CalConnect TC

Mobile Calendar Interoperability Test Suite

Published Specification

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. 2007

:2007

The Calendaring and Scheduling Consortium, Inc.

4390 Chaffin Lane
McKinleyville
California 95519
United States of America

copyright@calconnect.org
www.calconnect.org

Contents

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.

The Mobile Technical Committee of the Calendaring and Scheduling Consortium created this document. It describes a test suite to assess a mobile device's capability to synchronize calendar data with a calendar store.

Introduction

This document describes the test steps required to assess a mobile device's capability to synchronize calendar data with a calendar store (referred to from now on as "the server").

It is provided as a resource for implementers to help promote interoperability, but the successful execution of the test cases described herein should not be considered as any type of formal validation.

The tests described in this document are to be manually executed. Data synchronization sessions will be initiated between the mobile device under test and the server. A desktop or web based calendar client should be used to enter and verify data on the server.

The testing method used will be a black box testing approach, wherein inputs will be provided from both the server and device and outputs will be generated. Comparisons between the actual output and the expected output will determine the capability of the device to synchronize calendar data.

The test cases have been written with the understanding that some form of synchronization is occurring, but they make no assumptions concerning the underlying methodology. They can be used for testing OMA Data Synchronization (formerly SyncML) implementations, cable based synchronization solutions, or even mobile clients which reconcile a local store using CalDAV (or their own proprietary protocol). It should also be possible to test direct data transfer between devices using short link connectivity protocols such as Bluetooth or Infrared with minimal changes.

The test cases assume that the underlying data exchange formats used will be iCalendar and vCard 3.0.

It is possible to use these test cases with implementations that use vCalendar 1.0, vCard 2.1, or some other proprietary format. Arguments supporting the importance of iCalendar to the Mobile Industry are discussed in the Consortium white paper on The Benefits of iCalendar for the Mobile Industry.

The intent of this test suite is to verify the validity of the calendar data that is exchanged and to guarantee that the calendar user has an accurate representation of their calendar regardless of which side (mobile device or server) they are viewing it from.

It does not attempt to go beyond this scope, so it does not include tests for such things as:

— Conflict Resolution
— Correct protocol usage (e.g. support for OMA DS Server Alerted Notification or Suspend/ Resume)
— Stress tests for a large volume of calendar events
— Stress tests for calendar event content, such as very long `DESCRIPTION` properties (e.g. copying full text of an email into the calendar description)
— Effects experienced when multiple synchronization technologies are used in parallel (e.g. duplicate entry problems reported when using Palm HotSync or BlackBerry clients and OMA DS in conjunction)
— Negative test conditions. In other words, incorrect input data that deliberately generates errors such as invalid recurrence patterns

Each group of tests starts with basic test cases to verify that essential calendaring data can be transferred back and forth accurately. These are followed by tests that focus on aspects that have been identified as problem areas.

Some test cases identify problem areas for further discussion at a forthcoming CalConnect Mobile Interoperability Test Event. These 'Birds of a Feather' discussion items are marked with '*BOF Topic*' in the text. References to external specifications are shown in the text in square brackets.

A complete list of limitations and current omissions within this test suite are documented in Appendix C.

Mobile Calendar Interoperability Test Suite

1. Terms and definitions

For the purposes of this document, the terms and definitions given in CC/R 1102:2013 and the following apply.

1.1.
Access control
A mechanism to grant or deny privileges (Create, Read, Update, Delete) on calendars, events, tasks or journal entries to other calendar users.

1.2.
Alarm
A reminder for an event or a to-do. Alarms may be used to define a reminder for a pending event or an overdue to-do.

1.3.
Calendar
A collection of events, to-dos, journal entries, etc. A calendar could be the content of a person or resource's agenda; it could also be a collection of data serving a more specialized need. Calendars are the basic storage containers for calendaring information.

[SOURCE: IETF RFC 3283]

1.4.
Calendar User
CU
An entity (often a human) that accesses calendar information.

[SOURCE: IETF RFC 3283]

1.5.
Calendaring
An application domain that covers systems that allow the interchange, access and management of calendar data.

1.6.
CalConnect
The Calendaring and Scheduling Consortium consisting of vendors and user groups interested in promoting and improving calendaring and scheduling standards and interoperability.

1.7.
Coordinated Universal Time
UTC
An atomic realization of Universal Time (UT) or Greenwich Mean Time, the astronomical basis for civil time. Time zones around the world are expressed as positive and negative offsets from UT. UTC differs by an integral number of seconds from International Atomic Time (TAI), as measured by atomic clocks and a fractional number of seconds from UT.

[SOURCE: Wikipedia]

1.8.
Daylight Savings Time
DST
The period of the year in which the local time of a particular time zone is adjusted forward, most commonly by one hour, to account for the additional hours of daylight during summer months.

1.9.
Event
A calendar object that usually takes up time on an individual calendar. Events are commonly used to represent meetings, appointments, anniversaries, and day events.

1.10.
iCalendar
The Internet Calendaring and Scheduling Core Object Specification. An IETF standard (RFC 2445) for a text representation of calendar data.

1.11.
Instance
When used with recurrences, an instance refers to an item in the set of recurring items.

1.12.
Invite
To request the attendance of someone to a calendar event.

1.13.
Notification
1) The action of making known, an intimation, a notice.
2) Reminder or alarm sent when any resource or parties interested in the resource need an indicator that some attention is required. Possible notification methods include email, paging, audible signal at the computer, visual indicator at the computer, voice mail, telephone.

1.14.
Organizer
The originator of a calendar event typically involving more than one attendee.

1.15.
Priority
A level of importance and/or urgency calendar users can apply to Tasks and Events.

1.16.
Recurring
Happening more than once over a specified interval, such as weekly, monthly, daily, etc. See *Repeating* (Clause 1.17).

1.17.
Repeating
An event that happens more than once. You might want an event to occur on a regular basis. To do this you schedule a repeating event. Any changes you make to the event can automatically be made to all occurrences of the event. If necessary, changes can be made to individual events without affecting the others. For example, if you need to attend a weekly meeting, you can schedule a repeating event on your calendar. Using another example, if you want to schedule a five day vacation, schedule an all-day event that repeats daily for a total of five times. If you have to cancel one of the days, delete the one day without deleting the whole event.

1.18.
Reminders
See *Notification* (Clause 1.13).

1.19.
Time zone
Areas of the Earth that have adopted the same local time. Time zones are generally centered on meridians of a longitude, that is a multiple of 15 °, thus making neighboring time zones one hour apart. However, the one hour separation is not universal and the shapes of time zones can be quite

irregular because they usually follow the boundaries of states, countries or other administrative areas.

[SOURCE: Wikipedia]

2. Abbreviated terms

BOF	Birds of a Feather
CJK	Chinese, Japanese, Korean languages
GMT	Greenwich Mean Time
IETF	Internet Engineering Task Force
IOP	Interoperability
OMA	Open Mobile Alliance
OMA	DS Open Mobile Alliance Data Synchronization (formerly SyncML)
PIM	Personal Information Management
URL	Uniform Resource Locator
UTC	Universal Time Coordinated

3. Part I — Calendar Tests

3.1. Events

The following set of tests verifies that basic events can be passed back and forth between the mobile device and the server. They also attempt to verify the following:

— that events with large amounts of data do not adversely affect the mobile device.
— that any form of truncation that may need to occur on the device does not adversely effect the representation on the server
— that reminders can be part of the event
— that appropriate mappings are done for access level and priorities
— that special characters and multi-byte characters can be correctly displayed on either side.

NOTE 1 Appendix A — Special Information, has additional information on truncations, mappings, reminders, special characters, and multi-byte characters.

NOTE 2 All tests should be performed in succession.

Table 1 — Server to Device

Test ID	Objective	Procedure	Expected Result
1.1 Create an Event with a Reminder	Verify that basic events synchronize to the device. Verify that filling out all fields with long strings does	From the Server, create a simple future event, filling out all fields with maximum input, with a reminder. Perform a synchronization	The event should display on the device with all fields on the server correctly mapped to corresponding fields on the device. The device can only display what it supports and perhaps may need to truncate certain fields but the user should see an accurate representation of the

Test ID	Objective	Procedure	Expected Result
	not cause the device issues Verify that reminders can be sent	From the device, modify the event and remove the reminder. Perform a synchronization	event. The reminder should also be set on the device. After making modifications and synchronizing, the changes should display on the server as well. There should be no evidence of any truncation server-side if truncation occurred client-side but the field itself was not actually part of the modification done. The reminder should be removed from the server-side event.
1.2 Access Level and Priority	Verify that access level and priority values in events correctly synchronize to the device. Verify that any form of mapping that occurs does not have adverse effects	From the Server, create an event with default access and priority (e.g. Normal/Medium). Perform a synchronization From the device, modify the event. Perform a synchronization Repeat making sure to test all supported access level and priorities.	The event should display on the device with the access level and priority appropriately mapped if needed. After making modifications and synchronizing, the changes should display on the server as well. There should be no evidence of any change server-side to the access level or priority if not actually part of the modification done
1.3 Special Characters From Server	Verify that special characters in events correctly synchronize to the device.	From the Server, create an event filling out all fields with special characters. Perform a synchronization From the device, modify the event. Perform a synchronization	The event should display on the device with all fields on the server correctly mapped to corresponding fields on the device. All special characters should correctly display on the device as it is displayed on the server. After modifying the event, the characters should remain correctly displayed on both the device and server and the changes made on the device should be reflected on the server
1.4 Multi-Byte Characters From Server	Verify that multi-byte characters in events correctly synchronize to the device.	From the server, create a meeting filling out all fields with multi-byte characters. Perform a synchronization From the device modify the event. Perform a synchronization	The event should display on the device with all multi-byte characters correctly displayed. After modifying the event, the multi-byte characters should remain correctly displayed on both the device and server, and the changes made on the device should be reflected on the server.

Test ID	Objective	Procedure	Expected Result
1.5 Deletion	Verify that events deleted server side are deleted on the device	From the server delete all events created in previous tests. Perform a synchronization	The deleted events should not be displayed on the device

Table 2 — Device to Server

Test ID	Objective	Procedure	Expected Result
1.6 Create an Event with a Reminder	Verify that basic events synchronize to the server. Verify that reminders can be sent	From the device, create a simple future event with a reminder. Perform a synchronization From the server, edit the event and remove the reminder. Perform a synchronization	The event should display on the server with all fields mapped correctly. The reminder should be set on the server as well. After editing the meeting and syncing, the meeting should be updated on the device. The reminder should be removed from the device-side event.
1.7 Access Level and Priority (can only be done if device supports setting an access level or priority)	Verify that access level and priority values in events correctly synchronize to the server.	From the device, create an event with default access and priority (e.g. Normal/ Medium). Perform a synchronization Repeat making sure to test all supported access level and priorities.	The event should display on the server with the access level and priority appropriately mapped if needed.
1.8 Special Characters from Device	Verify that special characters in events correctly synchronize to the server.	From the device, create an event filling out all fields with special characters. Perform a synchronization From the server, modify the event. Perform a synchronization	The event should display on the server with all fields on the device correctly mapped to corresponding fields on the server. All special characters should correctly display on the server as it is displayed on the device. After modifying the event, the characters should remain correctly displayed on both the device and server.
1.9 Multi-Byte Characters from Device	Verify that multi-byte characters in meetings correctly synchronize to the server.	From the device, create a meeting filling out all fields with multi-byte characters. Perform a synchronization	The event should display on the server with all fields on the device correctly mapped to corresponding fields on the server. All multi-byte character should be displayed correctly. After modifying the event, the multi-byte characters should

Test ID	Objective	Procedure	Expected Result
		From the server, modify the event Perform a synchronization	remain correctly displayed on both the device and server, and the changes on the server should be reflected on the device.
1.10 Deletion	Verify that events deleted device side are deleted on the server	From the device delete all events created in previous tests. Perform a synchronization	The deleted events should be deleted off of the server as well.

3.2. All Day Events

The following set of tests verifies that all-day events can be passed back and forth between the mobile device and the server. They also attempt to verify the following:

— that all day events locked to a specific day remain locked to that day.
— that all day events which span multiple days can be handled

NOTE 1 Appendix A — Special Information, has additional information on all day events.

NOTE 2 All tests should be performed in succession.

Table 3 — Server to Device

Test ID	Objective	Procedure	Expected Result
2.1 Create all-day event in same time zone	Verify that all-day events can be synchronized when server and device are in the same time zone	Verify that time zone selected on server and mobile device is the same. Create an all-day event on server on 6th December 2006 (start date 06/12/06, end date 06/12/06) with a subject of 'all-day event'. Synchronize with mobile device.	Event should display as an all-day event in the device calendar on 06/12/06. If the mobile device uses an icon to distinguish an all-day event from a timed appointment this is displayed for this entry.
2.2 Create all-day event to device in different time zone	Verify that all-day events can be synchronized when server and device are in a different time zone	Set the time zone on the server to GMT (London) and the time zone on the mobile device to GMT-5 (Eastern Time, US & Canada) Create an all-day event on server on 6th December 2006 (start date 06/12/06, end date 06/12/06) with a subject of 'all-day event'. Synchronize with mobile device.	Event should display as an all-day event in the mobile device on 06/12/06. If the mobile device uses an icon to distinguish an all-day event from a timed appointment this is displayed for this entry.
2.3 Create a Single Instance All	Verify that basic calendar entries	From the Server, create a single instance future	The day event should display on the device as an all day

Test ID	Objective	Procedure	Expected Result
Day Event with Reminder	synchronize to the device.	all-day event, filling out all fields with maximum input, with a reminder. Perform a synchronization From the device, modify the day event and remove the reminder. Perform a synchronization	event with all fields on the server correctly mapped to corresponding fields on the device. The reminder should also be set on the device. After making modifications and synchronizing, the changes should display on the server as well. Any client side truncation of fields should not be propagated back to the server. *BOF Topic*: What form should the iCalendar be to represent a day event? Currently the vCalendar to send depends greatly on the manufacturer of the device.
2.4 Create an anniversary all-day event	Verify that anniversaries can be synchronized	Create an anniversary on the server on 4th May 2007 Perform a synchronization	The anniversary should display on the device on 4th May 2007
2.5 All-day event on last day of month & last day of year check	Verify boundary conditions	Create an all-day event/ anniversary on 31st March 2007 and 31st December 2007	The all-day event/anniversary should display on the device on 31/03/2007 and 31/12/2007
2.6 Create a Single Instance Holiday with Reminder	Verify that basic calendar entries synchronize to the device.	Perform previous test cases, but for Holidays instead of All Day Events A 'Holiday' is a special type of all-day event supported by some calendar products Holidays may not be supported in the same fashion for all systems.	The Holiday should display on the device as something appropriate (as a holiday or an all day event depending on what the device can support) with all fields on the server correctly mapped to corresponding fields on the device. On the ensuing synchronization changes should not be propagated to the server as holidays can't be changed. *BOF Topic*: What should be expected behaviour? Should a modify be sent back to the device to put the holiday back? Some systems would allow the holiday to be removed. How do you specify that something is a holiday in iCalendar?
2.7 Update an all-day event on server and synchronize back to mobile device in same time zone	Verify that all-day event modifications can be synchronized correctly	Verify that time zone selected on server and mobile device is the same. Create an all-day event on server on 6th December	Event title on device calendar should be modified to 'all-day event modified' and remain an (untimed) all-day event.

Test ID	Objective	Procedure	Expected Result
		2006 (start date 06/12/06, end date 06/12/06) with a subject of 'all-day event'. Perform a synchronization Update all-day event on server and modify subject to 'all-day event modified'. Perform a synchronization	If the device calendar application uses an icon to distinguish an all-day event from a timed appointment this is displayed for this entry.
2.8 Update an all-day event on server and synchronize back to a device in different time zone	Verify that all-day event modifications can be synchronized correctly	Set the time zone on the server to GMT (London) and the time zone on the mobile device to GMT-5 (Eastern Time, US & Canada). Create an all-day event on server on 6th December 2006 (start date 06/12/06, end date 06/12/06) with a subject of 'all-day event'. Perform a synchronization Update all-day event on server and modify subject to 'all-day event modified'. Perform a synchronization	Event title on device calendar should be modified to 'all-day event modified' and remain an (untimed) all-day event. If the device calendar application uses an icon to distinguish an all-day event from a timed appointment this is displayed for this entry.
2.9 Create a Single Instance Multi-day Day Event	Verify that basic calendar entries synchronize to the device.	For multi-day Day Events, please read Appendix A before testing. From the Server, create a single instance Day Event that starts tomorrow and ends 3 days later. Make sure this does not span outside your synchronization range. Perform a synchronization From the device, modify the end date to end 1 day earlier than the previous end date (If the device supports multi-day Day Events) Perform a synchronization	For devices that support multi-day Day Events, the entry should display on the device with a Day Event that spans for the 4 days, as it is on the server. For devices that do not support multi-day Day Events results may vary. After modifying the end date from the device and synchronizing, the entry on the server should be one day shorter, reflecting the change made on the device. *BOF Topic*: What should be expected behaviour? What should iCalendar look like?
2.10 Remove Single Instance Meeting, Day Event, and Holiday	Verify that a basic deletion synchronize to the device.	From the Server, delete a single instance meeting, day event, and holiday	All the selected entries are removed from the device. This should not affect any of the other existing entries.

Test ID	Objective	Procedure	Expected Result
		Perform a synchronization	

Table 4 — Device to Server

Test ID	Objective	Procedure	Expected Result
2.11 Create an all-day event and synchronize to a server in same time zone	Verify that basic calendar entries can be synchronized to the server	Verify that time zone selected on server and mobile device is the same. Create an all-day event on the mobile device on 6th December 2006 (start date 06/12/06, end date 06/12/06) with a subject of 'all-day event'. Perform a synchronization	Event should display on the server as an all-day event on 06/12/06. If the server calendar application uses an icon to distinguish an all-day event from a timed appointment this is displayed for this entry.
2.12 Create an all-day event and synchronize to a server in different time zone	Verify that basic calendar entries can be synchronized to the server	Set the time zone on the server to GMT (London) and the time zone on the mobile device to GMT-5 (Eastern Time, US & Canada). Create an all-day event on the mobile device on 6th December 2006 (start date 06/12/06, end date 06/12/06) with a subject of 'all-day event'. Perform a synchronization	Event should display on the server as an all-day event on 06/12/06. If the server calendar application uses an icon to distinguish an all-day event from a timed appointment this is displayed for this entry.
2.13 Create a Single Instance All Day Event with Reminder	Verify that basic calendar entries can be synchronized to the server	On the device, create a single instance future all-day event, filling out all fields with maximum input, with a reminder. Perform a synchronization From the device, modify the day event and remove the reminder. Perform a synchronization	The day event should display on the server as an all day event with all fields on the server correctly mapped to corresponding fields on the device. The reminder should also be set on the server. After making modifications and synchronizing, the changes should display on the server as well. Any client side truncation of fields should not be propagated back to the server. *BOF Topic*: What form should the iCalendar be to represent a day event?
2.14 Create an anniversary all-day event	Verify that anniversaries can be synchronized	Create an anniversary on the device on 4th May 2007 Perform a synchronization	The anniversary should display on the server on 4th May 2007

Test ID	Objective	Procedure	Expected Result
2.15 Update an all-day event on mobile device and synchronize back to server in same time zone	Verify that all-day event modifications can be synchronized correctly	Verify that time zone selected on server and mobile device is the same. Create an all-day event on server on 6th December 2006 (start date 06/12/06, end date 06/12/06) with a subject of 'all-day event'. Synchronize with mobile device. Update all-day event on mobile device and modify subject to 'all-day event modified'. Synchronize with server.	Event subject should be modified to 'all-day event modified' and remain an (untimed) all-day event. If the server calendar application uses an icon to distinguish an all-day event from a timed appointment this is displayed for this entry.
2.16 Update an all-day event on mobile device and synchronize back to a server in different time zone	Verify that all-day event modifications can be synchronized correctly	Set the time zone on the server to GMT (London) and the time zone on the mobile device to GMT-5 (Eastern Time, US & Canada). Create an all-day event on server on 6th December 2006 (start date 06/12/06, end date 06/12/06) with a subject of 'all-day event'. Synchronize with mobile device. Update all-day event on mobile device and modify subject to 'all-day event modified'. Synchronize with server.	Event subject should be modified to 'all-day event modified' and remain an (untimed) all-day event. If the server calendar application uses an icon to distinguish an all-day event from a timed appointment this is displayed for this entry.
2.17 Create a Single Instance Multi-Day Day Event	Verify that basic calendar entries can be synchronized to the server	For multi-day Day Events, please read section in Appendix A before testing. If the device allows for creation of multi-day Day Events, then create a single instance Day Event that starts tomorrow and ends 3 days later. Make sure this does not span outside your synchronization range. Perform a synchronization From the device, modify the end date to end 1 day earlier than the previous end date. Perform a synchronization	Upon the first synchronization, the multi-day Day Event should display on the Server spanning 4 days. After modifying the entry and synchronizing, the server side entry should display as being one day less.
2.18 Remove Single Instance Meeting, Day Event, and Holiday	Verify that a basic deletion synchronize to the server.	From the device, delete a single instance meeting, day event, and holiday Perform a synchronization	All the selected entries are removed from the server. This should not affect any of the other existing entries.

3.3. Repeating Entries

The following test cases verify that recurring events can be synchronized between device and server.

In particular

— Repeating calendar events can be created
— Repeating calendar events can be modified (exceptions added/removed)
— Repeating calendar events can be deleted

NOT ALL DEVICES SUPPORT RECURRING ENTRIES. Only perform these tests if the device does support the creation of recurring entries.
NOTE 1 The level of REPEATING/RECURRING meeting support will vary depending on the server. See Appendix A for explanation on possible levels of support.

NOTE 2 All tests should be performed in succession.

Table 5 — Server to Device

Test ID	Objective	Procedure	Expected Result
3.1 Create Daily Repeat (every day, bounded)	Verify that a daily repeat can be synchronized **EXAMPLE 1** Five day conference	Create an appointment on 23rd April 2007 with a daily repeat until 27th April 2007 Perform a synchronization	Device calendar should display 5 occurrences of a daily repeating meeting
3.2 Create Daily Repeat (every other day, unbounded)	Verify that a daily repeat can be synchronized **EXAMPLE 2** Meeting every other day	Create an appointment on 23rd April 2007 with a daily repeat every other day Perform a synchronization	Device calendar should display a daily repeating meeting every other day starting from 23rd April 2007 The repeat pattern should remain unbounded (in other words, it should repeat forever)
3.3 Create Daily Repeat (every 7 days, unbounded)	Verify that a daily repeat can be synchronized **EXAMPLE 3** Weekly Staff Meeting	Create an appointment on 2nd May 2007 with a daily repeat every 7 days Perform a synchronization	Device calendar should display a weekly meeting every seven days (every Wednesday) from 2nd May 2007 The repeat pattern should be the same visible pattern as a weekly repeat and it should be unbounded
3.4 Create Weekly Repeat (every Wed, unbounded)	Verify that a weekly repeat can be synchronized	Create an appointment on 2nd May 2007 with a weekly repeat on Wednesday	Device calendar should display a weekly meeting every Wednesday from 2nd May 2007

Test ID	Objective	Procedure	Expected Result
	EXAMPLE 4 Meeting every Wednesday	Perform a synchronization	The repeat pattern should remain unbounded
3.5 Create Weekly repeat (Wed & Fri, unbounded)	Verify that a weekly repeat can be synchronized **EXAMPLE 5** Swimming class held every week on Wednesday and Fridays	Create an appointment on 2nd May 2007 with a weekly repeat on Wednesday and Friday Perform a synchronization	Device calendar should display a weekly meeting with occurrences on Wednesday and Fridays from 2nd May The first two occurrences should be 02/05/07 and 04/05/07 The repeat pattern should remain unbounded
3.6 Create Fortnightly Repeat (unbounded)	Verify that a weekly repeat can be synchronized **EXAMPLE 6** Project status meeting held every two weeks	Create an appointment on 1st May 2007 with a fortnightly repeat Perform a synchronization	Device calendar should display an appointment every two weeks starting from 1st May The repeat pattern should remain unbounded
3.7 Create Monthly By Date Repeat (unbounded)	Verify that a monthly by date repeat can be synchronized **EXAMPLE 7** Utility bill payment on 15th day of the month	Create an appointment on 15th May 2007 with a monthly by date repeat Perform a synchronization	Device calendar should display an appointment on 15th of every month starting from 15th May 2007 The repeat pattern should remain unbounded
3.8 Create Monthly By Day Repeat (first occurrence, bounded)	Verify that a monthly by date repeat can be synchronized **EXAMPLE 8** Monthly meeting on the first Monday of every month	Create an appointment on 7th May 2007 repeating every month on the first Monday for a year (13 occurrences in total) Perform a synchronization	Device calendar should display an appointment on the following dates: 07/05/2007, 04/06/2007, 02/07/2007, 06/08/2007, 03/09/2007, 01/10/2007, 05/11/2007, 03/12/2007, 07/01/2008, 04/02/2008, 03/03/2008, 07/04/2008, 05/05/2008

Test ID	Objective	Procedure	Expected Result
3.9 Create Monthly By Day Repeat (nth occurrences, bounded)	Verify that a monthly by day repeat can be synchronized **EXAMPLE 9** Monthly meeting on the 2nd Tuesday of every month	Create an appointment 2nd & 3rd Sunday of every month starting on 13th May 2007 until 10th June 2007 (3 occurrences in total) Perform a synchronization	Device calendar should display an appointment on the following dates: 13/05/2007, 20/05/2007, 10/06/2007
3.10 Create Monthly By Day Repeat (last occurrence, bounded)	Verify that a monthly by day repeat can be synchronized **EXAMPLE 10** Monthly meeting on the last Friday of every month	Create an appointment on the last Friday of every month starting 25th May 2007 until 29th June 2007 (2 occurrences in total) Perform a synchronization	Device calendar should display an appointment on the following dates: 25/05/2007, 29/06/2007
3.11 Create Yearly Repeat (every year, unbounded)	Verify that anniversary events can be synchronized **EXAMPLE 11** Birthday **EXAMPLE 12** St Patrick's Day (17th March)	Create an anniversary entry for Valentine's Day (14th February 2007) on server Perform a synchronization	Device calendar should display an anniversary event on 14/02/2007 and future years (2008, 2009 etc.) The repeat pattern should remain unbounded
3.12 Create Yearly Repeat (every year for 5 years, bounded)	Verify that yearly events can be synchronized	Create an event with a yearly repeat on 1st April 2007 for five years Perform a synchronization	Device calendar should display an event on the following dates: 01/04/2007, 01/04/2008, 01/04/2009, 01/04/2010, 01/04/2011
3.13 Create Yearly Repeat (every 4 years, bounded)	Verify that yearly events can be synchronized **EXAMPLE 13** Leap year occurs every	Create an event with a yearly repeat on 29th February 2004 every four years until 01/03/2012 Perform a synchronization	Device calendar should display an event on the following dates 29/02/2004, 29/02/2008, 29/02/2012

Test ID	Objective	Procedure	Expected Result
	four years on 29th February		
3.14 Create custom repeat (RDATES only)	Verify that custom repeat can be synchronized **EXAMPLE 14** Dates for a lecture series: Tuesday this week, Wednesday next week, & Friday the following week.	Create custom repeat pattern Tuesday 1st May 2007, Wednesday 9th May 2007, Friday 18th May 2007 Perform a synchronization	Device calendar should display the event on the following dates: 01/05/2007, 09/05/2007, 18/05/2007
3.15 Create repeat combination	Verify that combinations of repeat patterns can be synchronized **EXAMPLE 15** Daylight Saving Time starts on second Sunday in March in 2007	Create appointment with a repeat combination, for example: Second Sunday in March every year (`RRULE:FREQ=YEARLY; BYMONTH=3; BYDAY=2SU`) Perform a synchronization	Device calendar should display event on 2nd Sunday of March (11th March 2007) The repeat pattern should remain unbounded
3.16 Create repeating event plus custom repeat (RRULE + RDATE)	Verify that more complex repeat patterns can be synchronized **EXAMPLE 16** Weekly meeting with an extra meeting this week	Create a weekly repeating meeting (every Monday starting 5th May 2007) and add an additional RDATE on Wed 7th May Perform a synchronization	Device calendar should display weekly repeat every Monday starting 5th May 2007 and an additional meeting occurrence on 7th May with the same event description
3.17 Create a repeating event with exceptions (RRULE + EXDATE, bounded)	Verify repeating meetings with exceptions can be synchronized **EXAMPLE 17** Daily repeating meeting	Create a daily repeating meeting starting 30th April 2007 until Friday 4th May 2007. Create meeting exception on Wednesday 2nd May (i.e. cancel meeting)	Device calendar should display daily repeat between 30/04/07 and 04/05/07 with no meeting on 02/05/07

Test ID	Objective	Procedure	Expected Result
	except for Wednesday		
3.18 Create a custom repeat with exceptions (RDATE + EXDATE, bounded)	Verify that more complex repeat patterns can be synchronized	Create custom repeat pattern Perform a synchronization	Device calendar correctly displays repeating pattern
3.19 Create repeating event plus custom repeat and exceptions (RRULE, RDATE & EXDATE)	Verify that more complex repeat patterns can be synchronized	Create custom repeat pattern Perform a synchronization	Device calendar correctly displays repeating pattern
3.20 Modify anniversary	Verify that modification of meeting details does not cause device representation to become non-repeating	Create anniversary event on 4th July 2007 on device Perform a synchronization Modify subject of event on server Perform a synchronization	Device calendar correctly displays anniversary event on 04/07/07 Server modification of event subject is correctly synchronized to client
3.21 Modify occurrences of repeating meeting	Verify that modification of the repeat rule is correctly synchronized to the device	Create daily repeating meeting starting on 1st May 2007 until 5th May (5 occurrences) Perform a synchronization Cancel occurrence on 4th May & 5th May (select this and future instances when deleting) from server Perform a synchronization	Device calendar displays repeating meeting 1-5th May After synchronization, device calendar displays meeting 1-3rd May only
3.22 Modify exceptions of repeating meeting	Verify that addition, modification and deletion of exceptions is correctly synchronized	Create daily repeating meeting starting on 1st May 2007 until 5th May (5 occurrences) Perform a synchronization Cancel occurrence on 4th May & 5th May (select this and future instances	Device calendar displays repeating meeting 1-5th May After second synchronization, device calendar displays meeting 1-3rd May After third synchronization, calendar

Test ID	Objective	Procedure	Expected Result
		when deleting) from server Perform a synchronization Extend meeting to 4th May (Modify exception) Perform a synchronization	displays meeting 1-4th May
3.23 Delete recurring meeting	Verify deletion of recurring meeting is correctly synchronized to the device	Create recurring meeting on server Perform a synchronization Delete recurring meeting from server Perform a synchronization	Device calendar displays recurring meeting After synchronization, device calendar no longer displays recurring meeting
3.24 Create Daily Repeat (every day, bounded)	Verify that a daily repeat can be synchronized **EXAMPLE 18** Five day conference	Create an appointment on 23rd April 2007 with a daily repeat until 27th April 2007 Perform a synchronization	Server calendar should display 5 occurrences of a daily repeating meeting
3.25 Create Daily Repeat (every other day, unbounded)	Verify that a daily repeat can be synchronized **EXAMPLE 19** Meeting every other day	Create an appointment on 23rd April 2007 with a daily repeat every other day Perform a synchronization	Server calendar should display a daily repeating meeting every other day starting from 23rd April 2007 The repeat pattern should remain unbounded (in other words, it should repeat forever)
3.26 Create Daily Repeat (every 7 days, unbounded)	Verify that a daily repeat can be synchronized **EXAMPLE 20** Weekly Staff Meeting	Create an appointment on 2nd May 2007 with a daily repeat every 7 days Perform a synchronization	Server calendar should display a weekly meeting every seven days (every Wednesday) from 2nd May 2007 The repeat pattern should be the same visible pattern as a weekly repeat and it should be unbounded
3.27 Create Weekly Repeat (every Wed, unbounded)	Verify that a weekly repeat can be synchronized	Create an appointment on 2nd May 2007 with	Server calendar should display a weekly meeting

Test ID	Objective	Procedure	Expected Result
	EXAMPLE 21 Meeting every Wednesday	a weekly repeat on Wednesday Perform a synchronization	every Wednesday from 2nd May 2007 The repeat pattern should remain unbounded
3.28 Create Weekly repeat (Wed & Fri, unbounded)	Verify that a weekly repeat can be synchronized **EXAMPLE 22** Swimming class held every week on Wednesday and Fridays	Create an appointment on 2nd May 2007 with a weekly repeat on Wednesday and Friday Perform a synchronization	Server calendar should display a weekly meeting with occurrences on Wednesday and Fridays from 2nd May The first two occurrences should be 02/05/07 and 04/05/07 The repeat pattern should remain unbounded
3.29 Create Fortnightly Repeat (unbounded)	Verify that a weekly repeat can be synchronized **EXAMPLE 23** Project status meeting held every two weeks	Create an appointment on 1st May 2007 with a fortnightly repeat Perform a synchronization	Server calendar should display an appointment every two weeks starting from 1st May The repeat pattern should remain unbounded
3.30 Create Monthly By Date Repeat (unbounded)	Verify that a monthly by date repeat can be synchronized **EXAMPLE 24** Utility bill payment on 15th day of the month	Create an appointment on 15th May 2007 with a monthly by date repeat Perform a synchronization	Server calendar should display an appointment on 15th of every month starting from 15th May 2007 The repeat pattern should remain unbounded
3.31 Create Monthly By Day Repeat (first occurrence, bounded)	Verify that a monthly by date repeat can be synchronized **EXAMPLE 25** Monthly meeting on the first Monday of every month	Create an appointment on 7th May 2007 repeating every month on the first Monday for a year (13 occurrences in total) Perform a synchronization	Server calendar should display an appointment on the following dates: 07/05/2007, 04/06/2007, 02/07/2007, 06/08/2007, 03/09/2007, 01/10/2007, 05/11/2007, 03/12/2007, 07/01/2008, 04/02/2008, 03/03/2008, 07/04/2008, 05/05/2008

Test ID	Objective	Procedure	Expected Result
3.32 Create Monthly By Day Repeat (nth occurrences, bounded)	Verify that a monthly by day repeat can be synchronized **EXAMPLE 26** Monthly meeting on the 2nd Tuesday of every month	Create an appointment 2nd & 3rd Sunday of every month starting on 13th May 2007 until 10th June 2007 (3 occurrences in total) Perform a synchronization	Server calendar should display an appointment on the following dates: 13/05/2007, 20/05/2007, 10/06/2007
3.33 Create Monthly By Day Repeat (last occurrence, bounded)	Verify that a monthly by day repeat can be synchronized **EXAMPLE 27** Monthly meeting on the last Friday of every month	Create an appointment on the last Friday of every month starting 25th May 2007 until 29th June 2007 (2 occurrences in total) Perform a synchronization	Server calendar should display an appointment on the following dates: 25/05/2007, 29/06/2007
3.34 Create Yearly Repeat (every year, unbounded)	Verify that anniversary events can be synchronized **EXAMPLE 28** Birthday **EXAMPLE 29** St Patrick's Day (17th March)	Create an anniversary entry for Valentine's Day (14th February 2007) on server Perform a synchronization	Server calendar should display an anniversary event on 14/02/2007 and future years (2008, 2009 etc.) The repeat pattern should remain unbounded
3.35 Create Yearly Repeat (every year for 5 years, bounded)	Verify that yearly events can be synchronized	Create an event with a yearly repeat on 1st April 2007 for five years Perform a synchronization	Server calendar should display an event on the following dates: 01/04/2007, 01/04/2008, 01/04/2009, 01/04/2010, 01/04/2011
3.36 Create Yearly Repeat (every 4 years, bounded)	Verify that yearly events can be synchronized **EXAMPLE 30** Leap year occurs every	Create an event with a yearly repeat on 29th February 2004 every four years until 01/03/2012 Perform a synchronization	Server calendar should display an event on the following dates: 29/02/2004, 29/02/2008, 29/02/2012

Test ID	Objective	Procedure	Expected Result
	four years on 29th February		
3.37 Create custom repeat (RDATES only) ***Optional test — Mobile UI may not allow creation of this type of repeat***	Verify that custom repeat can be synchronized **EXAMPLE 31** Dates for a lecture series: Tuesday this week, Wednesday next week, & Friday the following week.	Create custom repeat pattern Tuesday 1st May 2007, Wednesday 9th May 2007, Friday 18th May 2007 Perform a synchronization	Server calendar should display the event on the following dates: 01/05/2007, 09/05/2007, 18/05/2007
3.38 Create repeat combination _Optional test — Mobile UI may not allow creation of this type of repeat	Verify that combinations of repeat patterns can be synchronized **EXAMPLE 32** Daylight Saving Time starts on second Sunday in March in 2007	Create appointment with a repeat combination, for example: Second Sunday in March every year (`RRULE:FREQ=YEARLY; BYMONTH=3; BYDAY=2SU`) Perform a synchronization	Server calendar displays event on 2nd Sunday of March (11th March 2007) The repeat pattern should remain unbounded
3.39 Create repeating event plus custom repeat (RRULE + RDATE) ***Optional test — Mobile UI may not allow creation of this type of repeat***	Verify that more complex repeat patterns can be synchronized **EXAMPLE 33** Weekly meeting with an extra meeting this week	Create a weekly repeating meeting (every Monday starting 5th May 2007) and add an additional RDATE on Wed 7th May Perform a synchronization	Server calendar displays weekly repeat every Monday starting 5th May 2007 and an additional meeting occurrence on 7th May with the same event description
3.40 Create a repeating event with exceptions (RRULE + EXDATE, bounded) ***Optional test — Mobile UI may not allow creation of this type of repeat***	Verify repeating meetings with exceptions can be synchronized **EXAMPLE 34** Daily repeating meeting	Create a daily repeating meeting starting 30th April 2007 until Friday 4th May 2007. Create meeting exception on Wednesday 2nd May (i.e. cancel meeting)	Server calendar displays daily repeat between 30/04/07 and 04/05/07 with no meeting on 02/05/07

Test ID	Objective	Procedure	Expected Result
	except for Wednesday		
3.41 Create a custom repeat with exceptions (RDATE + EXDATE, bounded) ***Optional test — Mobile UI may not allow creation of this type of repeat***	Verify that more complex repeat patterns can be synchronized	Create custom repeat pattern Perform a synchronization	Server calendar correctly displays repeating pattern
3.42 Create repeating event plus custom repeat and exceptions (RRULE, RDATE & EXDATE) ***Optional test — Mobile UI may not allow creation of this type of repeat***	Verify that more complex repeat patterns can be synchronized	Create custom repeat pattern Perform a synchronization	Server calendar correctly displays repeating pattern
3.43 Modify anniversary	Verify that modification of meeting details does not cause server representation to become non-repeating	Create anniversary event on 4th July 2007 on server Perform a synchronization Modify subject of event on device Perform a synchronization	Server calendar correctly displays anniversary event on 04/07/07 Device modification of event subject is correctly synchronized to server
3.44 Modify occurrences of repeating meeting	Verify that modification of the repeat rule is correctly synchronized to the server	Create daily repeating meeting starting on 1st May 2007 until 5th May (5 occurrences) Perform a synchronization Cancel occurrence on 4th May & 5th May (select this and future instances when deleting) from device Perform a synchronization	Server calendar displays repeating meeting 1-5th May After synchronization, server calendar displays meeting 1-3rd May only
3.45 Delete recurring meeting	Verify deletion of recurring meeting is correctly synchronized to the server	Create recurring meeting on server Perform a synchronization Delete recurring meeting from device	Server calendar displays recurring meeting After synchronization, server calendar no longer displays recurring meeting

Test ID	Objective	Procedure	Expected Result
		Perform a synchronization	

3.4. Scheduling

The following set of tests verifies that basic scheduling of events can be accomplished. In particular they attempt to verify the following:

— that attendee information can be correctly displayed on a mobile device.
— that users invited to a meeting can accept or decline invitations from their mobile device.
— that users can initiate invitations from their mobile device.

NOTE 1 Appendix A — Special Information, has additional information on Scheduling.

NOTE 2 All tests should be performed in succession.

NOTE 3 In order to perform these tests the mobile device must be able to support the concept of attendees.

Test ID	Objective	Procedure	Expected Result
4.1 Create Entry as owner with Attendees from Server	Verify that basic synchronize with attendees work.	As the owner, from the server, create a meeting and a day event, each with the owner and 2 attendees. Perform a synchronization As the owner, from the server, add 1 more attendees to each entry and remove an attendee. As the original remaining attendee update your attendance status server side. Perform a synchronization	The two entries should display on the device with all attendees and the appropriate reply status. After adding and removing attendees and syncing, the new attendees should be reflected on the device as well as the updated attendance status for the original remaining attendee.
4.2 Accept Entry as Invitee from Device	Verify that modifying the reply status of an invitation is reflected on the server.	As another user, from the server, create a meeting and invite the mobile device user as well as one other user. Perform a synchronization From the device accept the invitation, server side update the attendance status for the other invite. Perform a synchronization	After the first synchronization, the meeting invitation should display on the device with all attendees displayed. After the second synchronization, the updated acceptance of the invitation should be evident server side as well as on the device. The updated attendance status of the other invitee should also display on the device.
4.3 Create Entry as owner with Attendees from Device	Verify that device initiated invitations work.	From the device, create a meeting and a day event, each with the owner and 2 attendees. Perform a synchronization	The two entries should display on the server with all attendees and the appropriate reply status. All attendees should receive invitations. After the second synchronization, the updated attendee status of the

Test ID	Objective	Procedure	Expected Result
		Server-side have the attendees accept and/or decline the invitations. Perform a synchronization	invitees should be evident device side.

3.5. Time Zones and Daylight Savings

The following set of tests verifies that events can be passed back and forth regardless of differing timezones and changes to or from daylight savings. In particular they attempt to verify the following:

— that events (simple and repeating) are displayed correctly regardless of whether or not the server and mobile device are set to the same timezone and that any change to the timezone on either side will not effect the events in any way.
— that when a synchronization date range spans over a change from daylight savings to standard time (or vice versa) that events (simple and repeating) on either side of the change are still displayed correctly and that when the current time actually does move into the new time setting there is no adverse effects.

NOTE 1 Appendix A — Special Information, has additional information on time zones and daylight savings.

NOTE 2 All tests should be performed in succession.

NOTE 3 In order to perform the daylight savings tests, you must be able to synchronize across that period of time.

Test ID	Objective	Procedure	Expected Result
5.1 Time Zones and Simple Meetings	Verify that simple meetings can be passed back and forth regardless of what time zone setting is in place on either side.	Make sure the current time zone on the server and the device matches. From the server, create a meeting that starts at 10am. Perform a synchronization Change the time zone on the device to a different time zone 3 hours later. Create a new meeting starting at 10am from the server. Perform a synchronization Change the time zone of the server to match the new time zone. Modify the title of one of the meetings. Perform a synchronization	The meeting should first display on the device at the correct time, 10am. After changing time zones, the device should automatically change the times for all entries. The first meeting should now be displayed at 7am on the device. After creating the second meeting and synchronizing, the server is still on the old time zone, so the times will differ. On the device, the first meeting is at 7am and the second at 10am, but on the server, the first meeting is at 10am, and the second at 1pm. After changing the time zone on the server and making modifications, both the device and server should display the first meeting at 7am and the second meeting at 10am. When tests are repeated the appropriate offsets should be seen but everything should still be where it is intended.

Test ID	Objective	Procedure	Expected Result
		Repeat tests starting with device and server in opposite time zones. Repeat tests with the server side meetings being created by another person inviting the owner of the mobile device where by the organizer is in a completely different time zone then those being used in the test.	
5.2 Time Zones and Repeating Meetings	Verify that repeating meetings can be passed back and forth regardless of what time zone setting is in place on either side.	Make sure the current time zone on the server and the device matches. From the server, create a repeating weekly meeting that starts at 10am. Perform a synchronization Change the time zone on the device to a different time zone 3 hours later. Create a new repeating weekly meeting starting at 10am from the server. Perform a synchronization Change the time zone of the server to match the new time zone. Modify the title of one of the meetings. Perform a synchronization Repeat tests starting with device and server in opposite time zones. Repeat tests with the server side meetings being created by another person inviting the owner of the mobile device where by the organizer is in a completely different time zone then those being used in the test.	The meetings should first display on the device at the correct time, 10am. After changing time zones, the device should automatically change the times for all entries. The meetings should now be displayed at 7am on the device. After creating the second meeting and synchronizing, the server is still on the old time zone, so the times will differ. On the device, the first meetings are at 7am and the second ones at 10am, but on the server, they are at 10am and at 1pm respectively. After changing the time zone on the server and making modifications, both the device and server should display the first set of meetings at 7am and the second set of meetings at 10am. When tests are repeated the appropriate offsets should be seen but everything should still be where it is intended.
5.3 Time Zones and All-Day Events	Verify that day events can be passed back and forth regardless of what time zone setting is in	Make sure the current time zone on the server and the device matches.	The Day Event should first display on the device. The Day Event should be displayed on the correct day on the device.

Test ID	Objective	Procedure	Expected Result
	place on either side.	From the server, create a Day Event filling out all fields. Perform a synchronization Change the time zone on the device to a different time zone 3 hours earlier. Modify the title of the Day Event from the device. Perform a synchronization Change the time zone of the server to match the new time zone. Modify the title of the Day Event from the server. Perform a synchronization Repeat tests starting with device and server in opposite time zones. Repeat tests with the server side meetings being created by another person inviting the owner of the mobile device where by the organizer is in a completely different time zone then those being used in the test.	After changing the time zone on the device, the device should automatically change the times for all the entries. After modifying and syncing, the Day Event, the Day Event should still be displayed on the same date with the new title on both server and device. After changing the time zone on the server, the Day Event should still be displayed on the same day with the new title on the device. Day Events are independent of time zone changes. When tests are repeated Day Events should continue to be seen as independent of differing time zones.
5.4 Spring Daylight Savings Single Entries from Server	Verify that entries originating form the server before and after a switch remain displayed correctly to users.	Make sure you modify the synchronization range to include the daylight savings period you are about to test. From the Server, create a single entry before and another single entry after the Spring daylight savings date at 10am, with all fields filled out. Perform a synchronization Push the time on the server and the device ahead to simulate crossing into daylight savings. Create a new meeting server side at 10am Perform a synchronization	The two entries should display on the device with all fields correctly mapped. The time of the two entries should be at 10am. After the time change the time of the two original entries and the new entry should still be at 10am When tests are repeated meetings between when the organizer's Time zone switch to DST and when the invitee's Time zone switches should be offset by an hour (e.g. A 10am London meeting will normally be a 5am Montreal meeting but it will be at 6am after the province of Quebec switches to DST up until the point where British Summer Time kicks in at which point it will go back to being at 5am).

Test ID	Objective	Procedure	Expected Result
		Repeat tests with the server side meetings being created by another person inviting the owner of the mobile device where by the organizer is in a completely different time zone then those being used in the test which has a different date for DST changes.	
5.5 Spring Daylight Savings Repeating Entry from Server	Verify that repeating entries originating form the server before and after a switch remain displayed correctly to users.	Make sure you modify the synchronization range to include the daylight savings period you are about to test. From the Server, create a repeating daily entry that starts at 10am, with all fields filled out, which spans across a Spring daylight savings date Perform a synchronization Repeat tests with the server side meetings being created by another person inviting the owner of the mobile device where by the organizer is in a completely different time zone then those being used in the test which has a different date for DST changes.	All instances of the repeating entry should display on the device. The times of **ALL** instances before and after the daylight savings date should be 10am. When tests are repeated meetings between when the organizer's Time zone switch to DST and when the invitee's Time zone switches should be offset by an hour (e.g. A 10am London meeting will normally be a 5am Montreal meeting but it will be at 6am after the province of Quebec switches to DST up until the point where British Summer Time kicks in at which point it will go back to being at 5am).
5.6 Autumn Daylight Savings Single Entries from Device	Verify that entries originating for the device before and after a switch remain displayed correctly to users.	Make sure you modify the synchronization range to include the daylight savings period you are about to test. From the device, create a single entry before and another single entry after the Autumn daylight savings date at 10am, with all fields filled out. Perform a synchronization Push the time on the server and the device ahead to simulate crossing into daylight savings. Create a new meeting device side at 10am Perform a synchronization	The two entries should display on the server with all fields correctly mapped. The time of the two entries should be at 10am. After the time change the time of the two original entries and the new entry should still be at 10am

Test ID	Objective	Procedure	Expected Result
5.7 Autumn Daylight Savings Recurring Entry from Device	Verify that repeating entries originating form the device before and after a switch remain displayed correctly to users.	Make sure you modify the synchronization range to include the daylight savings period you are about to test. From the device, create a repeating daily entry that starts at 10am, with all fields filled out, which spans across an Autumn daylight savings date. Perform a synchronization	All instances of the repeating entry should display on the Server. The times of **ALL** instances before and after the daylight savings date should be 10am

4. Part II — Task Tests

The following set of tests verifies that basic mobile task management can be accomplished. In particular they attempt to verify the following:

— that Tasks can be created, modified, and deleted on the device or server
— that Task Access Levels and Priorities can be correctly mapped
— that Task reminders can be correctly mapped
— that Tasks can be marked as completed and have this fact reflected on either side
— that special characters and multi byte characters can be correctly handled

NOTE 1 Appendix A — Special Information, has additional information on Mapping, Reminders, Special Characters, Multi Byte Characters, and Task Completion.

NOTE 2 All tests should be performed in succession.

Table 6 — Server to Device

Test ID	Objective	Procedure	Expected Result
6.1 Create task	Verify that basic task information can be synchronized	Create a task on the server Complete all fields with maximum input. Perform a synchronization Modify the task on the device. Perform a synchronization Modify the task on the server Perform a synchronization	The task should be synchronized to the device with server fields correctly mapped to the corresponding supported fields on the device. After second synchronization device modification should be propagated to the server. Any truncation that occurred device side should not affect the server. After third synchronization server modification should be reflected on the device.
6.2 Task Access Level and Priority	Verify that access level and priority values in tasks	From the Server, create a task with default access and priority (e.g. Normal/Medium).	The task should display on the device with the access level and priority appropriately mapped if needed.

Test ID	Objective	Procedure	Expected Result
	correctly synchronize to the device. Verify that any form of mapping that occurs does not have adverse effects	Perform a synchronization From the device, modify the event. Perform a synchronization Repeat making sure to test all supported access level and priorities.	After making modifications and synchronizing, the changes should display on the server as well. There should be no evidence of any change server-side to the access level or priority if not actually part of the modification done
6.3 Create task with alarm	Verify that task reminders can be synchronized	Create a task on the server Complete title, due date, priority, and access/privacy fields. Set category for task (if supported) Set an alarm reminder for both the start date and the due date. Perform a synchronization	The task should be synchronized to the device with all fields correctly mapped to the corresponding fields on the device. An appropriate mapping of the server side reminders should be present on the devices.
6.4 Mark task as completed	Verify that task completion can be synchronized	Create a task on the server Complete title, due date, priority and access/privacy fields Perform a synchronization Set task as complete on server Perform a synchronization	The task should be synchronized to the device, with all fields correctly mapped. After second synchronization, task should be marked as complete on the device
6.5 Special Characters From Server	Verify that special characters in tasks correctly synchronize to the device.	From the Server, create a task filling out all fields with special characters. Perform a synchronization From the device, modify the task. Perform a synchronization	The task should display on the device with all fields on the server correctly mapped to corresponding fields on the device. All special characters should correctly display on the device as it is displayed on the server. After modifying the task, the characters should remain correctly displayed on both the device and server and the changes made on the device should be reflected on the server

Test ID	Objective	Procedure	Expected Result
6.6 Multi-Byte Characters From Server	Verify that multi-byte characters in tasks correctly synchronize to the device.	From the server, create a task filling out all fields with multi-byte characters. Perform a synchronization From the device modify the event. Perform a synchronization	The task should display on the device with all multi-byte characters correctly displayed. After modifying the task, the multi-byte characters should remain correctly displayed on both the device and server, and the changes made on the device should be reflected on the server.
6.7 Deletion	Verify that tasks deleted server side are deleted on the device	From the server delete all tasks created in previous tests. Perform a synchronization	The deleted tasks should be deleted off of the device as well.

Table 7 — Device to Server

Test ID	Objective	Procedure	Expected Result
6.8 Create task	Verify that basic task information can be synchronized	Create a task on the device Complete all fields with maximum input. Perform a synchronization Modify the task on the device Perform a synchronization	The task should be synchronized to the server with fields correctly mapped to the corresponding supported fields on the server. After second synchronization device modification should be reflected on the server.
6.9 Task Access Level and Priority	Verify that access level and priority values in tasks correctly synchronize to the server.	From the device, create a task with default access and priority (e.g. Normal/Medium). Perform a synchronization Repeat making sure to test all supported access level and priorities.	The task should display on the server with the access level and priority appropriately mapped if needed.
6.10 Create task with alarm	Verify that task reminders can be synchronized	Create a task on the device Set an alarm reminder (as allowed by the device).	The task should be synchronized to the device with all fields correctly mapped to the corresponding fields on the device. An appropriate mapping of the device side reminder(s) should be present on the server.

Test ID	Objective	Procedure	Expected Result
		Perform a synchronization	
6.11 Mark task as completed	Verify that task completion can be synchronized	Create a task on the device Complete title, due date, priority and access/privacy fields Perform a synchronization Set task as complete on device Perform a synchronization	The task should be synchronized to the server, with all fields correctly mapped. After second synchronization, task should be marked as complete on the server
6.12 Special Characters From Device	Verify that special characters in tasks correctly synchronize to the server.	From the device, create a task filling out all fields with special characters. Perform a synchronization From the server, modify the task. Perform a synchronization	The task should display on the server with all fields on the device correctly mapped to corresponding fields on the server. All special characters should correctly display on the server as it is displayed on the device. After modifying the task, the characters should remain correctly displayed on both the device and server and the changes made on the server should be reflected on the device
6.13 Multi-Byte Characters From Device	Verify that multi-byte characters in tasks correctly synchronize to the server.	From the device, create a task filling out all fields with multi-byte characters. Perform a synchronization From the server modify the event. Perform a synchronization	The task should display on the server with all multi-byte characters correctly displayed. After modifying the task, the multi-byte characters should remain correctly displayed on both the device and server, and the changes made on the server should be reflected on the device.
6.14 Deletion	Verify that tasks deleted device side are deleted on the device	From the device delete all tasks created in previous tests. Perform a synchronization	The deleted tasks should be deleted off of the server as well.

5. Part III — Contact Tests

5.1. Contacts

The following set of tests verifies that basic synchronization of contact information can be accomplished. In particular they attempt to verify the following:

— that Contacts can be created, modified, and deleted on the device or server and that appropriate mappings are maintained to verify that there is no corruption or loss of data.
— that special characters and multi byte characters can be correctly handled

NOTE 1 Appendix A — Special Information, has additional information on Mapping, Special Characters, and Multi Byte Characters.

NOTE 2 All tests should be performed in succession.

Test ID	Objective	Procedure	Expected Result
7.1 Create new contact with minimal fields from the server	To verify that the fields that are already supported for a device are correctly transferred from server to device upon creation and from device to server upon modification.	Create a new contact entry from the server with all fields (except for addresses, emails, telephone numbers, and URLs) Perform a synchronization Modify it from the device, and then synchronize again.	The address book entry created on the server should be reflected on the device with an appropriate mapping of fields taking advantage of what the device supports. After modifying the entry and synchronizing, the modified entry is similar on both the device and server. The mappings used to send data to the devices should be the same on data returning to the server so that the expected fields are modified.
7.2 Create new contact with minimal fields from the device	To verify that the fields that are already supported for a device are correctly transferred from device to server upon creation and from device to server upon modification.	Create a new contact entry from the device with all fields filled out (except for addresses, emails, telephone numbers, and URLs) Perform a synchronization Modify it from the server, and then synchronize again.	The address book entry created on the device should be reflected on the server with an appropriate mapping of fields taking advantage of what the server supports After modifying the entry and synchronizing, the modified entry is similar on both the device and server. The mappings used to send data to the server should be the same on data returning to the device so that the expected fields are modified.
7.3 Special Characters	Verify that special characters in contacts correctly synchronize to and from the device.	From the Server, create a contact filling out all text fields with special characters.	The contact should display on the device with all fields on the server correctly mapped to corresponding fields on the device. All special characters should correctly display on the

Test ID	Objective	Procedure	Expected Result
		Perform a synchronization From the device modify the contact. Perform a synchronization Repeat creating contact on device and modifying on the server.	device as it is displayed on the server. After modifying the contact, the characters should remain correctly displayed on both the device and server and the changes made on the device should be reflected on the server
7.4 Multi-Byte Characters	Verify that multi-byte characters in contacts correctly synchronize to and from the device.	From the server, create a contact filling out all text fields with multi-byte characters. Perform a synchronization From the device modify the contact Perform a synchronization Repeat creating contact on device and modifying on the server.	The contact should display on the device with all multi-byte characters correctly displayed. After modifying the contact, the multi-byte characters should remain correctly displayed on both the device and server, and the changes made on the device should be reflected on the server.
7.5 Delete a contact from the server	To verify that when a contact is deleted on the server, it is also deleted on the device.	From the server, delete an existing contact entry and synchronize.	The corresponding address book entry is removed from the device.
7.6 Delete a contact from the device	To verify that when a contact is deleted on the device, it is also deleted on the server.	From the device, delete an existing contact entry and synchronize	The corresponding address book entry is removed from the server.

5.2. Addresses

The following set of tests target specifically the ability to synchronize addresses. In particular they attempt to verify the following:

— that multiple addresses can be handled.
— that address formatting is correctly maintained

NOTE 1 Appendix A — Special Information, has additional information on Addresses.

NOTE 2 All tests should be performed in succession.

Test ID	Objective	Procedure	Expected Result
8.1 Create new contact with addresses	To verify that the address fields that are supported for a device are correctly transferred from server to device upon creation,	Create a contact with several addresses (home, business, etc...) from the server	The contact should display on the device and address types supported by the device should

Test ID	Objective	Procedure	Expected Result
from the server	modification and deletion.	Perform a synchronization Modify the contact from device changing one of the addresses. From the server, delete the first address property and add a new address property Perform a synchronization Modify the new address property from the server. Perform a synchronization.	be available and formatted in a way that is usable to the user. The modification made on the device should be reflected on the server but any server side formatting of the address should be maintained and other address (not supported on the device) should remain unaffected. The device side address affected by the server side changes should be correctly updated and the correct ordering should be maintained. Last modification should update the corresponding address on the device.
8.2 Create new contact with addresses from the device	To verify that the address fields that are supported for a device are correctly transferred from device to server upon creation, modification and deletion.	Create a contact with several addresses (home, business, etc...) from the device (if the device supports it). Perform a synchronization Modify the contact from server changing one of the addresses. From the device, delete the first address property and add a new address property Perform a synchronization Modify the new address property from the device. Perform a synchronization.	The contact should display on the server and address types entered on the device should be correctly mapped, available, and formatted in a way that is usable to the user on the server. The modification made on the server should be reflected on the device but any device side formatting of the address should be maintained. The device side changes should get correctly reflected server side with the ordering correctly maintained. Last modification should update the corresponding address on the server.

5.3. Phone Numbers

The following set of tests target specifically the ability to synchronize phone numbers. In particular they attempt to verify the following:

— that multiple phone numbers can be handled.
— that phone number formatting is correctly maintained

NOTE 1 Appendix A — Special Information, has additional information on Phone Numbers.

NOTE 2 All tests should be performed in succession.

Test ID	Objective	Procedure	Expected Result
9.1 Create new contact with telephone numbers from the server	To verify that the telephone fields that are supported for a device are correctly transferred from server to device upon creation, modification and deletion.	Create a contact with several phone numbers (home1, home2, business, etc...) from the server. Perform a synchronization Modify the contact from device changing one of the phone numbers. From the server, delete the first phone number and add a new phone number Perform a synchronization Modify the new phone number from the server. Perform a synchronization	The contact should display on the device and the phone numbers supported by the device should be available and formatted in a way that is usable to the user. The modification made on the device should be reflected on the server but any server side formatting of the phone number should be maintained and other phone numbers (not supported on the device) should remain unaffected. The device side phone numbers affected by the server side changes should be correctly updated and the correct ordering should be maintained. Last modification should update the corresponding phone number on the device.
9.2 Create new contact with telephone numbers from the device	To verify that the telephone fields that are supported for a device are correctly transferred from device to server upon creation, modification and deletion.	Create a contact with several phone numbers (home1, home2, business, etc...) from the device (if the device supports it). Perform a synchronization Modify the contact from server changing one of the phone numbers. From the device, delete the first phone number and add a new phone number Perform a synchronization Modify the new phone number from the device. Perform a synchronization	The contact should display on the server and phone numbers entered on the device should be correctly mapped, available, and formatted in a way that is usable to the user on the server. The modification made on the server should be reflected on the device but any device side formatting of the phone number should be maintained. The device side changes should get correctly reflected server side with the ordering correctly maintained. Last modification should update the corresponding phone number on the server.

5.4. Emails and URLs

The following set of tests target specifically the ability to synchronize emails addresses and URLs.

Test ID	Objective	Procedure	Expected Result
10.1 Create new contact with emails from the server	To verify that the email fields that are supported for a device are correctly transferred from server to device upon creation, modification and deletion.	Create a contact with several email addresses (home1, home2, business, etc...) from the server. Perform a synchronization Modify the contact from device changing one of the email addresses. From the server, delete the first email address and add a new email address Perform a synchronization Modify the new email address from the server. Perform a synchronization	The contact should display on the device and the email addresses supported by the device should be available and formatted in a way that is usable to the user. The modification made on the device should be reflected on the server but other email addresses (not supported on the device) should remain unaffected. The device side email addresses affected by the server side changes should be correctly updated and the correct ordering should be maintained. Last modification should update the corresponding email address on the device.
10.2 Create new contact with URLs/web page addresses from the server.	To verify that the URL fields that are supported for a device are correctly transferred from server to device upon creation, modification and deletion.	Create a contact with several web page URLs (work web site, home web site, etc...) from the server. Perform a synchronization Modify the contact from device changing one of the web page URLs From the server, delete the first URL and add a new URL Perform a synchronization Modify the new URL from the server. Perform a synchronization	The contact should display on the device and the URLs supported by the device should be available and formatted in a way that is usable to the user. The modification made on the device should be reflected on the server but other URLs (not supported on the device) should remain unaffected. The device side URLs affected by the server side changes should be correctly updated and the correct ordering should be maintained. Last modification should update the corresponding URL on the device.

Test ID	Objective	Procedure	Expected Result
10.3 Create new contact with emails from the device	To verify that the email fields that are supported for a device are correctly transferred from device to server upon creation, modification and deletion.	Create a contact with several email addresses (home1, home2, business, etc...) from the device. Perform a synchronization Modify the contact from the server changing one of the email addresses. From the device, delete the first email address and add a new email address Perform a synchronization Modify the new email address from the device. Perform a synchronization	The contact should display on the server and email addresses entered on the device should be correctly mapped, available, and formatted in a way that is usable to the user on the server. The modification made on the server should be reflected on the device. The device side changes should get correctly reflected server side with the ordering correctly maintained. Last modification should update the corresponding email address on the server
10.4 Create new contact with URLs/web page addresses from the device	To verify that the URL fields that are supported for a device are correctly transferred from device to server upon creation, modification and deletion.	Create a contact with several web page URLs (work web site, home web site, etc...) from the device. Perform a synchronization Modify the contact from server changing one of the web page URLs From the device, delete the first URL and add a new URL Perform a synchronization Modify the new URL from the device. Perform a synchronization	The contact should display on the server and URLs entered on the device should be correctly mapped, available, and formatted in a way that is usable to the user on the server. The modification made on the server should be reflected on the device. The device side changes should get correctly reflected server side with the ordering correctly maintained. Last modification should update the corresponding URL on the server

Appendix A
(normative)
Supporting Information

A.1. Truncation

In an ideal scenario all servers and all mobile devices would store their calendar data as iCalendar and they would support all attributes that iCalendar supports and would support attribute values of any length. Given the inherit restrictions of memory and storage prevalent with most mobile devices this utopian vision however is a rarity.

More then likely (although this is changing as mobile devices become more and more sophisticated) the device may only be able to expose a subset of the information possible from the desktop application and certain attribute values may need to be truncated. This is acceptable as long as an adequately accurate representation of the event is still available to the calendar user and if such restrictions do not end up adversely effecting the original server-side representation.

EXAMPLE 1

A user creates a meeting with a start time, duration, title, and some details. The calendar application on the device the user uses can only display start time, duration, and title. If the user modifies the start time from the device and then synchronizes the details should not be lost on the server-side representation of the event.

EXAMPLE 2

A user creates a meeting with a start time, duration, title, and a long description of the agenda. The calendar application on the device the user uses can display the start time, the duration, the title, and some details but it is limited to an amount of characters less then the length of the agenda so the entire agenda cannot be seen by the user from the device. If the user modifies the start time from the device and then synchronizes the agenda should not get truncated server-side.

A.2. Mapping

Most iCalendar attribute values are either a string or a date and time and thus there is no real issues regarding mapping for such values. For access levels and priorities however the values are based on enumerated lists. The iCalendar specification does describe appropriate values for these attributes but often desktop calendar client solutions support a much richer set of values and often mobile devices will support a lesser set (if at all).

The case where the mobile device simply doesn't support such concepts is fairly easy to support. Simply don't display them and make sure changes on the device do not affect them server-side (as described in the section above about truncation).

It becomes much more complicated when the mobile device does support these attributes but supports a different set of accepted values. In such cases an appropriate mapping needs to occur. This is acceptable as long as an adequately accurate representation of the event is still available to the calendar user and if such mappings do not end up adversely effecting the original server-side representation.

EXAMPLE 1

A user creates a meeting with normal access (which is different then public for the system they use). The calendar application on the device the user uses can only display public or private so the event is displayed as public. If the user modifies the start time from the device and

then synchronizes the access level should not now be changed to Public on the server-side representation of the event. It is important that it remains normal.

EXAMPLE 2

A user creates a public meeting by mistake. Later the user uses the calendar application on the device to switch it to private. The access level should change on the server as the user explicitly modified it.

EXAMPLE 3

A user creates a meeting of highest priority (which is different than high for the system they use). The calendar application on the device the user uses can only display low, medium, or high so the event is displayed as high. If the user modifies the start time from the device and then synchronizes the priority level should not now be changed to high on the server-side representation of the event. It is important that it remains highest. This issue affects events and tasks.

EXAMPLE 4

A user creates a task in their mobile device calendar with a priority of 'Normal' (iCalendar created by device uses a `PRIORITY` value of 2). The user sends this task via Bluetooth to their desktop and imports the task to their desktop Calendar application. The desktop calendar application displays the task as 'Medium'.

A.3. Reminders

Reminders are interesting in that the reminder you may want your desktop calendar client to give you concerning an event is not necessarily the reminder you'd like your mobile calendar client to provide. You may want your desktop application to provide you with a popup reminder 15 minutes before the start of the event and you may want your device to beep and vibrate when the event is scheduled to actually start. Since they are both the same and yet different the way in which they get synchronized can sometimes be of interest.

If a user creates events using their desktop client they certainly don't want to have to manually create reminders on the events that get sent to their mobile device however if they edit when they want a desktop reminder or choose to remove a desktop reminder does this really correlate in any way to the mobile alert?

As a general rule it makes sense that any new event with a reminder being sent from either side (device or server) should result in an appropriate default reminder being setup on the other side.

EXAMPLE 1

A user's desktop calendar client by default sets up events with 15-minute popup reminders. The calendar application on the device the user uses by default sets up events with reminders to beep when the meeting is due to start. Events created on either side by the user result in the same type of expected reminder.

EXAMPLE 2

A user has a meeting in an hour with a 15-minute popup reminder. Because the user is currently giving a web conference presentation she does not want the reminder popping up while she is presenting so she removes the reminder. She still wants to be reminded of her next call however and is relying on the fact that her mobile phone will beep when it is time.

BOF Topic: *If a reminder is removed on the device (or vice versa) should the reminder be removed from the event on the server?*

:2007

Task reminders are also interesting since most desktop calendar applications will allow you to setup a reminder based on when you should start working on the task or based on the due date where is most mobile devices either don't support Task reminders or only support one reminder. Implementations should attempt map task reminders accordingly.

A.4. Special Characters

The following illustrates some of the special characters that can be used to test when dealing with test cases for special characters:

Accented Characters	çèéêëìíîïàáâãäåõöôóòùúûüñý
Euro Sign	€€€€€
Numbers	½¼
Omega Characters	ƒ†‡¶£¤¥§©
Special Characters	~`!@#$%^&()_+=-{}|\][:'";<>?/.,
New Line character	(↵)

A.5. Multi-Byte Characters

Characters from East Asian languages such as Chinese, Japanese and Korean (CJK) cannot be represented using 8-bit text like many European languages. CJK character sets typically use multi-byte variable-length encodings such as UTF-8.

The following illustrates some of the multi-byte characters that can be used to test when dealing with test cases for multi-byte characters:

Chinese Characters:

— U+5317 U+4EAC Beijing

Japanese Characters:

— U+3042 Hiragana Letter A
— U+3044 Hiragana Letter I
— U+3046 Hiragana Letter U
— U+3048 Hiragana Letter E
— U+304A Hiragana Letter O

Korean Characters:

— U+1100 Latin characters k/g
— U+1105 Latin characters r/l

Implementations should use UTF-8 encoding form as a default character set as recommended by iCalendar to guarantee correct display of multi-byte characters (such as CJK languages). However, mobile devices may wish to use specific character sets for the market the device is being sold within (e.g. Many Japanese phones use Shift-JIS exclusively). Regardless of the implementation however users never want to see "% ($%^^##@???"

A.6. All Day Events

An 'all-day event' is a scheduled activity covering an entire day or a block of days. These types of events are also known as 'day events' or 'memo' events.

A common use of all-day events is to represent a vacation and some products support a special type of all-day event just for holidays. Anniversary events, an annually repeating all-day event, are also very common provided in Calendar Scheduling products.

A.6.1. Representation of All-Day Events in vCalendar/iCalendar

The vCalendar 1.0 specification, which is widely adopted on mobile devices, does not describe a standard representation of all day events.

The OMA data synchronisation group has published a Minimum Interoperability Profile vObject which aims to provide guidance on how to interpret some ambiguous areas of the vCalendar 1.0 specification.

It recommends that all-day events should be represented using the same date for `DTSTART` and `DTEND`, with a time of 00:00:00 for `DTSTART` and 24:00:00 for `DTEND`. 24 hour events that begin at midnight should be represented using `DTSTART` and `DTEND` time of 00:00:00 and `DTEND` set to one day after the event. However, a time value of 24:00:00 is illegal syntax in iCalendar; the valid range for the hour value is 0 through 23.

The iCalendar specification states that the `DTEND` property is exclusive (i.e. the date specified in the `DTEND` is not included in the event duration). An event that lasts all day on June 19th 2007 should be represented as:

```
DTSTART;VALUE=DATE:20070619
DTSTART;VALUE=DATE:20070620
```

Most major Calendar implementations follow this guidance. For example, Microsoft Outlook:

```
BEGIN:VCALENDAR
PRODID:-//Microsoft Corporation//Outlook 9.0 MIMEDIR//EN
VERSION:2.0
METHOD:PUBLISH
BEGIN:VEVENT
ORGANIZER:MAILTO:< omitted >
DTSTART;VALUE=DATE:20070402
DTEND;VALUE=DATE:20070403
LOCATION:London office
TRANSP:OPAQUE
SEQUENCE:0
UID:040000008200E00074C5B7101A82E00800000000E0F088CC1875C7010000000000
    0000001000000008863E35F4A64624397C17A75BF6F4C4A
DTSTAMP:20070402T101937Z
SUMMARY:MS all day event\, time
PRIORITY:5
CLASS:PUBLIC
END:VEVENT
END:VCALENDAR
```

For example, Google™:

```
BEGIN:VCALENDAR
PRODID:-//Google Inc//Google Calendar 70.9054//EN
VERSION:2.0
CALSCALE:GREGORIAN
METHOD:REQUEST
BEGIN:VEVENT
DTSTART;VALUE=DATE:20070404
DTEND;VALUE=DATE:20070405
< additional properties omitted for readability >
STATUS:CONFIRMED
SUMMARY:Birthday
```



```
TRANSP:TRANSPARENT
END:VEVENT
END:VCALENDAR
```

A.6.2. Do All Day Events have a duration?

The biggest issue with day events has always been do they have duration and how should they be presented using iCalendar? iCalendar certainly is robust enough to support the concept but it does not explicitly dictate how to represent a day event.

Christmas is the 25th of December regardless of were you are in the world. It can be represented as an all day event on the 25 th. If a user is attending a conference in Hong Kong from Monday to Friday then for the user's boss in North America the user is actually starting to attend some time on Sunday. Which is right?

The reality is that some systems support one way or the other or in some cases both then add into the mix a mobile device that itself does its own thing and you have a dreadful mess.

BOF Topic: *What form of iCALENDAR should be passed back and forth to ensure these concepts are correctly handled?*

A.6.3. Multi-Day Day Events

Some servers support multi-day All Day events and treat them as entries that spans across a few days. For devices that support multi-day All day events, the entry will be displayed in the same manner as it is on the server.

Unfortunately many devices that support All Day Events do not support them across multiple days. For such cases the server has to do something appropriate. Refusing to send such day events is one solution or perhaps the entry is displayed on the device as a one day event, but the string "~(x)" is appended to the title field, where x is the number of days that entry spans across. For example: If the device does not support multi-day events, then creating one on the server with the title "Multi-Day Event" that spans across 5 days will be displayed on the device as a one day Day Event with the title "Multi-Day Event ~(5)". When testing multi-day day events it is important to understand what the device can support and what the server does to compensate.

A.7. Repeating Events and Recurrence Rules

The biggest problem in the area of repeating meetings is devices that claim they can support repeating meetings but which in reality can only support a very small subset of the repeating capabilities that iCalendar provides for.

For devices that simply don't claim any sort of repeating support server-side implementations should expand all repeating events and send simple single instance events for all instances within the synchronization range. The fact that they are repeating should be maintained server-side and any modification done on the device side should not affect this. This means that users can't deal with such entries as being repeating on the device but they at least always known when an instance is occurring.

It becomes much more complicated when the mobile device does claim support for repeating entries but does not fully support the concept. Often devices will support deleted exceptions but not modified exceptions or only support a small subset of recurrence rules. Whenever a server cannot reliably know what sort of support the device is claiming it should fallback to the method described above for devices with no support

Ideally, there should be a minimum level of support that mobile devices must have to claim they support repeating entries.

BOF Topic: *What is this minimum level of support?*

A server should fallback to splitting up instances for anything, which falls outside of this minimum implementation.

A.8. Scheduling

Ideally any mobile calendar solution should allow a mobile user to perform any sort of operation on their mobile device that they would expect to be able to do from their desktop. Most mobile calendar clients however are still very PIM centric. Basic mobile scheduling capabilities are beginning to emerge however. At a minimum mobile devices should be able to display information about who the attendees of a meeting are, should allow users to accept or decline invitations from their mobile device, and should allow users to initiate invitations for their mobile device.

A.9. Time Zones and Daylight Savings

In order to correctly display an event and interoperate with major calendar applications, it is necessary for mobile calendar solutions to provide support for time zones. Unfortunately, most mobile calendar solutions do not provide full support for time zones currently and are therefore unable to correctly display events in all situations.

The following different classes of device time zone support have been identified:

Full Time zone support

— has time zone configuration option
— stores all events as local time, UTC or local time with time zone rule (as appropriate)
— transfers recurring events in local time with a time zone definition
— supports day events independent of time zone changes (aka “floating” time)
— when the time zone is changed on the device, all events shift in the device calendar (excluding day events)
— switches to and from daylight savings do not effect the display of events

Type 1 Partial time zone support

— has time zone configuration option
— stores all events in UTC
— day events are stored as events starting at midnight with no duration
— when the time zone is changed on the device, all events shift, including day events which initially display at midnight (until the time zone was changed) the synchronization server requires the user’s default calendar server time zone to correctly synchronize day events to this device (so that they hopefully end up at midnight)

Type 2 Partial time zone support

— has time zone configuration option
— accepts UTC, but actually stores them in local time
— supports day events independent of time zone changes
— when the time zone is changed on the device, events are not shifted in the device calendar

No Timezone support

— the device may have an option for setting the time zone, but the information is not used by the device calendar application or the synchronization application
— all events are stored in local time
— day events may or may not be supported correctly
— when the time zone is changed on the device, events are not shifted in the device calendar

The timezone and daylight savings test cases described in this test suite assume that full time zone support has been correctly implemented. Other implementations are simply considered broken.

A.10. Task Completion

How should a task (to-do) be marked as completed?

iCalendar defines three properties to indicate that a task is completed. `STATUS` describes the overall status for the task, `PERCENT-COMPLETE` allows a delegate to communicate the percentage completion of the task to the organizer and `COMPLETED` indicates the date/time that the to-do was actually completed. An example of how a completed task can be expressed using these properties is shown below:

```
STATUS:COMPLETED
PERCENT-COMPLETE:100
COMPLETED:20070101T100000Z
```

Unfortunately, iCalendar does not prescribe a single way to mark a task as completed. Device-side clients may support any combination of these properties. A server must be able to send a device a completed task using the format it is expecting and be able to recognize how a device client indicates a completed task.

A.11. Contact Mappings

vCard as a data format supports an endless number of combinations. This makes it very difficult to guarantee that both sides can support each others representations. Despite its flexibility however it does not support a way to enumerate multiple properties of the same type.

A.12. Addresses

Address formats differ greatly from country to country. When an address is represented as just a text field it is easy to pass back and forth but often servers will store address information in separate fields (e.g. Street, City, Country, Postal Code) therefore when working with a device that expects things all in one label this information must be merged together. If it is then modified on the device it is important that server side the representation remains intact. Address formatting for majority of Asian countries will typically require more than one address field. Some software implementation allows for two fields (e.g. Address1 and Address2) to allow the extra information to be entered and rendered where as others provide a single Address field. It may be helpful to provide guidelines on how to handle these occurrences which are extremely common for Asian addresses.

EXAMPLE 1

The Yamasa Institute
Address 1: 1-2-1 Hanehigashi-machi
Address 2: AichiPrefecture
City: OkazakiCity
Country: JAPAN
Zip Code: 444-0832

EXAMPLE 2

Samsung Seoul (Head Office)
Address 1: SAMSUNGMainBuilding
Address 2: 250-2 ga,Taepyung-roChung-gu
City: Seoul
Country: Korea

A.13. Phone numbers

13336669999, +13336669999, 1 (333) 666-999, all of these are valid and some servers store the country code, area code, and actual number separately.

While this format generally works, there are instances where simple concatenation of country code, area code, and actual number may not work correctly. For instance, a phone number in Tokyo, Japan(e.g. 03-3580-3377) may not be dialled correctly by using +810335803377. The area code (03) must be translated to (3) so that the mobile device omits the extra (0) and dials +81335803377. These types of area codes are quite common for many Asian countries. It would be ideal if software applications can interpret these type of numbers and automatically convert the phone numbers accordingly. Also, the display of the phone numbers should be rendered according to where the end-user is. (e.g. If the user is in Japan, the number 03-3580-3377 should be used, but if the user is outside of Japan +81335803377 should be displayed).

Appendix B
(normative)
Sample iCalendar and vCard Streams

The Calendaring Scheduling Consortium plans to collect sample iCalendar and vCard contributions for each of the Calendar, Task, and Contact test cases. These will be made available from the Consortium's *CalConnect Interoperability Testing Resources* page:

http://www.calconnect.org/ioptesting.shtml

Appendix C
(normative)
Open Issues Known Omissions

There are currently no test cases defined for the following areas:

1) Calendar test cases do not include attachments
2) Day Event test cases are not considered complete. Issues such as time transparency for all-day events. For example, should all-day events block time (free/busy handling)? Is the meaning preserved between device server? A consensus within the industry needs to be reached before these tests can be considered complete.
3) Recurrence tests are not considered complete: Frequencies less than a day (secondly, minutely, hourly), various until/count/interval combinations, various `BYXXX` repeat patterns, `WKST`, limited coverage of combinations of repeat patterns, limited coverage of exception combinations and repeat modification/deletion. If iCalendar test data is supplied, verify that recurrence property parameters can be specified in any order. A consensus within the industry on what the minimal level of support should be on a mobile device is required to complete a concrete set of tests.
4) Minimal Alarms/Meeting reminders and the handling of alarms on the device are provided but a proper consensus on how to have reminders specified that make sense for the environment they are going off in needs to be reached.
5) Tests covering meetings that start, end, or cross over the shift between standard and daylight savings are not covered. Consensus on what should be the expected behaviour is needed.
6) Various scheduling operations such as delegation, meeting cancellation, and meeting reschedules are not covered. Once Scheduling capabilities are more prevalent on mobile devices additional test cases should be added.
7) Repeating Tasks or Task Assignments are not covered. When such support becomes more widespread on mobile devices additional test cases should be added.
8) Character Encoding IOP issues. For example: Shift JIS support confusion with encoding Yen symbol/backslash characters, base64/quoted printable encodings, line folding, UTF-8 support

Bibliography

[1] IETF RFC 4791, C. DABOO, B. DESRUISSEAUX and L. DUSSEAULT. *Calendaring Extensions to WebDAV (CalDAV)*. 2007. RFC Publisher. https://www.rfc-editor.org/info/rfc4791.

[2] CC/R 1102:2013, *Calendaring and Scheduling Glossary of Terms V2.2 (CD 1102)*.

[3] 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.

[4] OMA DS, Open Mobile Alliance Data Synchronization V1.1.2, Open Mobile Alliance, 10 July 2006 http://www.openmobilealliance.org/release_program/ds_v12.html

[5] CC/S 0601:2006, *Min-IOP (Minimum Interoperable Subset) Use Cases V1.1 (CD 0601)*. https://standards.calconnect.org/pubdocs/CD0601%20Min-IOP%20Use%20Cases%20V1.1.pdf.

[6] IETF RFC 3283, B. MAHONEY, G. BABICS and A. TALER. *Guide to Internet Calendaring*. 2002. RFC Publisher. https://www.rfc-editor.org/info/rfc3283.

[7] CC/Adv 0611:2006, *The Benefits of iCalendar for the Mobile Industry V1.0 (CD 0611)*. https://standards.calconnect.org/pubdocs/CD0611%20The%20Benefits%20of%20iCalendar%20for%20the%20Mobile%20Industry%20V1.0.pdf.

[8] vCalendar 1.0, vCalendar The Electronic Calendaring and Scheduling Exchange Format Version 1.0, Versit Consortium, September 18 1996 http://www.imc.org/pdi/vcal-10.txt

[9] vCard 2.1, vCard The Electronic Business Card Version 2.1, Versit Consortium, September 18 1996 http://www.imc.org/pdi/vcard-21.txt

[10] IETF RFC 2426, F. DAWSON and T. HOWES. *vCard MIME Directory Profile*. 1998. RFC Publisher. https://www.rfc-editor.org/info/rfc2426.

[11] vObject, OMA vObject Minimum Interoperability Profile Version 1.0, Open Mobile Alliance, 18 January 2005 http://www.openmobilealliance.org/release_program/vObject_v10.html

