Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 27, 2015

M. Douglass
RPI
C. Daboo
Apple
August 26, 2014

# Time Zone Data Distribution Service
## draft-ietf-tzdist-service-00

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 February 27, 2015.

Table of Contents

## 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 (sometimes 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 REST style API, with JSON [RFC7159] responses.

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].

### 1.2. Glossary of terms

The following terms with the given meanings are used throughout this document.

Time Zone Data: Data that defines a single time zone, including an identifier, UTC offset values, and DST rules;

Time Zone Server: A server implementing the Time Zone Data Distribution Service Protocol defined by this specification;

Time Zone Identifier: A globally unique name which identifies time zone data, and which may include other information such as time zone abbreviations.

## 2. Architectural Overview

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

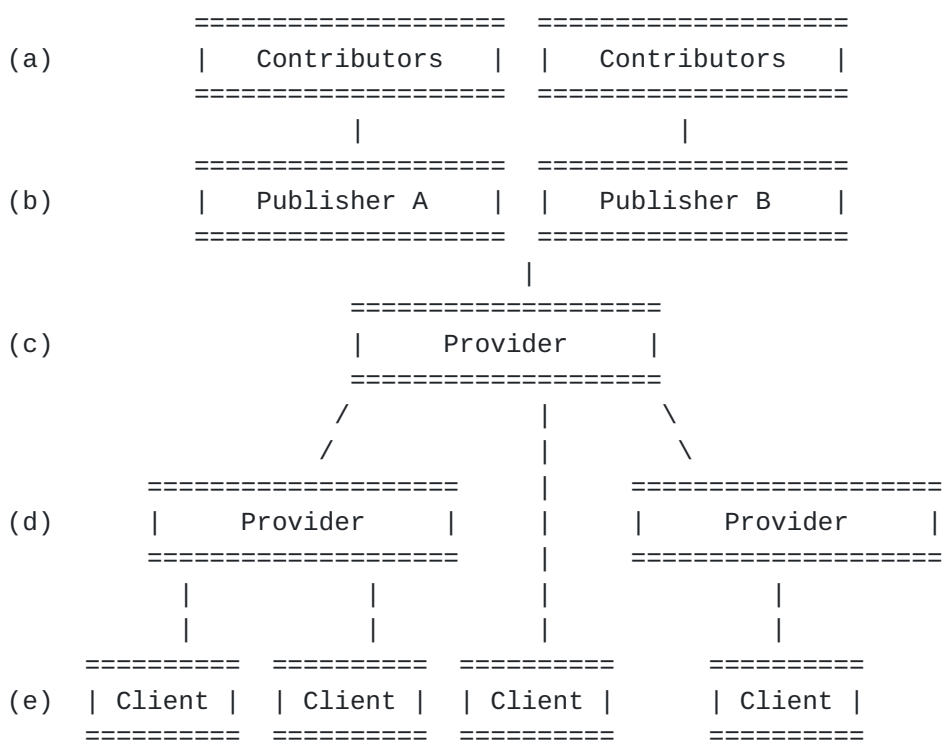


Figure 1: Time Zone Data Distribution Service Architecture

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) Local 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 local provider to allow clients within their network to make requests of that server rather than making

requests of servers outside their network. Local 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 local 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 local servers and a small number of root servers.

This specification is only concerned 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

### 3.1. 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.

### 3.2. 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 equivalent 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 equivalent identifiers for "America/New_York" or "America/Montreal".

### 3.3. 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.

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.

### 3.4. Truncated Time Zones

Time zones and daylight saving times rules have been in use for over a century. Time zone data can thus contain a large amount of "historical" information 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 only create new events for the present and future. To avoid having to send unnecessary data, servers are allowed to truncate time zone data at some appropriate date in the past, and only provide accurate offsets and rules from that point on. The server will need to advertise the cut-off dates it is using so that clients that need time zone data for earlier dates can take appropriate action. To simplify the set of data a server needs to support, truncation always occurs at the start of a year, i.e., midnight on 1st January for the time zone's local time. A server will advertise a set of years for truncated data it can supply, or provide an indicator that it can truncate at any past year. In addition, the server will advertise that it can supply untruncated data. In the absence of any indication of truncated data available on the server, the server will only supply untruncated data.

When truncating a "VTIMEZONE" component, the server MUST include either a "STANDARD" or "DAYLIGHT" sub-component with a "DTSTART" property value that matches the date-time where the truncation occurred, and appropriate "TZOFFSETFROM" and "TZOFFSETTO" properties to indicate the correct offset in effect right after the point of truncation. This sub-component thus represents the earliest valid

date-time covered by the time zone data in the truncated "VTIMEZONE" component.

# 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. Queries are made on a single HTTP resource using the GET method, with specific client request attributes passed in request-URI parameters.

The "action" request-URI parameter defines the overall function being requested, with other request parameters acting as arguments to that function.

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" request parameter is used then the value may be an identifier or an alias. When the "name" parameter is used it may be an identifier, an alias or a localized name.

### 4.1.2. Time Zone Formats

The default format for returning time zone definitions is the iCalendar [RFC5545] data format. In addition, the iCalendar-in-XML [RFC6321], and iCalendar-in-JSON [RFC7265] representations are also available. The "format" request-URI parameter can be used to select which data format is returned.

### 4.1.3. Conditional Time Zone Requests

Time zone data is generally slow moving, with the set of time zones that change from even year-to-year being relatively small. However, any changes that do occur, need to be distributed in a timely manner. Typically it is more efficient to just provide the set of changes to time zone data, so a client can do updates to any locally cached data.

When listing time zones, a timestamp is returned by the server, and that can be used later by clients to determine if any "substantive" change has occurred in the time zone data. Clients can use a conditional "list" action (see Section 6.2), supplying a previous timestamp value, to limit the results to time zones which have changed in a "substantive" manner since that previous timestamp. This allows clients to cache the last timestamp and to periodically poll the server for possible changes.

A "substantive" change is one which affects the calculated onsets for a time zone. Changes to properties such as a description are not treated as a "substantive" change.

Clients SHOULD poll for such changes at least once a day. A server acting as a local provider, caching time zone data from another server, SHOULD poll for changes once per hour. See Section 9 on expected client and server behavior regarding high request rates.

#### 4.1.4. 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 definition into a set of "observances" over a fixed period of time (see Section 6.4). Each of these observances describes a local 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.

#### 4.1.5. 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.

#### 4.1.6. 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.

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

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

When an error status is set the server SHOULD respond with some descriptive text in an error object as per Section 7.4. In the case of an invalid "action" query parameter, the following error code can be used:

invalid-action The "action" query parameter has an incorrect value.

#### 4.1.7. 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.

##### 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 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. 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 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 only 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 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" 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.

Note, this process makes no provision for handling deleted time zones. In general it is bad practice to delete time zones as they might still be in use by consumers of time zone data.

## 5. Request Parameters

The "action" request-URI parameter MUST be included in all requests to define what action is required of the server.

The following request-URI parameters are used with the various actions.

### 5.1. "action" Parameter

Name: action

Description: Specify the action to be carried out.

Value: Any IANA registered action name (see Section 10.2.1).

### 5.2. "format" Parameter

Name: format

Description: Specify the format of the time zone data returned by the server as a standard MIME [RFC2046] media-type. If absent, the iCalendar [RFC5545] format will be returned with the time zones contained within a "VCALENDAR" object (i.e., a default media-type of "text/calendar").

Value: A MIME [RFC2046] media-type. The following values MAY be used, with servers advertising the values they do support via the "capabilities" action response (see Section 6.1):

text/calendar: Return data as "VTIMEZONE" components embedded in a "VCALENDAR" object as per [RFC5545].

application/calendar+xml: Return data using the XML representation of iCalendar data as per iCalendar-in-XML [RFC6321].

application/calendar+json: Return data using the JSON representation of iCalendar data as per iCalendar-in-JSON.

### 5.3. "changedsince" Parameter

Name: changedsince

Description: Specify the timestamp for a conditional "list" (see Section 6.2) or "expand" (see Section 6.4) action in order to restrict the results to only changes since the given timestamp.

Value: An [RFC3339] UTC date-time value, typically a value returned by a previous request.

### 5.4. "start" Parameter

Name: start

Description: Specify the inclusive start of a period.

Value: An integer representing a year..

### 5.5. "end" Parameter

Name: end

Description: Specify the exclusive end of a period.

Value: An integer representing a year.

### 5.6. "lang" Parameter

Name: lang

Description: Specify the language in which locale specific values are to be returned. e.g., if a language is specified, only localized names for that language would be returned.

Value: The value follows the specifications in [RFC5646].

### 5.7. "tzid" Parameter

Name: tzid

Description: Specify a time zone to be targeted by an action.

Value: A time zone identifier or alias.

### 5.8. "name" Parameter

Name: name

Description: Specify a name for queries.

Value: A time zone identifier, alias or localized name. This parameter is used when searching for matching time zones (see Section 6.5).

### 5.9. "truncate" Parameter

Name: truncate

Description: Specify a year for time zone data truncation.

Value: An integer representing a year in the past. The use of this depends on the "truncated" object returned in the server's "capabilities" response:

If "truncated" object is not present in the "capabilities" response, then the "truncated" parameter MUST NOT be used - the server will always return untruncated time zone data.

If "any" is set to "true" in the "truncated" object, then any past year is valid for truncation (though typically data prior to 1880 is unlikely to be present).

If "any" is "false" and "years" is present with at least one value, then any of the values in the "years" array can be used.

If "untruncated" is set to "true", then omitting the "truncated" parameter will result in untruncated data being returned.

If "untruncated" is set to "false", and "years" contains only one value, and the "truncated" query parameter is omitted, then the server will return time zone data truncated at the one value specified in "years".

Example: a server that can only return one set of truncated data - client can omit the "truncate" query parameter:

```
truncated: {
  "any": false,
  "years": [1970],
  "untruncated": false
}
```

Example: a server that can return truncated data for any past year as well as untruncated data if client omits the "truncate" query parameter:

```
truncated: {
  "any": true,
  "untruncated": true
}
```

Example: a server that can return only untruncated data - the "truncate" query parameter would always be omitted:

```
truncated: {
  "any": false,
  "untruncated": true
}
```

## 6. Actions

Servers MUST support the following actions.

### 6.1. "capabilities" Action

Name: capabilities

Description: This action returns the capabilities of the server, allowing clients to determine if a specific feature has been deployed and/or enabled. Note that each request always includes an "action" query parameter set to the name of the action, even though that parameter is not listed in the "capabilities" response for each action.

Parameters:

action REQUIRED with value "capabilities"

Response A JSON object containing a "version" member, an "info" member, and an "actions" member, see Section 7.1.

Possible Error Codes No specific code.

#### 6.1.1. Example: Get Capabilities

```
>> Request <<

GET /?action=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",
    "truncate" : {
      "any": false,
      "years": [1970, 2000, 2010],
      "untruncated": true
    },
```

```
      "contacts": ["mailto:tzs@example.org"]
    },

    "actions": [
      {
        "name": "list",
        "parameters": [
          {
            "name": "lang",
            "required": false,
            "multi": true
          },
          {
            "name": "changedsince",
            "required": false,
            "multi": false
          }
        ]
      },

      {
        "name": "get",
        "parameters": [
          {
            "name": "format",
            "required": false,
            "multi": false,
            "values": [
              "text/calendar",
              "application/calendar+xml",
              "application/calendar+json"
            ]
          },
          {
            "name": "lang",
            "required": false,
            "multi": true
          },
          {
            "name": "tzid",
            "required": true,
            "multi": false
          },
          {
            "name": "truncate",
            "required": false,
            "multi": false
          }
```

```
        ]
      },

      {
        "name": "expand",
        "parameters": [
          {
            "name": "tzid",
            "required": true,
            "multi": false
          },
          {
            "name": "start",
            "required": false,
            "multi": false
          },
          {
            "name": "end",
            "required": false,
            "multi": false
          }
        ]
      },

      {
        "name": "find",
        "parameters": [
          {
            "name": "name",
            "required": true,
            "multi": false
          },
          {
            "name": "lang",
            "required": false,
            "multi": true
          }
        ]
      },

      {
        "name":"capabilities",
        "parameters": []
      }
    ]
  }
```

### 6.2. "list" Action

Name: list

Description: This action lists all time zone identifiers or the requested 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" 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. If the "tzid" parameter is present one or more times, then the server MUST only return information for the specified time zone identifiers.

Parameters:

action REQUIRED with value "list"

lang=<lang-code> OPTIONAL, but MAY occur multiple times.

changedsince OPTIONAL, but MUST occur only once. MUST NOT be present if the "tzid" parameter is present.

tzid=<identifier> OPTIONAL, and MAY occur multiple times. MUST NOT be present if the "changedsince" parameter is present. The value of the "dtstamp" member in the response applies to the entire set of data, rather than the subset requested with the "tzid" query parameter, and allows the client to determine if it needs to refresh its full set of time zone data.

Response: A JSON object containing a "dtstamp" member and a "timezones" member, see Section 7.2.

Possible Error Codes

invalid-changedsince The "changedsince" query parameter has an incorrect value, or appears more than once.

invalid-tzid The "tzid" query parameter is present along with the "changedsince", or has an incorrect value.

#### 6.2.1. 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 /?action=list&lang=en_US 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"
        }
      ]
    },
    ...
  ]
}
```

### 6.3. "get" Action

Name: get

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

If the identifier 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 "EQUIVALENT-TZID" properties (see Section 8) in the time zone data to indicate equivalent identifiers for the alias.

Parameters:

action REQUIRED with value "get"

format=<media-type> OPTIONAL, but MUST occur only once.

lang=<lang-code> OPTIONAL, but MAY occur multiple times.

tzid=<identifier> REQUIRED, and MUST occur only once.

truncate=<year> OPTIONAL, and MUST occur only once. See Section 5.9 for details.

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

Possible Error Codes

invalid-tzid The "tzid" query parameter is not present, or appears more than once.

tzid-not-found No time zone associated with the specified "tzid" query parameter value was found.

invalid-format The "format" query parameter appears more than once, or has an invalid value.

invalid-truncate The "truncate" query parameter appears more than once, or has an invalid year specified as its value.

### 6.3.1. Example: Get time zone

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

```
>> Request <<

GET /?action=get&tzid=America/New_York
                  &format=text/calendar HTTP/1.1
Host: tz.example.com

>> 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
```

### 6.3.2. Example: Get 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 equivalents

>> Request <<

```
GET /?action=get&tzid=US/Eastern
                  &format=text/calendar
                  &truncate=2000 HTTP/1.1
Host: tz.example.com
```

>> 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
EQUIVALENT-TZID:America/New_York
EQUIVALENT-TZID:America/Montreal
...
END:VTIMEZONE
END:VCALENDAR
```

### 6.3.3. Example: Get truncated time zone

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

```
truncated: {
  "any": false,
  "years": [1970, 2000],
  "untruncated": false
}
```

In this example the client requests the time zone with a specific time zone identifier truncated at one of the years specified as available by the server, to be returned

>> Request <<

```
GET /?action=get&tzid=America/New_York
                  &format=text/calendar
                  &truncate=2000 HTTP/1.1
Host: tz.example.com
```

>> 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
```

### 6.4. "expand" Action

Name: expand

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

Parameters:

action REQUIRED with value "expand"

tzid=<identifier> REQUIRED, but MUST only occur once.

lang=<lang-code> OPTIONAL, but MAY occur multiple times.

start=year: OPTIONAL, but MUST occur only once. If present, specifies the start of the period of interest. The value is an integer representing the starting year, with the start date

assumed to be January 1st of that year. If "start" is omitted, the value is assumed to be the current year.

end=year: OPTIONAL, but MUST occur only once. If present, specifies the ending year of the period of interest. The value is an integer representing the ending year, with the end date assumed to be January 1st of that year. If "end" is omitted, the value is the start year + 10. 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" and an end value of "2015". An error occurs if the end year is less than or equal to the start year.

changedsince OPTIONAL, but MUST occur only once. If present, its value MUST be taken from the "dtstamp" result of a previous expand result. If the targeted time zone has not changed over the expansion range queried in the request, then the server MUST return a 304 HTTP status response.

Response: A JSON object containing a "dtstamp" member and an "observances" member, see Section 7.3. The server MUST include an expanded observance representing the time zone information in effect at the start of the period (midnight local time, January 1st of the start year).

Possible Error Codes

invalid-tzid The "tzid" query parameter is not present, or appears more than once.

tzid-not-found No time zone associated with the specified "tzid" query parameter value was found.

invalid-start The "start" query parameter has an incorrect value, or appears more than once.

invalid-end The "end" query parameter has an incorrect value, or appears more than once, or is missing, or has a value less than or equal to the "start" query parameter.

invalid-changedsince The "changedsince" query parameter has an incorrect value, or appears more than once.

### 6.4.1. Example: Expanded JSON Data Format

In this example the client requests a time zone in the expanded form.

```
>> Request <<

GET /?action=expand&tzid=America/New_York&start=2008&end=2009 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"

{
  "dtstamp": "2009-10-11T09:32:11Z",
  "observances": [
    {
      "name": "Standard",
      "onset": "2008-01-01T00:00:00",
      "utc-offset-from": -18000,
      "utc-offset-to": -18000
    },
    {
      "name": "Daylight",
      "onset": "2008-03-09T02:00:00",
      "utc-offset-from": -18000,
      "utc-offset-to": -14400
    },
    {
      "name": "Standard",
      "onset": "2008-11-02T02:00:00",
      "utc-offset-from": -14400,
      "utc-offset-to": -18000
    },
  ]
}
```

## 6.5. "find" Action

Name: find

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 match against the names known to

the server (with an asterisk * as the wildcard character). 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 An 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.

Parameters:

action REQUIRED with value "find"

name=<text> REQUIRED, but MUST only occur once.

lang=<lang-code> OPTIONAL, but MAY occur multiple times.

Response: The response has the same format as the "list" action, with one result object per successful match, see Section 7.2.

Possible Error Codes

invalid-name The "name" query parameter is not present, or appears more than once.

### 6.5.1. Example: Find action

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

>> Request <<

```
GET /?action=find&name=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"
        }
      ]
    },
    ...
  ]
}
```

## 7. 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.

### 7.1. 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,
  ?truncated,
  contacts
}

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

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

; Present if the server is providing truncated time zone data. The
; value is the truncation date-time. Time zone data will not be
; valid for dates prior to this value.
; [RFC3339] UTC value
truncated "truncated" : {
  any,
  ?years,
  ?untruncated
}

; Indicates whether the server can truncate time zone data at any year
; boundary in the past. When set to "true" any past year is a valid
```

```
  ; value for use with the "truncated" query parameter in an action
  ; "get" request
  any "any" : boolean

  ; Indicates which year boundaries the server has truncated data for.
  ; A value from this list may be used with the "truncated" query
  ; parameter in an action "get" request. Not present if "any" is set
  ; to "true"
  years "years" : [ * : integer ]

  ; Indicates whether the server can can supply untruncated data. When
  ; set to "true" indicates that, in addition to truncated data being
  ; available, the server can return untruncated data if an action "get"
  ; request is executed without a "truncated" query parameter
  untruncated "untruncated" : boolean

  ; 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_params
  }

  ; Name of the action
  action_name "name" : 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 ]
```

### 7.2. list action response

JSON Content Rules for the JSON document returned for a "list" 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
```

### 7.3. expand action response

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

```
; root object

root {
  dtstamp,
  observances
}

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

; The local time at which the observance takes effect
; [RFC3339] value modified to exclude "time-offset" part
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
```

### 7.4. error response

JSON Content Rules for the JSON document returned when an error occurs.

```
; root object

root {
  error,
  ?description
}

; Error code
error "error" : string

; Description of the error
description "description" : string
```

## 8. Equivalent Time Zone Identifier Property

Property Name: EQUIVALENT-TZID

Purpose: This property specifies an equivalent time zone identifier representing the same time zone data as the aliased "VTIMEZONE" component.

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: This property specifies an equivalent time zone identifier for a "VTIMEZONE" component when the "TZID" property of the time zone is an alias identifier.

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

```
equivalent-tzid    = "EQUIVALENT-TZID" etzidpropparam ":"
                         [tzidprefix] text CRLF

etzidpropparam     = *(";" other-param)

;tzidprefix defined in [RFC5545].
```

Example: The following is an example of this property:

EQUIVALENT-TZID:US/Eastern

## 9. 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 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 4.1.3

## 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].

### 10.1. Service Actions Registration

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

#### 10.1.1. 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 Tracks RFC is REQUIRED for the registration of new time zone data distribution service actions. A Standard Tracks RFC is also REQUIRED for changes to actions previously documented in a Standard Tracks RFC.

The registration procedure begins when a completed registration template, defined in the sections 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.

#### 10.1.2. Registration Template for Actions

An action is defined by completing the following template.

Name: The name of the action. This is also the value of the "action" parameter used in time zone data distribution service requests.

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

Parameters: A list of allowed request parameters, indicating whether they are "REQUIRED" or "OPTIONAL" and whether they can occur only once or multiple times.

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

### 10.1.3. Registration Template for Action Parameters

An action parameter is defined by completing the following template.

Name: The name of the parameter.

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

Value: The format of the parameter value, or an indication that the parameter has no value.

## 10.2. Initial Time Zone Data Distribution Service Registries

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

### 10.2.1. Actions Registry

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

| Action Name | Status | Reference |
|---|---|---|
| capabilities | Current | RFCXXXX, Section 6.1 |
| list | Current | RFCXXXX, Section 6.2 |
| get | Current | RFCXXXX, Section 6.3 |
| expand | Current | RFCXXXX, Section 6.4 |
| find | Current | RFCXXXX, Section 6.5 |

### 10.2.2. Action Parameters Registry

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

| Parameter | Status | Reference |
|---|---|---|
| action | Current | RFCXXXX, Section 5.1 |
| changedsince | Current | RFCXXXX, Section 5.3 |
| end | Current | RFCXXXX, Section 5.5 |
| format | Current | RFCXXXX, Section 5.2 |
| lang | Current | RFCXXXX, Section 5.6 |
| name | Current | RFCXXXX, Section 5.8 |
| start | Current | RFCXXXX, Section 5.4 |
| truncate | Current | RFCXXXX, Section 5.9 |
| tzid | Current | RFCXXXX, Section 5.7 |

### 10.3. timezone Well-Known URI Registration

URI suffix: timezone

Change controller: IETF.

Specification document(s): This RFC.

Related information:

### 10.4. Service Name Registrations

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

#### 10.4.1. 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>

### 10.4.2. 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>

## 10.5. iCalendar Property Registration

This document defines the following new iCalendar property to be added to the registry defined in Section 8.2.3 of [RFC5545]:

| Property | Status | Reference |
|---|---|---|
| EQUIVALENT-TZID | Current | RFCXXXX, Section 8 |

## 11. Acknowledgements

The authors would like to thank the members of the Calendaring and Scheduling Consortium's Time Zone Technical Committee and the following individuals for contributing their ideas and support: Steve Allen, Steve Crocker, John Haug, Ciny Joy, Bryan Keller, Andrew McMillan, Ken Murchison, Arnaud Quillaud, and Jose Edvaldo Saraiva.

The authors would also like to thank the Calendaring and Scheduling Consortium for advice with this specification.

## 12. Normative References

[I-D.newton-json-content-rules]
Newton, A., "A Language for Rules Describing JSON Content", draft-newton-json-content-rules-01 (work in progress), January 2013.

[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.

[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.

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, 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.

[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.

[RFC7234] Fielding, R., Nottingham, M., and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, June 2014.

[RFC7265] Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON Format for iCalendar", RFC 7265, May 2014.

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

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: douglm@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/