

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 16, 2014

M. Douglass  
RPI  
C. Daboo  
Apple  
March 15, 2014

**Timezone Service Protocol**  
**draft-douglass-timezone-service-11**

Abstract

This document defines a timezone service protocol that allows reliable, secure and fast delivery of timezone 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 September 16, 2014.

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 . . . . .](#) [4](#)
- [1.1. Conventions . . . . .](#) [4](#)
- [1.2. Glossary of terms . . . . .](#) [4](#)
- [2. Architectural Overview . . . . .](#) [5](#)
- [3. General Considerations . . . . .](#) [6](#)
- [3.1. Timezone Identifiers . . . . .](#) [6](#)
- [3.2. Timezone Aliases . . . . .](#) [7](#)
- [3.3. Timezone Localized Names . . . . .](#) [7](#)
- [3.4. Truncated Timezones . . . . .](#) [7](#)
- [4. Timezones Service Protocol . . . . .](#) [8](#)
- [4.1. Server Protocol . . . . .](#) [8](#)
- [4.1.1. Timezone Queries . . . . .](#) [8](#)
- [4.1.2. Timezone Formats . . . . .](#) [9](#)
- [4.1.3. Conditional Timezone Requests . . . . .](#) [9](#)
- [4.1.4. Expanded Timezone Data . . . . .](#) [9](#)
- [4.1.5. Server Requirements . . . . .](#) [10](#)
- [4.1.6. Error Responses . . . . .](#) [10](#)
- [4.1.7. Extensions . . . . .](#) [10](#)
- [4.2. Client Guidelines . . . . .](#) [10](#)
- [4.2.1. Discovery . . . . .](#) [10](#)
- [4.2.1.1. Timezone Service SRV Service Labels . . . . .](#) [11](#)
- [4.2.1.2. Timezone Service TXT records . . . . .](#) [11](#)
- [4.2.1.3. Timezone Service Well-Known URI . . . . .](#) [12](#)
- [4.2.1.3.1. Example: well-known URI redirects to  
                  actual context path . . . . .](#) [12](#)
- [4.2.2. Initial Synchronization of All Timezones . . . . .](#) [12](#)
- [4.2.3. Subsequent Synchronization of All Timezones . . . . .](#) [13](#)
- [5. Request Parameters . . . . .](#) [13](#)
- [5.1. "action" Parameter . . . . .](#) [13](#)
- [5.2. "format" Parameter . . . . .](#) [13](#)
- [5.3. "changedsince" Parameter . . . . .](#) [14](#)
- [5.4. "start" Parameter . . . . .](#) [14](#)
- [5.5. "end" Parameter . . . . .](#) [14](#)
- [5.6. "lang" Parameter . . . . .](#) [14](#)
- [5.7. "tzid" Parameter . . . . .](#) [15](#)
- [5.8. "name" Parameter . . . . .](#) [15](#)
- [5.9. "truncate" Parameter . . . . .](#) [15](#)
- [6. Actions . . . . .](#) [16](#)
- [6.1. "capabilities" Action . . . . .](#) [16](#)
- [6.1.1. Example: Get Capabilities . . . . .](#) [17](#)
- [6.2. "list" Action . . . . .](#) [19](#)
- [6.2.1. Example: List timezone identifiers . . . . .](#) [21](#)
- [6.3. "get" Action . . . . .](#) [21](#)
- [6.3.1. Example: Get timezone . . . . .](#) [23](#)
- [6.3.2. Example: Get timezone alias . . . . .](#) [24](#)
- [6.3.3. Example: Get truncated timezone . . . . .](#) [24](#)



- [6.4. "expand" Action . . . . .](#) [25](#)
- [6.4.1. Example: Expanded JSON Data Format . . . . .](#) [27](#)
- [6.5. "find" Action . . . . .](#) [27](#)
- [6.5.1. Example: Find action . . . . .](#) [29](#)
- [7. JSON Definitions . . . . .](#) [30](#)
- [7.1. capabilities action response . . . . .](#) [30](#)
- [7.2. list action response . . . . .](#) [32](#)
- [7.3. expand action response . . . . .](#) [34](#)
- [7.4. error response . . . . .](#) [35](#)
- [8. Equivalent Timezone Identifier Property . . . . .](#) [35](#)
- [9. Security Considerations . . . . .](#) [36](#)
- [10. IANA Considerations . . . . .](#) [36](#)
- [10.1. Service Actions Registration . . . . .](#) [37](#)
- [10.1.1. Service Actions Registration Procedure . . . . .](#) [37](#)
- [10.1.2. Registration Template for Actions . . . . .](#) [37](#)
- [10.1.3. Registration Template for Action Parameters . . . . .](#) [38](#)
- [10.2. Initial Timezone Service Registries . . . . .](#) [38](#)
- [10.2.1. Actions Registry . . . . .](#) [38](#)
- [10.2.2. Action Parameters Registry . . . . .](#) [38](#)
- [10.3. timezone Well-Known URI Registration . . . . .](#) [39](#)
- [10.4. Service Name Registrations . . . . .](#) [39](#)
- [10.4.1. timezone Service Name Registration . . . . .](#) [39](#)
- [10.4.2. timezones Service Name Registration . . . . .](#) [40](#)
- [10.5. iCalendar Property Registration . . . . .](#) [40](#)
- [11. Acknowledgements . . . . .](#) [40](#)
- [12. Normative References . . . . .](#) [40](#)
- [Appendix A. Change History \(to be removed prior to](#)  
    [publication as an RFC\) . . . . .](#) [42](#)
- [Authors' Addresses . . . . .](#) [45](#)



## **1. Introduction**

Timezone data typically combines a coordinated universal time (UTC) offset with daylight saving time (DST) rules. Timezones 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 timezone data to determine the correct local time. As such they need to be kept up to date with changes to timezone data. To date there has been no fast and easy way to do that. Timezone 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 timezone data being present for significant amounts of time.

This specification defines a timezone service protocol that allows for fast, reliable and accurate delivery of timezone data to client systems. This protocol is based on HTTP [[RFC2616](#)] using a REST style API, with JSON [[RFC7159](#)] responses.

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

Discussion of this document should take place on the calsify mailing list [calsify@ietf.org](mailto:calsify@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.



Timezone Data: Data that defines a single timezone, including an identifier, UTC offset values, and DST rules;

Timezone Server: A server implementing the Timezone Service Protocol defined by this specification;

Timezone Identifier: A globally unique name which identifies timezone data.

### 2. Architectural Overview

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

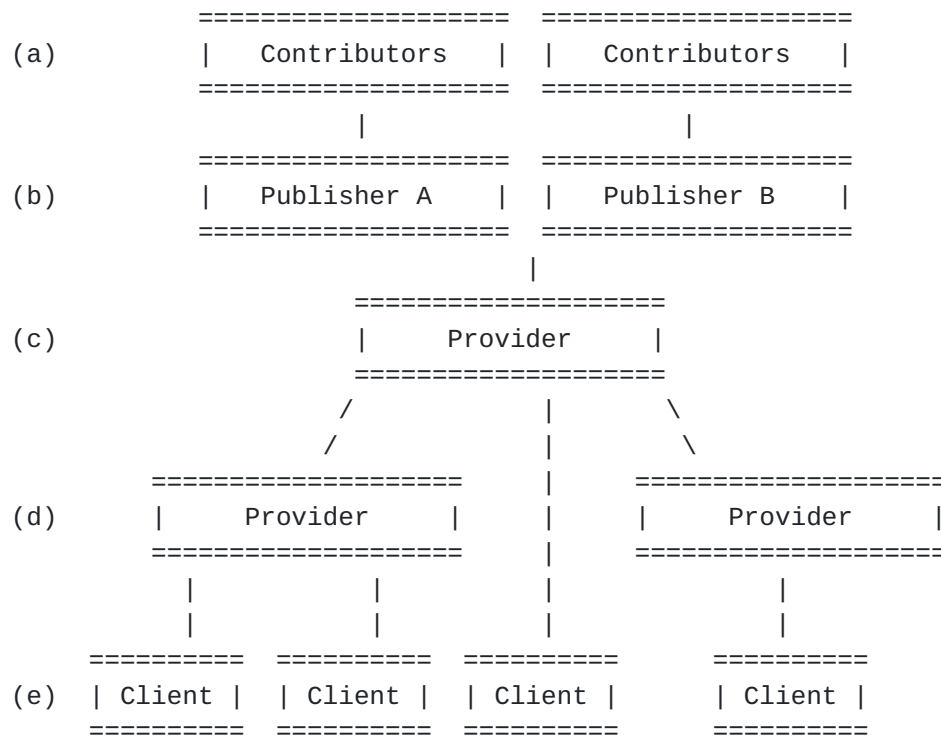


Figure 1: Timezone Service Architecture

The overall service is made up of several layers:

(a) Contributors: Individuals, governments or organizations which provide information about timezones 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 timezone 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 timezone data from publishers and make that available to other servers or clients. There can be many root providers. Root providers can choose to supply timezone 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 timezone 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. Timezone Identifiers**

Timezone identifiers are unique names associated with each timezone, as defined by publishers. The iCalendar [[RFC5545](#)] specification has a "TZID" property and parameter whose value is set to the



corresponding timezone identifier, and used to identify timezone data and relate timezones to start and end dates in events, etc. This specification does not define what format of timezone identifiers should be used. It is possible that timezone 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) timezone identifier naming scheme is out of scope for this specification.

### **3.2. Timezone Aliases**

Timezone aliases map a name onto a timezone identifier. For example "US/Eastern" is usually mapped on to "America/New\_York". Timezone aliases are typically used interchangeably with timezone identifiers when presenting information to users.

A timezone service needs to maintain timezone alias mapping information, and expose that data to clients as well as allow clients to query for timezone data using aliases. When returning timezone 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. Timezone Localized Names**

Localized names are names for timezones which can be presented to a user in their own language. Each timezone 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 timezone service might need to maintain localized name information, for one or more chosen languages, as well as allow clients to query for timezone data using localized names.

### **3.4. Truncated Timezones**

Timezones and daylight saving times rules have been in use for over a century. Timezone 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 timezone 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 timezone 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 timezone 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 timezone'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 "UTC-OFFSET-FROM" and "UTC-OFFSET-TO" 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 timezone data in the truncated "VTIMEZONE" component.

## **4. Timezones Service Protocol**

### **4.1. Server Protocol**

The timezone service protocol uses HTTP [[RFC2616](#)] 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. Timezone Queries**

Timezone identifiers, aliases or localized names can be used to query for timezone 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. Timezone Formats**

The default format for returning timezone definitions is the iCalendar [[RFC5545](#)] data format. In addition, the iCalendar-in-XML [[RFC6321](#)], and iCalendar-in-JSON [[I-D.ietf-jcardcal-jcal](#)] representations are also available. The "format" request-URI parameter can be used to select which data format is returned.

#### **4.1.3. Conditional Timezone Requests**

Timezone data is generally slow moving, with the set of timezones 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 timezone data, so a client can do updates to any locally cached data.

When listing timezones, 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 timezone data. Clients can use a conditional "list" action (see [Section 6.2](#)), supplying a previous timestamp value, to limit the results to timezones 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 timezone. 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 timezone 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 Timezone Data**

Determining timezone 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 timezone service provides an action that allows clients to request the server to expand a timezone 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 timezone 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 timezones, from various publishers. That allows client implementations to configure a single server to get all timezone 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 timezone 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 timezone actions will be added in the future (e.g., mapping a geographical location to timezone identifiers, getting change history for timezones), 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 timezone service is located or discover it through some mechanism. To use a timezone



service, a client needs a fully qualified domain name (FQDN), port and HTTP request-URI path.

#### **4.2.1.1. Timezone 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 Timezone server that uses HTTP without transport layer security ([RFC2818]).

timezones: Identifies a Timezone 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. Timezone Service TXT records**

When SRV RRs are used to advertise a timezone 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.](#) Timezone Service Well-Known URI**

A "well-known" URI [[RFC5785](#)] is registered by this specification for the Timezone 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 timezone 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 14.9 of \[RFC2616\]](#)) 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 Timezone 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 Timezones**

When a secondary service or a client wishing to cache all timezone 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 timezone in the returned list can then be fetched and stored locally. In addition a mapping of aliases to timezones can be built.



### **[4.2.3.](#) Subsequent Synchronization of All Timezones**

A secondary service or a client caching all timezone 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 timezone in the returned list can then be fetched and stored locally.

Note, this process makes no provision for handling deleted timezones. In general it is bad practice to delete timezones as they might still be in use by consumers of timezone 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 timezone data returned by the server as a standard MIME [[RFC2046](#)] media-type. If absent, the iCalendar [[RFC5545](#)] format will be returned with the timezones 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 timezone to be targeted by an action.

Value: A timezone identifier or alias.

### **[5.8.](#) "name" Parameter**

Name: name

Description: Specify a name for queries.

Value: A timezone identifier, alias or localized name. This parameter is used when searching for matching timezones (see [Section 6.5](#)).

### **[5.9.](#) "truncate" Parameter**

Name: truncate

Description: Specify a year for timezone 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 timezone 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 timezone 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": "changesince",
          "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 timezone identifiers or the requested timezone 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 "changesince" query parameter is present its value MUST correspond to a previously returned timestamp value. When "changesince" timestamp is used, the server MUST return only those timezones 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 timezone 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 timezone 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 timezone identifiers**

In this example the client requests the timezone 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 timezone. If a single timezone is specified, the response MUST contain an ETag response header field indicating the current value of the strong entity tag of the timezone resource.

If the identifier is actually a timezone alias, the server will return the matching timezone data with the alias as the identifier in the timezone data. The server MAY include one or more "EQUIVALENT-TZID" properties (see [Section 8](#)) in the timezone 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 timezone 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 timezone 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 is not present, or appears more than once, or has an invalid year specified as its value.



**6.3.1. Example: Get timezone**

In this example the client requests the timezone with a specific timezone 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 timezone alias**

In this example the client requests the timezone with an aliased timezone identifier to be returned, and the server returns the timezone 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 timezone**

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 timezone with a specific timezone 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 timezone 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.

changesince OPTIONAL, but MUST occur only once. If present, its value MUST be taken from the "dstamp" result of a previous expand result. If the targeted timezone 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 "dstamp" member and an "observances" member, see [Section 7.3](#). The server MUST include an expanded observance representing the timezone 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 timezone 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-changesince The "changesince" 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 timezone 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 timezone 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 timezone 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 timezone "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 timezone data provided by a "primary" server  
primary\_source "primary-source" : string

; The timezone server from which data is provided by a "secondary"

; server

```
secondary_source "secondary-source" : uri
```

; Present if the server is providing truncated timezone data. The

; value is the truncation date-time. Timezone 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 timezone 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 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 timezone objects
timezones "timezones" [ * timezone ]

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

; Timezone identifier
tzid "tzid" : string

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

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

; An array that lists the set of localized names available
; for the corresponding timezone
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 timezone objects  
observances "observances" [ \* observance ]

; Information about a timezone 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 Timezone Identifier Property**

Property Name: EQUIVALENT-TZID

Purpose: This property specifies an equivalent timezone identifier representing the same timezone 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 timezone identifier for a "VTIMEZONE" component when the "TZID" property of the timezone 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

Timezone 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 timezone data is used. Servers providing a timezone service MUST support HTTP over Transport Layer Security (TLS) (as defined by [\[RFC2818\]](#)) with a valid certificate. Clients and servers making use of a timezone service SHOULD use HTTP over TLS and verify the authenticity of the service being used before accepting and using any timezone 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.

Timezone 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 timezone 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 timezone service actions with IANA.

### **10.1.1. Service Actions Registration Procedure**

The IETF will create a mailing list, `timezone-service@ietf.org`, which can be used for public discussion of timezone 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 `timezone-service@ietf.org` mailing list and review registrations.

Registration of new timezone 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 timezone 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 `timezone-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 timezone 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 Timezone Service Registries**

The IANA is requested to create and maintain the following registries for timezone 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, <a href="#">Section 6.1</a>
list	Current	RFCXXXX, <a href="#">Section 6.2</a>
get	Current	RFCXXXX, <a href="#">Section 6.3</a>
expand	Current	RFCXXXX, <a href="#">Section 6.4</a>
find	Current	RFCXXXX, <a href="#">Section 6.5</a>

**10.2.2. Action Parameters Registry**

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



Parameter	Status	Reference
action	Current	RFCXXXX, <a href="#">Section 5.1</a>
changedsince	Current	RFCXXXX, <a href="#">Section 5.3</a>
end	Current	RFCXXXX, <a href="#">Section 5.5</a>
format	Current	RFCXXXX, <a href="#">Section 5.2</a>
lang	Current	RFCXXXX, <a href="#">Section 5.6</a>
name	Current	RFCXXXX, <a href="#">Section 5.8</a>
start	Current	RFCXXXX, <a href="#">Section 5.4</a>
truncate	Current	RFCXXXX, <a href="#">Section 5.9</a>
tzid	Current	RFCXXXX, <a href="#">Section 5.7</a>

### **[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](mailto:iesg@ietf.org)>

Contact: IETF Chair <[chair@ietf.org](mailto:chair@ietf.org)>

Description: Timezone Service Protocol - non-TLS

Reference: [[draft-douglass-timezone-service](#)]

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: Timezone Service Protocol - over TLS

Reference: [[draft-douglass-timezone-service](#)]

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, <a href="#">Section 8</a>

**11. Acknowledgements**

The authors would like to thank the members of the Calendaring and Scheduling Consortium's Timezone 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.ietf-jcardcal-jcal]  
 Kewisch, P., Daboo, C., and M. Douglass, "jCal: The JSON format for iCalendar", [draft-ietf-jcardcal-jcal-09](#) (work in progress), February 2014.



## [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.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", [RFC 2616](#), June 1999.

[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.

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

Changes for -11

1. start/end query parameter values are now just year numbers, no dates.

Changes for -10

1. Expand start/end query parameter values are now years rather than a date/date-time.
2. Added tzid-not-found error code for get and expand actions.

Changes for -09

1. Servers are allowed to truncate timezone data but need to advertise when they do so. Clients can select from server-specified truncations.
2. Explicitly list suggested polling intervals.
3. Removed used of \* for tzid value.
4. Removed substitute alias.
5. Added EQUIVALENT-TZID property.
6. Added more details on truncation.



7. Various editorial issues and clarifications.

Changes for -08

1. Editorial changes.
2. Fixed JSON content rule syntax.
3. Added a "version" to capabilities.
4. Changed "error" member to a string.
5. Added error codes.
6. Updated reference.
7. Removed inactive timezone feature and returnall parameter.

Changes for -07

1. Switched to JSON instead of XML and clean-ed up schema a little bit.
2. Added changedsince to expand action.
3. Added find into registry table.
4. Re-organized some sections.

Changes for -06

1. Refresh prior to last call

Changes for -05

1. Replaced reference to draft RFC with [RFC6557](#) and [RFC6125](#).
2. New XML namespace contact.
3. Templates for service name.
4. Various typos fixed.
5. More acknowledgements.

Changes for -04



1. Replaced reference to [RFC4646](#) with reference to [RFC5646](#)
2. New wording on polling.

Changes for -03

1. Replaced erroneous reference to ISO3036 with reference to [RFC4646](#)
2. Update reference to iCalendar in XML ([RFC6321](#))
3. More description of ids/aliases/names
4. Add substitute-alias parameter for action=get
5. Allow tzid on list
6. Added name request parameter
7. Added find action

Changes for -02

1. Missed definitions of the inactive element
2. Restrict.UtcOffsetFromType,.UtcOffsetToType to a pattern - allow seconds.
3. Use restricted XML dateTime as base for onset
4. Use restricted XML dateTime for lastmodified and dtstamp
5. Note that 0 and 1 are valid values for an XML boolean.
6. Set pref to a default value of false
7. Server will now set tzid of aliased timezones to the alias name
8. Remove returnaliases option
9. Aliases should not have lang attribute - removed
10. Add text on status codes and an error element
11. Added capabilities info element containing source | primary-source and contacts.



Authors' Addresses

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

Email: [doug1m@rpi.edu](mailto:doug1m@rpi.edu)  
URI: <http://www.rpi.edu/>

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

Email: [cyrus@daboo.name](mailto:cyrus@daboo.name)  
URI: <http://www.apple.com/>



