

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 13, 2015

M. Douglass
RPI
C. Daboo
Apple
December 10, 2014

Time Zone Data Distribution Service
draft-ietf-tzdist-service-04

Abstract

This document defines a time zone data distribution service that allows reliable, secure and fast delivery of time zone data to client systems such as calendaring and scheduling applications or operating systems.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on June 13, 2015.

Copyright Notice

Copyright (c) 2014 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	3
1.1.	Conventions	4
2.	Architectural Overview	4
3.	General Considerations	6
3.1.	Time Zone	6
3.2.	Time Zone Data	6
3.3.	Time Zone Data Server	7
3.4.	Observance	7
3.5.	Time Zone Identifiers	7
3.6.	Time Zone Aliases	7
3.7.	Time Zone Localized Names	8
3.8.	Truncating Time Zones	8
4.	Time Zone Data Distribution Service Protocol	9
4.1.	Server Protocol	9
4.1.1.	Time Zone Queries	10
4.1.2.	Time Zone Formats	10
4.1.3.	Time Zone Localization	10
4.1.4.	Conditional Time Zone Requests	10
4.1.5.	Expanded Time Zone Data	12
4.1.6.	Server Requirements	12
4.1.7.	Error Responses	12
4.1.8.	Extensions	12
4.2.	Client Guidelines	13
4.2.1.	Discovery	13
4.2.1.1.	Time Zone Data Distribution Service SRV Service Labels	13
4.2.1.2.	Time Zone Data Distribution Service TXT records	14
4.2.1.3.	Time Zone Data Distribution Service Well-Known URI	14
4.2.1.3.1.	Example: well-known URI redirects to actual context path	15
4.2.2.	Initial Synchronization of All Time Zones	15
4.2.3.	Subsequent Synchronization of All Time Zones	15
5.	Actions	15
5.1.	"capabilities" Action	16
5.1.1.	Example: Get Capabilities	16
5.2.	"list" Action	18
5.2.1.	Example: List time zone identifiers	19
5.3.	"get" Action	20
5.3.1.	Example: Get time zone data	22
5.3.2.	Example: Conditional Get time zone data	22
5.3.3.	Example: Get time zone data using a time zone alias	23
5.3.4.	Example: Get truncated time zone data	23
5.3.5.	Example: Get a non-existent time zone data	24
5.4.	"expand" Action	25
5.4.1.	Example: Expanded JSON Data Format	26

5.5.	"find" Action	27
5.5.1.	Example: Find action	28
6.	JSON Definitions	30
6.1.	capabilities action response	30
6.2.	list/find action response	32
6.3.	expand action response	34
7.	New iCalendar Properties	36
7.1.	Time Zone Upper Bound	36
7.2.	Time Zone Identifier Alias Property	37
8.	Security Considerations	37
9.	IANA Considerations	38
9.1.	Service Actions Registration	38
9.1.1.	Service Actions Registration Procedure	38
9.1.2.	Registration Template for Actions	39
9.2.	Initial Time Zone Data Distribution Service Registry	39
9.2.1.	Actions Registry	40
9.3.	timezone Well-Known URI Registration	40
9.4.	Service Name Registrations	40
9.4.1.	timezone Service Name Registration	40
9.4.2.	timezones Service Name Registration	41
9.5.	tzdist URN sub-namespace	41
9.6.	iCalendar Property Registrations	41
10.	Acknowledgements	42
11.	References	42
11.1.	Normative References	42
11.2.	Informative References	44
Appendix A.	Change History (to be removed prior to publication as an RFC)	44
	Authors' Addresses	47

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.

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

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

Discussion of this document should take place on the tzdist working group mailing list <tzdist@ietf.org>.

[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, [[RFC3339](#)] UTC date-time values use a "Z" suffix, and not fixed numeric offsets.

[2.](#) Architectural Overview

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

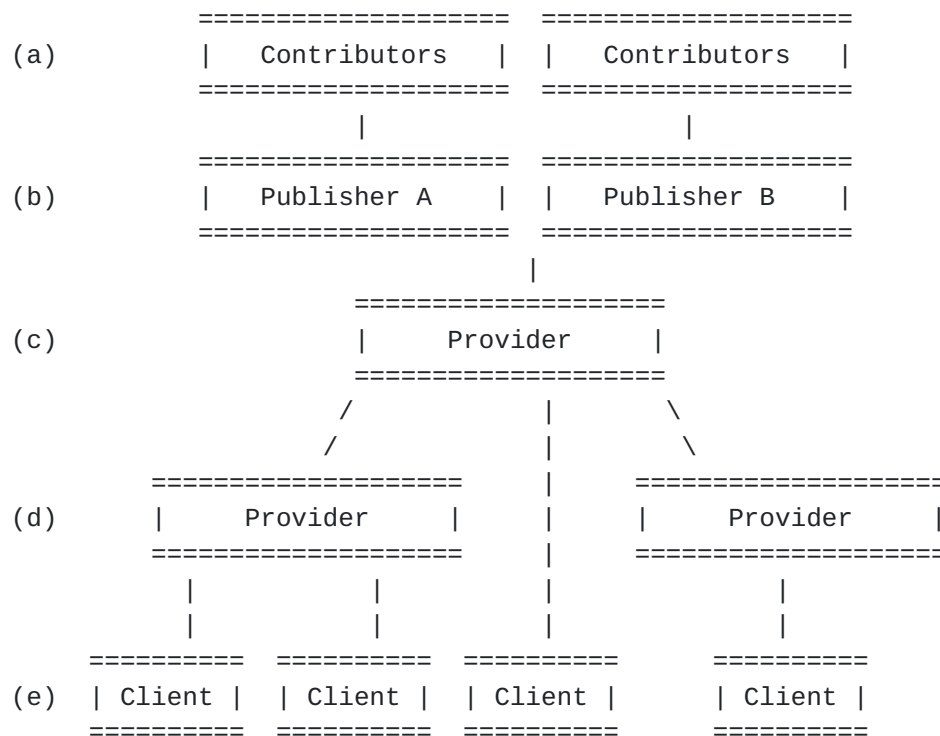


Figure 1: Time Zone Data Distribution Service Architecture

The overall service is made up of several layers:

- (a) Contributors: Individuals, governments or organizations which provide information about time zones to the publishing process. There can be many contributors.
- (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 "re-publish" data from another publisher.
- (c) Root Providers: Servers which 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 which 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 setup their own secondary provider to allow clients within their network to make requests of that server rather than

making 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 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

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 Data Server

A server implementing the Time Zone Data Distribution Service Protocol defined by this specification.

3.4. Observance

Data that defines a portion of a time zone where the UTC offset is a constant. A time zone with varying rules for the UTC offset will have many adjacent observances, with the total set covering the range of validity of the time zone data.

3.5. 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) time zone identifier naming scheme is out of scope for this specification.

3.6. 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 3.8](#)). 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.

3.7. Time Zone Localized Names

Localized names are names for time zones which 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.

3.8. 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 only new events 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 redundant.

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 and time values, and 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 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 either a "STANDARD" or "DAYLIGHT" sub-component 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 truncation range start point. This sub-component, which is the first observance defined by the time zone data, thus 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.

[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 data. The interactions with the HTTP server can be broken down into a set "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 distribution service has a "context path" request-URI 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 "{/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, which 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 template-URI 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.

Most security considerations are already handled adequately by HTTP. However, given the nature of the data being transferred and the requirement it be correct, all interactions between client and server SHOULD use an HTTP connection protected with TLS [RFC5246] as defined in [RFC2818].

[4.1.1.](#) Time Zone Queries

Time zone identifiers, aliases or localized names can be used to query for time zone data. 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 also 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 3.7](#), 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 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.

To support conditional time zone requests, based on whether the underlying time zone data has changed, the server supports two options: a timestamp based synchronization mechanism, and HTTP ETag based conditional requests (as per [\[RFC7232\]](#)).

For timestamp based synchronization, when a client requests a list of all time zones, a timestamp is returned by the server, and that serves as a synchronization "token" for later requests. Clients can use a timestamp token in a conditional "list" action (see [Section 5.2](#)) to limit the results to time zones which have changed since the previous request that returned the timestamp token. A new timestamp is then returned with the results, and that can be used in a subsequent request. This allows clients to periodically poll the server for possible changes, using the timestamp value from the previous poll.

If a client only needs data for one time zone (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. When the client retrieves the time zone data from the server using a "get" action (see [Section 5.3](#)) the server will include an ETag HTTP header field in the response. The client will store the value of that header field along with the request-URI used for the request. 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 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 which will include the changed data, as well as a new value for the ETag. Thus clients can poll the server for changes, and only retrieve new data when it is actually different from what it got before.

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

The following are examples of response codes one would expect to be used by the server. Note, however, that unless explicitly prohibited any 2/3/4/5xx series response code may be used in a response.

200 (OK) - The command succeeded.

304 (Not Modified) - The requested data is unchanged.

400 (Bad Request) - The Sender has provided an invalid request parameter.

404 (Not Found) - The time zone was not found.

When an HTTP error response is returned to the client, the server SHOULD return a JSON "problem detail" object in the response body, as per [\[I-D.ietf-appsawg-http-problem\]](#). Every JSON "problem detail" 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 9](#)). 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 sub-sections 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.](#) Time Zone Data Distribution Service SRV Service Labels

[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. Time Zone Data Distribution Service TXT records

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 next.

Example: text record for service with transport layer security.

```
_timezones._tcp TXT path=/timezones
```

4.2.1.3. Time Zone Data Distribution Service Well-Known URI

A "well-known" URI [\[RFC5785\]](#) is registered by this specification for the Time Zone Data Distribution service, "timezone" (see [Section 9](#)). 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. Servers MUST NOT locate the actual time zone data distribution service endpoint at the ".well-known" URI as per [Section 1.1 of \[RFC5785\]](#).

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 a "no-cache" value would be used.

To facilitate "context path's" that might differ from user to user, 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.

4.2.2. 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 first issuing a "list" action. The client would preserve the returned datestamp for subsequent use. Each time zone in the returned list can then be fetched and stored locally. In addition a mapping of aliases to time zones can be built.

4.2.3. Subsequent Synchronization of All Time Zones

A secondary service or a client caching all time zone data needs to periodically synchronize with a server. To do so it would issue a "list" action with the "changedsince" URI query parameter set to the value of the datestamp returned by the last synchronization. The client would again preserve the returned datestamp for subsequent use. Each time zone in the returned list can then be fetched and stored locally.

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.

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.

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 /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": "/capabilities",
    "parameters": []
  },
  {
    "name": "list",
    "uri-template": "/zones{?changedsince}",
    "parameters": [
      {
        "name": "changedsince",
        "required": false,
        "multi": false
      }
    ]
  }
],
{
  "name": "get",
  "uri-template": "/zones{/tzid}{?start,end}",
  "parameters": [
    {
      "name": "start",
      "required": false,
      "multi": false
    },
    {
      "name": "end",
      "required": false,
      "multi": false
    }
  ]
}
```



```
    ]
  },
  {
    "name": "expand",
    "uri-template":
      "/zones{/tzid}/observances{?start,end}",
    "parameters": [
      {
        "name": "start",
        "required": true,
        "multi": false
      },
      {
        "name": "end",
        "required": true,
        "multi": false
      }
    ]
  },
  {
    "name": "find",
    "uri-template": "/zones{?pattern}",
    "parameters": [
      {
        "name": "pattern",
        "required": true,
        "multi": false
      }
    ]
  }
]
```

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 aliases and optional localized data. In addition, it returns a timestamp which is the current server last modification value. If the "changedsince" URI query parameter is present, its value MUST correspond to a previously returned timestamp value.

When "changedsince" timestamp is used, the server MUST return only those time zones that have changed since the specified timestamp.

Parameters:

changedsince OPTIONAL, but MUST occur only once.

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

Possible Error Codes

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

[5.2.1](#). Example: List time zone identifiers

In this example the client requests the time zone identifiers and in addition requests that the US-English local names be returned.

>> Request <<

```
GET /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
```

```
{
  "dtstamp": "2009-10-11T09:32:11Z",
  "timezones": [
    {
      "tzid": "America/New_York",
      "last-modified": "2009-09-17T01:39:34Z",
      "aliases": ["US/Eastern"],
      "local-names": [
        {
          "name": "America/New_York",
          "lang": "en_US"
        }
      ]
    },
    ...
  ]
}
```

5.3. "get" Action

Name: get

Request-URI Template: {/service-prefix,data-prefix}/zones{/tzid}{?start,end}

The "tzid" variable value is REQUIRED 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 occur only 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 occur only 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 the time zone with a specific time zone identifier to be returned.

>> Request <<

```
GET /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 the time zone with a specific time zone identifier to be returned, but uses an If-None-Match header field in the request, set to the value of a previously returned ETag header field. In this example, the data on the server has not changed, so a 304 response is returned.

>> Request <<

```
GET /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 the time zone with an aliased time zone identifier to be returned, and the server returns the time zone data with that identifier, and two aliases.

>> Request <<

```
GET /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 the time zone with a specific time zone identifier truncated at one of the ranges specified by the server, to 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 of the time zone data.

>> Request <<

```
GET /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:20191231T111159Z
BEGIN:STANDARD
DTSTART:20101231T190000
TZNAME:EST
TZOFFSETFROM:-0500
TZOFFSETTO:-0500
END:STANDARD
...
END:VTIMEZONE
END:VCALENDAR
```

[5.3.5](#). Example: Get a non-existent time zone data

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

>> Request <<

```
GET /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 (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. e.g., 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 point 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 /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 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 sub-string text match is done,
e.g., "xyz"

In addition, when matching, underscore characters (0x5F) SHOULD be mapped to a single space character (0x20) prior to string comparison. This allows time zone identifiers such as "America/New_York" to match a query for "*New York*". ASCII characters in the range 0x41 ("A") through 0x5A ("Z") SHOULD be mapped to their lowercase equivalents. To match characters 0x2A ("*") and 0x5C ("\") in the pattern, a single 0x5C ("\") is prepended to act as an "escaping" mechanism. i.e., a pattern "Test\" implies an exact match test against the string "Test*".

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 /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
```

```
{
  "dtstamp": "2009-10-11T09:32:11Z",
  "timezones": [
    {
      "tzid": "America/New_York",
      "last-modified": "2009-09-17T01:39:34Z",
      "aliases": ["US/Eastern"],
      "local-names": [
        {
          "name": "America/New_York",
          "lang": "en_US"
        }
      ]
    },
    {
      "tzid": "America/Detroit",
      "last-modified": "2009-09-17T01:39:34Z",
      "aliases": ["US/Eastern"],
      "local-names": [
        {
          "name": "America/Detroit",
          "lang": "en_US"
        }
      ]
    },
    ...
  ]
}
```


6. JSON Definitions

JSON members used by this specification are defined here using the syntax in [[I-D.newton-json-content-rules](#)]. Clients MUST ignore any JSON members they do not expect.

6.1. capabilities action response

JSON Content Rules for the JSON document returned for a "capabilities" action request.

; root object

```
root {  
  version,  
  info,  
  actions  
}
```

; The version number of the protocol supported - MUST be 1
version "version" : integer 1..1

; object containing service information

```
info "info" {  
  primary_source / secondary_source,  
  formats,  
  ?truncated,  
  ?provider_details,  
  ?contacts  
}
```

; The source of the time zone data provided by a "primary" server
primary_source "primary-source" : string

; The time zone data server from which data is provided by a
; "secondary" server
secondary_source "secondary-source" : uri

; Array of media types for the time zone data formats that the
; server can return
formats "formats" [1* : string]

; Present if the server is providing truncated time zone data. The
; value is an object providing details of the supported truncation
; modes.

```
truncated "truncated" : {  
  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
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"
ranges "ranges" : [ * : range ]

; A range of time
range {
  range-start,
  range-end
}

; [RFC3339] UTC date-time value 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
range-start "start" : date-time

; [RFC3339] UTC date-time value 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
range-end "end" : date-time

; 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
untruncated "untruncated" : boolean

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

; Array of URIs providing contact details for the server
; administrator
contacts "contacts" [ * : uri ]

; Array of actions supported by the server
actions "actions" [ * action ]
```



```
; An action supported by the server
action {
    action_name,
    action_uri_template,
    action_params
}

; Name of the action
action_name "name" : string

; URI template for action - this is the URI template from the action
; definitions in Section 5 without the {/service-prefix,data-prefix}
; elements
action_uri_template "uri-template" : string

; Array of request-URI query parameters supported by the action
action_params "parameters" [ * parameter ]

; Object defining an action parameter
parameter {
    param_name,
    ?param_required,
    ?param_multi,
    ?param_values
}

; Name of the parameter
param_name "name" : string

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

; If true the parameter can occur more than once in the request-URI
; default is false
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
param_values "values" [ * : string ]
```

6.2. list/find action response

JSON Content Rules for the JSON document returned for a "list" or "find" action request.


```
; root object

root {
    dtstamp,
    timezones
}

; Server generated timestamp used for synchronizing changes,
; [RFC3339] UTC value
dtstamp "dtstamp" : date-time

; Array of time zone objects
timezones "timezones" [ * timezone ]

; Information about a time zone available on the server
timezone {
    tzid,
    last_modified,
    ?aliases,
    ?local_names,
}

; Time zone identifier
tzid "tzid" : string

; Date/time when the time zone data was last modified
; [RFC3339] UTC value
last_modified "last-modified" : date-time

; An array that lists the set of time zone aliases available
; for the corresponding time zone
aliases "aliases" [ * : string ]

; An array that lists the set of localized names available
; for the corresponding time zone
local_names "local-names" [ * local_name ]

local_name [lang, lname, ?pref]

; Language tag for the language of the associated name
lang : string

; Localized name
lname : string

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


6.3. expand action response

JSON Content Rules for the JSON document returned for a "expand" action request.

```
; root object

root {
    tzid,
    start?,
    end?,
    observances
}

; Time zone identifier
tzid "tzid" : string

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

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

; Array of time zone objects
observances "observances" [ * observance ]

; Information about a time zone available on the server
observance {
    oname,
    ?olocal_names,
    onset,
    utc_offset_from,
    utc_offset_to
}

; Observance name
oname "name" : string

; Array of localized observance names
olocal_names "local-names" [ * : string]

; \[RFC3339\] UTC date-time value at which the observance takes effect
onset "onset" : date-time

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

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


7. New iCalendar Properties

7.1. Time Zone Upper Bound

Property Name: TZUNTIL

Purpose: This property specifies an upper bound for the validity 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 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" sub-components. However, [\[RFC5545\]](#) does not define a way to indicate an upper bound on the validity of the time zone data, which cannot be simply derived from the observance with the latest onset time. This specification introduces the "TZUNTIL" property for that purpose. It specifies an "inclusive" 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 3.8](#)).

Format Definition: This property is defined by the following notation:

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:20251231T235959Z

7.2. Time Zone Identifier Alias Property

Property Name: TZID-ALIAS-OF

Purpose: This property specifies a time zone identifier that the main time zone identifier is an alias of.

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 3.6](#)).

Format Definition: This property is defined by the following notation:

```
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 [\[RFC2818\]](#)) with a valid certificate. Clients and servers making use of a time zone data distribution service SHOULD use HTTP over TLS and verify the authenticity of the service being used before accepting and using any time zone data from that source.

Clients that support transport layer security as defined by [\[RFC2818\]](#) SHOULD try the "_timezones" service first before trying the

"_timezone" service. Clients MUST follow the certificate verification process specified in [\[RFC6125\]](#).

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 matches the original service domain that was queried. If the target FQDN is not in the queried domain, clients SHOULD verify with the user that the SRV target FQDN is suitable for use before executing any connections to the host.

Time zone data servers SHOULD protect themselves against errant 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) to restrict which clients are allowed to access their service, or provide better identification of errant clients. As such, servers MAY require HTTP-based authentication as per [\[RFC7235\]](#).

9. 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" sub-namespace to the IETF parameters URN sub-namespace as per [\[RFC3553\]](#) for use with protocol related error codes.

9.1. Service Actions Registration

This section defines the process to register new or modified time zone data distribution service actions with IANA.

9.1.1. Service Actions Registration Procedure

The IETF will create a mailing list, tzdist-service@ietf.org, which can be used for public discussion of time zone data distribution service actions proposals prior to registration. Use of the mailing list is strongly encouraged. The IESG will appoint a designated expert who will monitor the tzdist-service@ietf.org mailing list and review registrations.

Registration of new time zone data distribution service actions **MUST** be reviewed by the designated expert and published in an RFC. A Standard Track RFC is **REQUIRED** for the registration of new time zone data distribution service actions. A Standard Track RFC is also **REQUIRED** for changes to actions previously documented in a Standard Track RFC.

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 within two weeks whether the registration is approved, approved with minor changes, or rejected with cause. When a registration is rejected with cause, it can be re-submitted if the concerns listed in the cause are addressed. Decisions made by the designated expert can be appealed to the IESG Applications Area Director, then to the IESG. They follow the normal appeals procedure for IESG decisions.

9.1.2. 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.

9.2. Initial Time Zone Data Distribution Service Registry

The IANA is requested to create and maintain the following registry for time zone data distribution service actions with pointers to appropriate reference documents.

9.2.1. Actions Registry

The following table is to be used to initialize the actions registry.

+-----+-----+-----+			
Action Name	Status	Reference	
+-----+-----+-----+			
capabilities	Current	RFCXXXX, Section 5.1	
list	Current	RFCXXXX, Section 5.2	
get	Current	RFCXXXX, Section 5.3	
expand	Current	RFCXXXX, Section 5.4	
find	Current	RFCXXXX, Section 5.5	
+-----+-----+-----+			

9.3. timezone Well-Known URI Registration

URI suffix: timezone

Change controller: IETF.

Specification document(s): This RFC.

Related information:

9.4. Service Name Registrations

This document registers two new service names as per [[RFC6335](#)]. Both are defined within this document.

9.4.1. 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: [This Draft]

Assignment Note: This is an extension of the http service. Defined
 TXT keys: path=<context path>

9.4.2. 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: [This Draft]

Assignment Note: This is an extension of the https service. Defined
TXT keys: path=<context path>

9.5. tzdist URN sub-namespace

IANA is requested to register a new URN sub-namespace within the IETF URN Sub-namespace for Registered Protocol Parameter Identifiers defined in [[RFC3553](#)].

Registry name: tzdist

Specification): This RFC

Repository: Time Distribution Service protocol elements registry.

Index value:: This specification defines the single "error" sub-parameter, which itself has sub-parameters representing 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](#)).

9.6. iCalendar Property Registrations

This document defines the following new iCalendar properties to be added to the registry defined in [Section 8.2.3 of \[RFC5545\]](#):

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

10. 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, John Haug, Ciny Joy, Bryan Keller, 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.

11. References

11.1. Normative References

- [I-D.ietf-appsawg-http-problem]
Nottingham, M. and E. Wilde, "Problem Details for HTTP APIs", [draft-ietf-appsawg-http-problem-00](#) (work in progress), September 2014.
- [I-D.newton-json-content-rules]
Newton, A., "A Language for Rules Describing JSON Content", [draft-newton-json-content-rules-03](#) (work in progress), October 2014.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", [RFC 2046](#), November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", [RFC 2782](#), February 2000.
- [RFC2818] Rescorla, E., "HTTP Over TLS", [RFC 2818](#), May 2000.
- [RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", [RFC 3339](#), July 2002.
- [RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", [BCP 73](#), [RFC 3553](#), June 2003.

- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, [RFC 3629](#), November 2003.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), August 2008.
- [RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", [RFC 5545](#), September 2009.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", [RFC 5785](#), April 2010.
- [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](#), March 2011.
- [RFC6321] Daboo, C., Douglass, M., and S. Lees, "xCal: The XML Format for iCalendar", [RFC 6321](#), August 2011.
- [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](#), August 2011.
- [RFC6557] Lear, E. and P. Eggert, "Procedures for Maintaining the Time Zone Database", [BCP 175](#), [RFC 6557](#), February 2012.
- [RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", [RFC 6570](#), March 2012.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", [RFC 6763](#), February 2013.
- [RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", [RFC 7159](#), March 2014.
- [RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", [RFC 7230](#), June 2014.
- [RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", [RFC 7231](#), June 2014.

- [RFC7232] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", [RFC 7232](#), June 2014.
- [RFC7234] Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", [RFC 7234](#), June 2014.
- [RFC7235] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Authentication", [RFC 7235](#), June 2014.
- [RFC7265] Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON Format for iCalendar", [RFC 7265](#), May 2014.

11.2. Informative References

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

Appendix A. Change History (to be removed prior to publication as an RFC)

Changes for -04

1. Tweaked invalid-start/end for action expand to indicate outside truncation range.
2. Added text on use of Accept-Language.
3. Added text on requirement to percent-encode {/tzid}.
4. Moved /observances under /zones{/tzid}.
5. Observances response now includes start/end of actual range returned if different from what was requested.
6. Truncation end and &end= for get action are now exclusive.
7. Added capabilities action in capabilities example response.
8. Added uri-template items to capabilities action definitions.
9. Added start/end items to the observances response.
10. Error codes are now URNs (with an IANA registration for a tzdist sub-namespace) and the URNs are used as the type value in JSON problem reports.

11. Removed "changesince" from expand action - ETag can be used instead.

Changes for -03

1. Reworked conditional list section
(<https://tools.ietf.org/wg/tzdist/trac/ticket/22> &
<https://tools.ietf.org/wg/tzdist/trac/ticket/33>).
2. Moved definitions into General Considerations section.
3. Now makes use of ietf-appsawg-http-problem for error responses.
4. Switched to using a more RESTful design with resources used to identify endpoints for actions
(<https://tools.ietf.org/wg/tzdist/trac/ticket/29>).
5. Tweaked TZUNTIL text to further address
(<https://tools.ietf.org/wg/tzdist/trac/ticket/15>).
6. Tweaked "outright" deletion text to match latest on mailing list
(<https://tools.ietf.org/wg/tzdist/trac/ticket/18>).
7. Added additional text suggesting other discovery mechanisms could be used (<https://tools.ietf.org/wg/tzdist/trac/ticket/30>).
8. Now require "end" parameter on expand to avoid issues with truncated data upper bounds
(<https://tools.ietf.org/wg/tzdist/trac/ticket/10>).

Changes for -02

1. "time zone server" -> "time zone data server".
2. Re-worded some text containing reference to "historical" time zone data, and truncation behavior.
3. Removed "REST".
4. Use "TZID-ALIAS-OF" in place of "EQUIVALENT-TZID".
5. Added \-escape mechanism for find action.
6. Revised [Section 4.2.3](#) to address
(<https://tools.ietf.org/wg/tzdist/trac/ticket/18>).

Changes for -01

1. Query attribute: "name" -> "pattern"
(<https://tools.ietf.org/wg/tzdist/trac/ticket/4>).
2. UTF-8 used for time zone ids and in all responses.
3. Added glossary term and note for "time zone"
(<https://tools.ietf.org/wg/tzdist/trac/ticket/12>).
4. Glossary term change and alias text from
(<https://tools.ietf.org/wg/tzdist/trac/ticket/13>).
5. "Local Provider" -> "Secondary Provider".
6. Additional security text for
(<https://tools.ietf.org/wg/tzdist/trac/ticket/25>).
7. Added additional text to better describe localized names.
8. Added "tzid" member to expand response.
9. Added optional "provider-details" member to capabilities response, and also made "contacts" optional.
10. Definition of "invalid-action" moved to [Section 6](#), and clarified text related to error responses in Sections [4.1.6](#) and [6](#)
(<https://tools.ietf.org/wg/tzdist/trac/ticket/17>).
11. Added "Observance" to glossary.
12. Added "TZUNTIL" iCalendar property (part of
<https://tools.ietf.org/wg/tzdist/trac/ticket/15>).
13. Revamped truncation to always use UTC date-time values and support end points (<https://tools.ietf.org/wg/tzdist/trac/ticket/21>, and <https://tools.ietf.org/wg/tzdist/trac/ticket/10>).
14. Expand always uses UTC date-time values for query parameters, and always returns UTC date-time onset values
(<https://tools.ietf.org/wg/tzdist/trac/ticket/21>).

Changes for -00

1. Initial WG draft derived from [draft-douglass-timezone-service-11](#), with some terminology changes to match WG name.
2. Updated references.

3. "timezone" -> "time zone" (<https://tools.ietf.org/wg/tzdist/trac/ticket/6>).
4. Glossary tweak (first part of <https://tools.ietf.org/wg/tzdist/trac/ticket/13>).
5. Fix iCalendar property names: UTC-OFFSET-* -> TZOFFSET*.
6. Fix invalid-truncate error code description.

Authors' Addresses

Michael Douglass
Rensselaer Polytechnic Institute
110 8th Street
Troy , NY 12180
USA

Email: dougln@rpi.edu
URI: <http://www.rpi.edu/>

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

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

