last-modified": "2009-09-17T01:39:34Z",
      "publisher": "Example.com",
      "version": "2015a",
      "aliases": ["US/Eastern"],
      "local-names": [
        {
```

```

        "name": "America/New_York",
        "lang": "en_US"
    }
]
},

```

```

{
  "tzid": "America/Detroit",
  "etag": "123456789-999-222",
  "last-modified": "2009-09-17T01:39:34Z",
  "publisher": "Example.com",
  "version": "2015a",
  "aliases": ["US/Eastern"],
  "local-names": [
    {
      "name": "America/Detroit",
      "lang": "en_US"
    }
  ]
},
...
]
}

```

[5.6.](#) "leapseconds" Action

Name: leapseconds

Request-URI Template:

```
{/service-prefix,data-prefix}/leapseconds
```

Description: This action allows a client to query the time zone data distribution service to retrieve the current leap-second information available on the server.

Parameters: None

Response: A JSON object containing an "expires" member, a "publisher" member, a "version" member, and a "leapseconds" member; see [Section 6.4](#). The "expires" member in the JSON response indicates the latest date covered by leap-second information. For example (as in [Section 5.6.1](#)), if the "expires"

value is set to "2014-06-28" and the latest leap-second change indicated was at "2012-07-01", then the data indicates that there are no leap seconds added (or removed) between those two dates, and information for leap seconds beyond the "expires" date is not yet available.

The "leapseconds" member contains a list of JSON objects each of which contains a "utc-offset" and "onset" member. The "onset" member specifies the date (with the implied time of 00:00:00 UTC) at which the corresponding UTC offset from TAI takes effect. In other words, a leap second is added or removed just prior to time 00:00:00 UTC of the specified onset date. When a leap second is

added, the "utc-offset" value will be incremented by one; when a leap second is removed, the "utc-offset" value will be decremented by one.

Possible Error Codes No specific code.

[5.6.1.](#) Example: Get Leap-Second Information

In this example, the client requests the current leap-second information from the server.

>> Request <<

```
GET /servlet/timezone/leapseconds HTTP/1.1
Host: tz.example.com
```

>> Response <<

```
HTTP/1.1 200 OK
Date: Wed, 4 Jun 2008 09:32:12 GMT
Content-Type: application/json; charset="utf-8"
Content-Length: xxxx
```

```
{
  "expires": "2015-12-28",
  "publisher": "Example.com",
  "version": "2015d",
  "leapseconds": [
    {
```

```

    "utc-offset": 10,
    "onset": "1972-01-01",
  },
  {
    "utc-offset": 11,
    "onset": "1972-07-01",
  },
  ...
  {
    "utc-offset": 35,
    "onset": "2012-07-01",
  },
  {
    "utc-offset": 36,
    "onset": "2015-07-01",
  }
]
}

```

[6.](#) JSON Definitions

[RFC7159] defines the structure of JSON objects using a set of primitive elements. The structure of JSON objects used by this specification is described by the following set of rules:

OBJECT represents a JSON object, defined in [Section 4 of \[RFC7159\]](#). "OBJECT" is followed by a parenthesized list of "MEMBER" rule names. If a member rule name is preceded by a "?" (0x3F) character, that member is optional; otherwise, all members are required. If two or more member rule names are present, each separated from the other by a "|" (0x7C) character, then only one of those members MUST be present in the JSON object. JSON object members are unordered, and thus the order used in the rules is not significant.

MEMBER represents a member of a JSON object, defined in [Section 4 of \[RFC7159\]](#). "MEMBER" is followed by a rule name, the name of the member, a ":", and then the value. A value can be one of "OBJECT", "ARRAY", "NUMBER", "STRING", or "BOOLEAN" rules.

ARRAY represents a JSON array, defined in [Section 5 of \[RFC7159\]](#).

"ARRAY" is followed by a value (one of "OBJECT", "ARRAY", "NUMBER", "STRING", or "BOOLEAN"), indicating the type of items used in the array.

NUMBER represents a JSON number, defined in [Section 6 of \[RFC7159\]](#).

STRING represents a JSON string, defined in [Section 7 of \[RFC7159\]](#).

BOOLEAN represents either of the JSON values "true" or "false", defined in [Section 3 of \[RFC7159\]](#).

; a line starting with a ";" (0x3B) character is a comment.

Note, clients MUST ignore any unexpected JSON members in responses from the server.

[6.1.](#) capabilities Action Response

Below are the rules for the JSON document returned for a "capabilities" action request.

; root object

OBJECT (version, info, actions)

; The version number of the protocol supported - MUST be 1

MEMBER version "version" : NUMBER

; object containing service information

; Only one of primary_source or secondary_source MUST be present

MEMBER info "info" : OBJECT (
 primary_source | secondary_source,
 formats,
 ?truncated,
 ?provider_details,
 ?contacts
)

; The source of the time zone data provided by a "primary" server

MEMBER primary_source "primary-source" : STRING

; The time zone data server from which data is provided by a

; "secondary" server

```

MEMBER secondary_source "secondary-source" : STRING

; Array of one or more media types for the time zone data formats
; that the server can return
MEMBER formats "formats" : ARRAY STRING

; Present if the server is providing truncated time zone data. The
; value is an object providing details of the supported truncation
; modes.
MEMBER truncated "truncated" : OBJECT: (
    any,
    ?ranges,
    ?untruncated
)

; Indicates whether the server can truncate time zone data at any
; start or end point. When set to "true", any start or end point is
; a valid value for use with the "start" and "end" URI query
; parameters in a "get" action request.
MEMBER any "any" : BOOLEAN

; Indicates which ranges of time the server has truncated data for.
; A value from this list may be used with the "start" and "end" URI
; query parameters in a "get" action request. Not present if "any"
; is set to "true".
MEMBER ranges "ranges" : ARRAY OBJECT (range-start, range-end)

; UTC date-time value (per [RFC3339]) for inclusive start of the
; range, or the single character "*" to indicate a value
; corresponding to the lower bound supplied by the publisher of the
; time zone data
MEMBER range-start "start" : STRING

```

```

; UTC date-time value (per [RFC3339]) for exclusive end of the range,
; or the single character "*" to indicate a value corresponding to
; the upper bound supplied by the publisher of the time zone data
MEMBER range-end "end" : STRING

; Indicates whether the server can supply untruncated data. When
; set to "true", indicates that, in addition to truncated data being
; available, the server can return untruncated data if a "get"

```



```

; action request is executed without a "start" or "end" URI query
; parameter.
MEMBER untruncated "untruncated" : BOOLEAN

; A URI where human-readable details about the time zone service
; is available
MEMBER provider_details "provider-details" : STRING

; Array of URIs providing contact details for the server
; administrator
MEMBER contacts "contacts" : ARRAY STRING

; Array of actions supported by the server
MEMBER actions "actions" : ARRAY OBJECT (
    action_name,
    action_params
)

; Name of the action
MEMBER action_name: "name" : STRING

; Array of request-URI query parameters supported by the action
MEMBER action_params: "parameters" ARRAY OBJECT (
    param_name,
    ?param_required,
    ?param_multi,
    ?param_values
)

; Name of the parameter
MEMBER param_name "name" : STRING

; If true, the parameter has to be present in the request-URI
; default is false
MEMBER param_required "required" : BOOLEAN

; If true, the parameter can occur more than once in the request-URI
; default is false
MEMBER param_multi "multi" : BOOLEAN,

```

```

; An array that defines the allowed set of values for the parameter

```

; In the absence of this member, any string value is acceptable
MEMBER param_values "values" ARRAY STRING

[6.2.](#) list/find Action Response

Below are the rules for the JSON document returned for a "list" or "find" action request.

; root object
OBJECT (synctoken, timezones)

; Server-generated opaque token used for synchronizing changes
MEMBER synctoken "synctoken" : STRING

; Array of time zone objects
MEMBER timezones "timezones" : ARRAY OBJECT (
 tzid,
 etag,
 last_modified,
 publisher,
 version,
 ?aliases,
 ?local_names,
)

; Time zone identifier
MEMBER tzid "tzid" : STRING

; Current ETag for the corresponding time zone data resource
MEMBER etag "etag" : STRING

; Date/time when the time zone data was last modified
; UTC date-time value as specified in [[RFC3339](#)]
MEMBER last_modified "last-modified" : STRING

; Time zone data publisher
MEMBER publisher "publisher" : STRING

; Current version of the time zone data as defined by the
; publisher
MEMBER version "version" : STRING

; An array that lists the set of time zone aliases available
; for the corresponding time zone
MEMBER aliases "aliases" : ARRAY STRING

```
; An array that lists the set of localized names available
; for the corresponding time zone
MEMBER local_names "local-names" : ARRAY OBJECT (
  lname, lang, ?pref
)

; Language tag for the language of the associated name
MEMBER: lang "lang" : STRING

; Localized name
MEMBER lname "name" : STRING

; Indicates whether this is the preferred name for the associated
; language default: false
MEMBER pref "pref" : BOOLEAN
```

[6.3.](#) expand Action Response

Below are the rules for the JSON document returned for a "expand" action request.

```
; root object
OBJECT (
  tzid,
  ?start,
  ?end,
  observances
)

; Time zone identifier
MEMBER tzid "tzid" : STRING

; The actual inclusive start point for the returned observances
; if different from the value of the "start" URI query parameter
MEMBER start "start" : STRING

; The actual exclusive end point for the returned observances
; if different from the value of the "end" URI query parameter
MEMBER end "end" : STRING

; Array of time zone objects
MEMBER observances "observances" : ARRAY OBJECT (
  oname,
  ?olocal_names,
  onset,
  utc_offset_from,
```

```
    utc_offset_to
)
```

```
; Observance name
MEMBER oname "name" : STRING

; Array of localized observance names
MEMBER olocal_names "local-names" : ARRAY STRING

; UTC date-time value (per [RFC3339]) at which the observance takes
; effect
MEMBER onset "onset" : STRING

; The UTC offset in seconds before the start of this observance
MEMBER utc_offset_from "utc-offset-from" : NUMBER

; The UTC offset in seconds at and after the start of this observance
MEMBER utc_offset_to "utc-offset-to" : NUMBER
```

[6.4.](#) leapseconds Action Response

Below are the rules for the JSON document returned for a "leapseconds" action request.

```
; root object
OBJECT (
    expires,
    publisher,
    version,
    leapseconds
)

; Last valid date covered by the data in this response
; full-date value as specified in [RFC3339]
MEMBER expires "expires" : STRING

; Leap-second information publisher
MEMBER publisher "publisher" : STRING

; Current version of the leap-second information as defined by the
; publisher
MEMBER version "version" : STRING
```

```
; Array of leap-second objects
MEMBER leapseconds "leapseconds" : ARRAY OBJECT (
    utc_offset,
    onset
)
```

```
; The UTC offset from TAI in seconds in effect at and after the
; specified date
MEMBER utc_offset "utc-offset" : NUMBER
```

```
; full-date value (per [RFC3339]) at which the new UTC offset takes
; effect, at T00:00:00Z
MEMBER onset "onset" : STRING
```

[7.](#) New iCalendar Properties

[7.1.](#) Time Zone Upper Bound

Property Name: TZUNTIL

Purpose: This property specifies an upper bound for the validity period of data within a "VTIMEZONE" component.

Value Type: DATE-TIME

Property Parameters: IANA and non-standard property parameters can be specified on this property.

Conformance: This property can be specified zero times or one time within "VTIMEZONE" calendar components.

Description: The value MUST be specified in the UTC time format.

Time zone data in a "VTIMEZONE" component might cover only a fixed period of time. The start of such a period is clearly indicated by the earliest observance defined by the "STANDARD" and "DAYLIGHT" subcomponents. However, an upper bound on the validity period of the time zone data cannot be simply derived from the

observance with the latest onset time, and [\[RFC5545\]](#) does not define a way to get such an upper bound. This specification introduces the "TZUNTIL" property for that purpose. It specifies an "exclusive" UTC date-time value that indicates the last time at which the time zone data is to be considered valid.

This property is also used by time zone data distribution servers to indicate the truncation range end point of time zone data (as described in [Section 3.9](#)).

Format Definition: This property is defined by the following notation in ABNF [\[RFC5234\]](#):

```
tzuntil      = "TZUNTIL" tzuntilparam ":" date-time CRLF
tzuntilparam = *(";" other-param)
```

Example: Suppose a time zone based on astronomical observations has well-defined onset times through the year 2025, but the first onset in 2026 is currently known only approximately. In that case, the "TZUNTIL" property could be specified as follows:

```
TZUNTIL:20260101T000000Z
```

[7.2.](#) Time Zone Identifier Alias Property

Property Name: TZID-ALIAS-OF

Purpose: This property specifies a time zone identifier for which the main time zone identifier is an alias.

Value Type: TEXT

Property Parameters: IANA and non-standard property parameters can be specified on this property.

Conformance: This property can be specified zero or more times within "VTIMEZONE" calendar components.

Description: When the "VTIMEZONE" component uses a time zone identifier alias for the "TZID" property value, the "TZID-ALIAS-OF" property is used to indicate the time zone identifier of the

other time zone (see [Section 3.7](#)).

Format Definition: This property is defined by the following notation in ABNF [\[RFC5234\]](#):

```
tzid-alias-of      = "TZID-ALIAS-OF" tzidaliasofparam ":"  
                    [tzidprefix] text CRLF
```

```
tzidaliasofparam = *(";" other-param)
```

;tzidprefix defined in [\[RFC5545\]](#).

Example: The following is an example of this property:

TZID-ALIAS-OF:America/New_York

[8.](#) Security Considerations

Time zone data is critical in determining local or UTC time for devices and in calendaring and scheduling operations. As such, it is vital that a reliable source of time zone data is used. Servers providing a time zone data distribution service MUST support HTTP over Transport Layer Security (TLS) (as defined by [\[RFC2818\]](#) and [\[RFC5246\]](#), with best practices described in [\[RFC7525\]](#)). Servers MAY support a time zone data distribution service over HTTP without TLS. However, secondary servers MUST use TLS to fetch data from a primary server.

Clients SHOULD use Transport Layer Security as defined by [\[RFC2818\]](#), unless they are specifically configured otherwise. Clients that have been configured to use the TLS-based service MUST NOT fall back to using the non-TLS service if the TLS-based service is not available. In addition, clients MUST NOT follow HTTP redirect requests from a TLS service to a non-TLS service. When using TLS, clients MUST

verify the identity of the server, using a standard, secure mechanism such as the certificate verification process specified in [[RFC6125](#)] or DANE [[RFC6698](#)].

A malicious attacker with access to the DNS server data, or able to get spoofed answers cached in a recursive resolver, can potentially cause clients to connect to any server chosen by the attacker. In the absence of a secure DNS option, clients SHOULD check that the target FQDN returned in the SRV record is the same as the original service domain that was queried, or is a sub-domain of the original service domain. In many cases, the client configuration is likely to be handled automatically without any user input; as such, any mismatch between the original service domain and the target FQDN is treated as a failure and the client MUST NOT attempt to connect to the target server. In addition, when Transport Layer Security is being used, the Transport Layer Security certificate SHOULD include an SRV-ID field as per [[RFC4985](#)] matching the expected DNS SRV queries clients will use for service discovery. If an SRV-ID field is present in a certificate, clients MUST match the SRV-ID value with the service type and domain that matches the DNS SRV request made by the client to discover the service.

Time zone data servers SHOULD protect themselves against poorly implemented or malicious clients by throttling high request rates or frequent requests for large amounts of data. Clients can avoid being throttled by using the polling capabilities outlined in [Section 4.1.4](#). Servers MAY require some form of authentication or authorization of clients (including secondary servers), as per [[RFC7235](#)], to restrict which clients are allowed to access their service or provide better identification of problematic clients.

[9.](#) Privacy Considerations

The type and pattern of requests that a client makes can be used to "fingerprint" specific clients or devices and thus potentially used to track information about what the users of the clients might be doing. In particular, a client that only downloads time zone data on an as-needed basis, will leak the fact that a user's device has moved from one time zone to another or that the user is receiving scheduling messages from another user in a different time zone.

Clients need to be aware of the potential ways in which an untrusted

server or a network observer might be able to track them and take precautions such as the following:

1. Always use TLS to connect to the server.
2. Avoid use of TLS session resumption.
3. Always fetch and synchronize the entire set of time zone data to avoid leaking information about which time zones are actually in use by the client.
4. Randomize the order in which individual time zones are fetched using the "get" action, when retrieving a set of time zones based on a "list" action response.
5. Avoid use of conditional HTTP requests [[RFC7232](#)] with the "get" action to prevent tracking of clients by servers generating client-specific ETag header field values.
6. Avoid use of cookies in HTTP requests [[RFC6265](#)].
7. Avoid use of authenticated HTTP requests.
8. When doing periodic polling to check for updates, apply a random (positive or negative) offset to the next poll time to avoid servers being able to identify the client by the specific periodicity of its polling behavior.
9. A server trying to "fingerprint" clients might insert a "fake" time zone into the time zone data, using a unique identifier for each client making a request. The server can then watch for client requests that refer to that "fake" time zone and thus track the activity of each client. It is hard for clients to identify a "fake" time zone given that new time zones are added occasionally. One option to mitigate this would be for the client to make use of two time zone data distribution servers from two independent providers that provide time zone data from

the same publisher. The client can then compare the list of time zones from each server (assuming they both have the same version of time zone data from the common publisher) and detect ones that appear to be added on one server and not the other.

Alternatively, the client can check the publisher data directly to verify that time zones match the set the publisher has.

Note that some of the above recommendations will result in less efficient use of the protocol due to fetching data that might not be relevant to the client.

An organization can set up a secondary server within their own domain and configure their clients to use that server to protect the organization's users from the possibility of being tracked by an untrusted time zone data distribution server. Clients can then use more-efficient protocol interactions, free from the concerns above, on the basis that their organization's server is trusted. When doing this, the secondary server would follow the recommendations for clients (listed in the previous paragraph) so that the untrusted server is not able to gain information about the organization as a whole. Note, however, that client requests to the secondary server are subject to tracking by a network observer, so clients ought to apply some of the randomization techniques from the list above.

Servers that want to avoid accidentally storing information that could be used to identify clients can take the following precautions:

1. Avoid logging client request activity, or anonymize information in any logs (e.g., client IP address, client user-agent details, authentication credentials, etc.).
2. Add an unused HTTP response header to each response with a random amount of data in it (e.g., to pad the overall request size to the nearest power-of-2 or 128-byte boundary) to avoid exposing which time zones are being fetched when TLS is being used, via network traffic analysis.

10. IANA Considerations

This specification defines a new registry of "actions" for the time zone data distribution service protocol, defines a "well-known" URI using the registration procedure and template from [Section 5.1 of \[RFC5785\]](#), creates two new SRV service label aliases, and defines one new iCalendar property parameter as per the registration procedure in [\[RFC5545\]](#). It also adds a new "TZDIST Identifiers Registry" to the IETF parameters URN sub-namespace as per [\[RFC3553\]](#) for use with protocol related error codes.

10.1. Service Actions Registration

IANA has created a new top-level category called "Time Zone Data Distribution Service (TZDIST) Parameters" and has put all the registries created herein into that category.

IANA has created a new registry called "TZDIST Service Actions", as defined below.

10.1.1. Service Actions Registration Procedure

This registry uses the "Specification Required" policy defined in [[RFC5226](#)], which makes use of a designated expert to review potential registrations.

The IETF has created a mailing list, tzdist-service@ietf.org, which is used for public discussion of time zone data distribution service actions proposals prior to registration. The IESG has appointed a designated expert who will monitor the tzdist-service@ietf.org mailing list and review registrations.

A Standards Track RFC is REQUIRED for changes to actions previously documented in a Standards Track RFC; otherwise, any public specification that satisfies the requirements of [[RFC5226](#)] is acceptable.

The registration procedure begins when a completed registration template, as defined below, is sent to tzdist-service@ietf.org and iana@iana.org. The designated expert is expected to tell IANA and the submitter of the registration whether the registration is approved, approved with minor changes, or rejected with cause, within two weeks. When a registration is rejected with cause, it can be resubmitted if the concerns listed in the cause are addressed. Decisions made by the designated expert can be appealed as per [Section 7 of \[RFC5226\]](#).

The designated expert MUST take the following requirements into account when reviewing the registration:

1. A valid registration template MUST be provided by the submitter, with a clear description of what the action does.
2. A proposed new action name MUST NOT conflict with any existing registered action name. A conflict includes a name that duplicates an existing one or that appears to be very similar to an existing one and could be a potential source of confusion.

3. A proposed new action **MUST NOT** exactly duplicate the functionality of any existing actions. In cases where the new action functionality is very close to an existing action, the designated expert **SHOULD** clarify whether the submitter is aware of the existing action, and has an adequate reason for creating a new action with slight differences from an existing one.
4. If a proposed action is an extension to an existing action, the changes **MUST NOT** conflict with the intent of the existing action, or in a way that could cause interoperability problems for existing deployments of the protocol.

The IANA registry contains the name of the action ("Action Name") and a reference to the section of the specification where the action registration template is defined ("Reference").

[10.1.2.](#) Registration Template for Actions

An action is defined by completing the following template.

Name: The name of the action.

Request-URI Template: The URI template used in HTTP requests for the action.

Description: A general description of the action, its purpose, etc.

Parameters: A list of allowed request URI query parameters, indicating whether they are "REQUIRED" or "OPTIONAL" and whether they can occur only once or multiple times, together with the expected format of the parameter values.

Response: The nature of the response to the HTTP request, e.g., what format the response data is in.

Possible Error Codes: Possible error codes reported in a JSON "problem details" object if an HTTP request fails.

[10.1.3.](#) Actions Registry

The following table provides the initial content of the actions registry.

Action Name	Reference
capabilities	RFC 7808, Section 5.1
list	RFC 7808, Section 5.2
get	RFC 7808, Section 5.3
expand	RFC 7808, Section 5.4
find	RFC 7808, Section 5.5
leapseconds	RFC 7808, Section 5.6

[10.2.](#) timezone Well-Known URI Registration

IANA has added the following to the "Well-Known URIs" [[RFC5785](#)] registry:

URI suffix: timezone

Change controller: IESG.

Specification document(s): [RFC 7808](#)

Related information: None.

[10.3.](#) Service Name Registrations

IANA has added two new service names to the "Service Name and Transport Protocol Port Number Registry" [[RFC6335](#)], as defined below.

[10.3.1.](#) timezone Service Name Registration

Service Name: timezone

Transport Protocol(s): TCP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: Time Zone Data Distribution Service - non-TLS

Reference: [RFC 7808](#)

Douglass & Daboo

Standards Track

[Page 47]

[RFC 7808](#)

TZDIST Service

March 2016

Assignment Note: This is an extension of the http service. Defined
TXT keys: path=<context path> (as per [Section 6 of \[RFC6763\]](#)).

[10.3.2.](#) timezones Service Name Registration

Service Name: timezones

Transport Protocol(s): TCP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: Time Zone Data Distribution Service - over TLS

Reference: [RFC 7808](#)

Assignment Note: This is an extension of the https service. Defined
TXT keys: path=<context path> (as per [Section 6 of \[RFC6763\]](#)).

[10.4.](#) TZDIST Identifiers Registry

IANA has registered a new URN sub-namespace within the IETF URN Sub-namespace for Registered Protocol Parameter Identifiers defined in [\[RFC3553\]](#).

Registrations in this registry follow the "IETF Review" [[RFC5226](#)] policy.

Registry name: TZDIST Identifiers

URN prefix: urn:ietf:params:tzdist

Specification: [RFC 7808](#)

Repository: <http://www.iana.org/assignments/tzdist-identifiers>

Index value: Values in this registry are URNs or URN prefixes that start with the prefix "urn:ietf:params:tzdist:". Each is registered independently. The prefix "urn:ietf:params:tzdist:error:" is used to represent specific error codes within the protocol as defined in the list of actions in [Section 5](#) and used in problem reports ([Section 4.1.7](#)).

Each registration in the "TZDIST Identifiers" registry requires the following information:

URN: The complete URN that is used or the prefix for that URN.

Description: A summary description for the URN or URN prefix.

Specification: A reference to a specification describing the URN or URN prefix.

Contact: Email for the person or groups making the registration.

Index Value: As described in [[RFC3553](#)], URN prefixes that are registered include a description of how the URN is constructed. This is not applicable for specific URNs.

The "TZDIST Identifiers" registry has the initial registrations included in the following sections.

[10.4.1](#). Registration of invalid-action Error URN

The following URN has been registered in the "tzdist Identifiers" registry.

URN: urn:ietf:params:tzdist:error:invalid-action

Description: Generic error code for any invalid action.

Specification: [RFC 7808, Section 5](#)

Repository: <http://www.iana.org/assignments/tzdist-identifiers>

Contact: IESG <iesg@ietf.org>

Index value: N/A.

[10.4.2.](#) Registration of invalid-changedsince Error URN

The following URN has been registered in the "tzdist Identifiers" registry.

URN: urn:ietf:params:tzdist:error:invalid-changedsince

Description: Error code for incorrect use of the "changedsince" URI query parameter.

Specification: [RFC 7808, Section 5.2](#)

Repository: <http://www.iana.org/assignments/tzdist-identifiers>

Contact: IESG <iesg@ietf.org>

Index value: N/A.

[10.4.3.](#) Registration of tzid-not-found Error URN

The following URN has been registered in the "tzdist Identifiers" registry.

URN: urn:ietf:params:tzdist:error:tzid-not-found

Description: Error code for missing time zone identifier.

Specification: [RFC 7808](#), Sections [5.3](#) and [5.4](#)

Repository: <http://www.iana.org/assignments/tzdist-identifiers>

Contact: IESG <iesg@ietf.org>

Index value: N/A.

10.4.4. Registration of invalid-format Error URN

The following URN has been registered in the "tzdist Identifiers" registry.

URN: urn:ietf:params:tzdist:error:invalid-format

Description: Error code for unsupported HTTP Accept request header field value.

Specification: [RFC 7808, Section 5.3](#)

Repository: <http://www.iana.org/assignments/tzdist-identifiers>

Contact: IESG <iesg@ietf.org>

Index value: N/A.

10.4.5. Registration of invalid-start Error URN

The following URN has been registered in the "tzdist Identifiers" registry.

URN: urn:ietf:params:tzdist:error:invalid-start

Description: Error code for incorrect use of the "start" URI query parameter.

Specification: [RFC 7808](#), Sections [5.3](#) and [5.4](#)

Repository: <http://www.iana.org/assignments/tzdist-identifiers>

Contact: IESG <iesg@ietf.org>

Index value: N/A.

10.4.6. Registration of invalid-end Error URN

The following URN has been registered in the "tzdist Identifiers" registry.

URN: urn:ietf:params:tzdist:error:invalid-end

Description: Error code for incorrect use of the "end" URI query parameter.

Specification: [RFC 7808](#), Sections [5.3](#) and [5.4](#)

Repository: <http://www.iana.org/assignments/tzdist-identifiers>

Contact: IESG <iesg@ietf.org>

Index value: N/A.

[10.4.7.](#) Registration of invalid-pattern Error URN

The following URN has been registered in the "tzdist Identifiers" registry.

URN: urn:ietf:params:tzdist:error:invalid-pattern

Description: Error code for incorrect use of the "pattern" URI query parameter.

Specification: [RFC 7808, Section 5.5](#)

Repository: <http://www.iana.org/assignments/tzdist-identifiers>

Contact: IESG <iesg@ietf.org>

Index value: N/A.

10.5. iCalendar Property Registrations

This document defines the following new iCalendar properties, which have been added to the "Properties" registry under "iCalendar Element Registries" [[RFC5545](#)]:

Property	Status	Reference
TZUNTIL	Current	RFC 7808, Section 7.1
TZID-ALIAS-OF	Current	RFC 7808, Section 7.2

11. References

11.1. Normative References

- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", [RFC 2046](#), DOI 10.17487/RFC2046, November 1996, <<http://www.rfc-editor.org/info/rfc2046>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", [RFC 2782](#), DOI 10.17487/RFC2782, February 2000, <<http://www.rfc-editor.org/info/rfc2782>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", [RFC 2818](#), DOI 10.17487/RFC2818, May 2000, <<http://www.rfc-editor.org/info/rfc2818>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", [RFC 3339](#), DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.
- [RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", [BCP 73](#), [RFC 3553](#), DOI 10.17487/RFC3553, June 2003, <<http://www.rfc-editor.org/info/rfc3553>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, [RFC 3629](#), DOI 10.17487/RFC3629, November 2003, <<http://www.rfc-editor.org/info/rfc3629>>.

[RFC 7808](#)

TZDIST Service

March 2016

- [RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name", [RFC 4985](#), DOI 10.17487/RFC4985, August 2007, <<http://www.rfc-editor.org/info/rfc4985>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, [RFC 5234](#), DOI 10.17487/RFC5234, January 2008, <<http://www.rfc-editor.org/info/rfc5234>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", [RFC 5545](#), DOI 10.17487/RFC5545, September 2009, <<http://www.rfc-editor.org/info/rfc5545>>.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", [RFC 5785](#), DOI 10.17487/RFC5785, April 2010, <<http://www.rfc-editor.org/info/rfc5785>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", [RFC 6125](#), DOI 10.17487/RFC6125, March 2011, <<http://www.rfc-editor.org/info/rfc6125>>.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", [RFC 6265](#), DOI 10.17487/RFC6265, April 2011, <<http://www.rfc-editor.org/info/rfc6265>>.
- [RFC6321] Daboo, C., Douglass, M., and S. Lees, "xCal: The XML

Format for iCalendar", [RFC 6321](#), DOI 10.17487/RFC6321, August 2011, <<http://www.rfc-editor.org/info/rfc6321>>.

[RFC 7808](#)

TZDIST Service

March 2016

- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", [BCP 165](#), [RFC 6335](#), DOI 10.17487/RFC6335, August 2011, <<http://www.rfc-editor.org/info/rfc6335>>.
- [RFC6557] Lear, E. and P. Eggert, "Procedures for Maintaining the Time Zone Database", [BCP 175](#), [RFC 6557](#), DOI 10.17487/RFC6557, February 2012, <<http://www.rfc-editor.org/info/rfc6557>>.
- [RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", [RFC 6570](#), DOI 10.17487/RFC6570, March 2012, <<http://www.rfc-editor.org/info/rfc6570>>.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", [RFC 6698](#), DOI 10.17487/RFC6698, August 2012, <<http://www.rfc-editor.org/info/rfc6698>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", [RFC 6763](#), DOI 10.17487/RFC6763, February 2013, <<http://www.rfc-editor.org/info/rfc6763>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", [RFC 7159](#), DOI 10.17487/RFC7159, March 2014, <<http://www.rfc-editor.org/info/rfc7159>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", [RFC 7230](#), DOI 10.17487/RFC7230, June 2014, <<http://www.rfc-editor.org/info/rfc7230>>.

- [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, <<http://www.rfc-editor.org/info/rfc7231>>.
- [RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", [RFC 7232](#), DOI 10.17487/RFC7232, June 2014, <<http://www.rfc-editor.org/info/rfc7232>>.

- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", [RFC 7234](#), DOI 10.17487/RFC7234, June 2014, <<http://www.rfc-editor.org/info/rfc7234>>.
- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", [RFC 7235](#), DOI 10.17487/RFC7235, June 2014, <<http://www.rfc-editor.org/info/rfc7235>>.
- [RFC7265] Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON Format for iCalendar", [RFC 7265](#), DOI 10.17487/RFC7265, May 2014, <<http://www.rfc-editor.org/info/rfc7265>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", [BCP 195](#), [RFC 7525](#), DOI 10.17487/RFC7525, May 2015, <<http://www.rfc-editor.org/info/rfc7525>>.
- [RFC7807] Nottingham, M. and E. Wilde, "Problem Details for HTTP APIs", [RFC 7807](#), DOI 10.17487/RFC7807, March 2016, <<http://www.rfc-editor.org/info/rfc7807>>.

[11.2.](#) Informative References

- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", [RFC 2131](#), DOI 10.17487/RFC2131, March 1997,

<<http://www.rfc-editor.org/info/rfc2131>>.

Acknowledgements

The authors would like to thank the members of the Calendaring and Scheduling Consortium's Time Zone Technical Committee, and the participants and chairs of the IETF tzdist working group. In particular, the following individuals have made important contributions to this work: Steve Allen, Lester Caine, Stephen Colebourne, Tobias Conradi, Steve Crocker, Paul Eggert, Daniel Kahn Gillmor, John Haug, Ciny Joy, Bryan Keller, Barry Leiba, Andrew McMillan, Ken Murchison, Tim Parenti, Arnaud Quillaud, Jose Edvaldo Saraiva, and Dave Thewlis.

This specification originated from work at the Calendaring and Scheduling Consortium, which has supported the development and testing of implementations of the specification.

Authors' Addresses

Michael Douglass
Spherical Cow Group
226 3rd Street
Troy, NY 12180
United States

Email: mdouglass@sphericalcowgroup.com
URI: <http://sphericalcowgroup.com>

Cyrus Daboo
Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
United States

Email: cyrus@daboo.name
URI: <http://www.apple.com/>

