CalConnect TC

iCalendar Timezone Problems and Recommendations

Published Report

Warning for drafts

This document is not a CalConnect Standard. It is distributed for review and comment, and is subject to change without notice and may not be referred to as a Standard. Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

The Calendaring and Scheduling Consortium, Inc. 2006

:2006

The Calendaring and Scheduling Consortium, Inc.

4390 Chaffin Lane
McKinleyville
California 95519
United States of America

copyright@calconnect.org
www.calconnect.org

Contents

:2006

Foreword

This document incorporates by reference the CalConnect Intellectual Property Rights, Appropriate Usage, Trademarks and Disclaimer of Warranty for External (Public) Documents as located at

http://www.calconnect.org/documents/disclaimerpublic.pdf.

This document was created by the TIMEZONE Technical Committee of the Calendaring and Scheduling Consortium. It contains information about common time zone implementation issues and recommendations on how to resolve these issues. We first justify the need for time zones, then we identify time zone implementation problems, and then offer some guidelines and recommendations. Take note that the listed recommendations often describe what vendors are currently doing to solve the problems.

Introduction

This document contains information about common time zone implementation issues and recommendations on how to resolve these issues. We first justify the need for time zones, then we identify time zone implementation problems, and then offer some guidelines and recommendations. Take note that the listed recommendations often describe what vendors are currently doing to solve the problems.

iCalendar Timezone Problems and Recommendations

1. Why do we need time zones?

This section contains emails gathered on various mailing lists (e.g.: ietfcalsify@osafoundation.org, tc-timezone-l@calconnect.org...) and a summary of discussions between Calendaring and Scheduling Consortium members.

1.1. Why should time zone information be preserved in a VCALENDAR object rather than converting to UTC?

1.1.1.

Recurring events that occur on both sides of a daylight savings time change need the appropriate time zone information to ensure they happen at the correct local time.

1.1.2.

Keeping the time zone information during the whole life of the event is important because it allows the events to be adjusted if a change to the time zone rules occurs.

1.1.3.

Interoperability with well-established products is a requirement for most calendar applications. Anything that those products support will most likely have to be supported by the majority of standards-based calendar applications, so we should keep time zones in the standard, clarifying them if needed instead of removing them entirely.

1.1.4.

It gives more context information about the meeting creation that can be used in the future if the meeting times have to be adjusted.

1.1.5.

Because the day of the week corresponding to a local time and a UTC time may be different, recurrence rules cannot reliably be expanded using only UTC time.

EXAMPLE

> A meeting occurring every first Monday of the month at 0:30 CEST would be 22:30 every Sunday in UTC. Since the first Sunday of the month is not always occurring just before the first Monday of the month, converting that event to UTC would not expand that rule correctly.

1.2. Should a VCALENDAR object contain whole VTIMEZONE objects or reference to a time zone?

Here are some arguments for having a whole VTIMEZONE object inside a VCALENDAR object instead of just a reference to an external time zone definition:

1.2.1.

Including the VTIMEZONE data in the VCALENDAR object makes the entity self contained and very portable.

1.2.2.

An application with only offline viewing capabilities does not have to worry about keeping a time zone database if all the VCALENDAR objects contain the necessary time zone information.

2. Identified implementation problems and recommendations

Implementing a standards-compliant calendar client or server that supports time zones correctly is not an easy task. This section describes some common implementation problems and recommendations in italics.

2.1. General time zone issues

2.1.1.

How should a client / server react towards existing meetings when a time zone rule changes? For example, if DST changes for a time zone, should the existing meetings created in that time zone be adjusted to the new rules or should they remain the same and let the changes be applied only to newly created events?

2.1.1.1.

In most cases, meetings created using a time zone should use the new time zone definition so that effected recurrences reflect the time zone change (Example: Lunch at noon every day). Cases where the timing has to be fixed should be created in UTC (Example: Enter a few digits on a terminal every 108 minutes), which has no dependency on time zone rules.

2.1.2.

How should a client / server handle a new time zone?

2.1.2.1.

If a client /server model is used, the server should have a means of updating the time zone list. This can be accomplished in many different ways such as keeping a checksum of the “current” time zone list; when that list changes on the server, clients simply re-download their local copies.

2.1.2.2.

If a PIM client is used (no server), a patch could be issued updating its time zone list.

2.1.3.

How should a client / server handle a time zone fork? (For example: if DST changes in the US only and not in Canada and meetings were created using an EST5EDT time zone). This could create two new time zones EST5EDT_CA for Canada and EST5EDT_US for the US (and deprecating EST5EDT).

2.1.3.1.

The GEO property might be used to detect which of the new time zones every meeting belongs in.

2.1.3.2.

The LOCATION property can also be used to detect the appropriate time zone for the meetings.

2.1.3.3.

If no information in the calendar object can be used to detect the appropriate new time zone, a default choice could be used based on where the majority of the meetings usually occur.

2.2. Consuming time zones

Quite often, applications have a list of time zones stored internally or in a non-standard registry, and when a meeting is created with a time zone, the application will:

1) Try to match the supplied time zone with its internal representation. This match can be accomplished by using any of these techniques:
   — Have an algorithm pick an internal time zone with the closest corresponding set of rules to the supplied one.
   — Try to match the `TZID` with its internal counterpart.
2) If no match can be found the application will do one of these:
   — Return an error.
   — Add the new time zone to its internal time zone list.

At this point, these problems can occur:

2.2.1.

If the application cannot add “unmatchable” time zones to its internal list, even a perfectly valid VTIMEZONE can be rejected.

2.2.1.1.

The meeting could be converted to UTC using the provided time zone, take note that this solution is less than ideal since time zone information is lost.

2.2.2.

If the application has in its internal list a `TZID` “America/New_York”, and receives a `VTIMEZONE` with a `TZID` “America/New_York” with a different definition than its internal one, the server may react unpredictably.

2.2.2.1.

If using a time zone registry the time zone definition of the registry should be the one used.

2.2.2.2.

If using an internal time zone list that can be changed, a versioning of the `TZID` could be used allowing different versions of the same `TZID`s to be kept. This would allow applications to have events using different time zones with the same `TZID`. Take note however that consuming and preserving all time zones can become quite problematic:

— Since time zone ids are often shown to users, the time zone list could become quite large and confusing over time.
— The need to “preserve” the original consumed time zone can also look like a bad idea when a time zone rule changes; meetings that have yet to occur will most likely need to be updated with the new time zone rule.

2.3. Client application time zone issues

2.3.1. How should client applications present a choice of time zones to users?

2.3.1.1.

Use TZDATA database (a.k.a. Olson Database, “America/New_York”) names.

2.3.1.2.

Use a GMT Offset scheme (“GMT+2, Cairo”...).

2.3.1.3.

Use common name presentation (EST5EDT, PST8PDT...).

2.3.1.4.

Additionally, a “clickable” world map is often used.

2.3.2. Where should a client application get its supported time zone list?

2.3.2.1.

Use TZDATA database (a.k.a. Olson Database).

2.3.2.2.

Use standardized registry.

2.3.3. How can a client application map the Operating System time zone to the calendar server time zone?

2.3.3.1.

Time zone registry with “aliases” could be used. The mapping of OS time zone to a standardized time zone database could be provided by OS Vendors or by a third party. For the long term, platform vendors should ideally start using data coming from a standardized registry.

2.3.4. How often should a client update its time zone definition

2.3.4.1.

Clients can compare a checksum of its time zone list against the server’s checksum, if it’s different; the client refreshes its time zone list with the server’s time zone list.

2.3.4.2.

By doing periodic checks on the server.

2.3.4.3.

Using a push mechanism notifying the client that a time zone has been updated. Possibly by adding a new iTIP method.

2.4. Time zone registry and service issues

2.4.1. Why do we need a time zone registry and service?

2.4.1.1.

A time zone registry would be useful in producing a standardized list of time zones.

2.4.1.2.

A time zone service protocol would be useful in defining how time zones should be retrieved and in what format.

2.4.1.3.

A time zone registry and service would promote interoperability.

2.4.1.4.

A time zone registry and service would solve the problems encountered when trying to consume time zones.

2.4.1.5.

A time zone service means having centralized time zone definitions that are easy to update.

2.4.1.6.

A time zone registry and service could open the door for time zones being sent by reference (Useful for mobile devices and other bandwidth limited platforms).

2.4.2. How should a time zone registry be implemented?

2.4.2.1.

By using an IANA registry to store time zone data with a standardized naming scheme.

2.4.3. How should a time zone service be implemented?

2.4.3.1.

By defining a standardized way to retrieve the IANA registered time zones (i.e.: `VTIMEZONE` objects accessible through an extension of CalDAV, DNS, HTTP....).

2.4.4. Who should be responsible for updating it, how can it be trusted?

2.4.4.1.

Updates could be done through: RFC, informational RFC, appointed/elected committee/individual who approves updates to the list.

Bibliography

[1] Time zone registry draft http://www.ietf.org/internet-drafts/draft-royer-timezone-registry-02.txt

[2] Windows time zones to TZID mapping http://unicode.org/cldr/data/diff/supplemental/supplemental.html#windows___tzid

[3] CalDAV Draft http://ietfreport.isoc.org/all-ids/draft-dusseault-caldav-08.txt

[4] IETF RFC 2445, F. DAWSON and D. STENERSON. *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. 1998. RFC Publisher. https://www.rfc-editor.org/info/rfc2445.

[5] IETF RFC 2446, S. SILVERBERG, S. MANSOUR, F. DAWSON and R. HOPSON. *iCalendar Transport-Independent Interoperability Protocol (iTIP) Scheduling Events, BusyTime, To-dos and Journal Entries*. 1998. RFC Publisher. https://www.rfc-editor.org/info/rfc2446.

[6] TZ Database (Olson) http://www.twinsun.com/tz/tz-link.htm

