Network Working Group
Request for Comments: 5546
Obsoletes: 2446
Updates: 5545
Category: Standards Track

C. Daboo, Ed.
Apple Inc.

December 2009

iCalendar Transport-Independent Interoperability Protocol (iTIP)

Abstract

This document specifies a protocol that uses the iCalendar object specification to provide scheduling interoperability between different calendaring systems. This is done without reference to a specific transport protocol so as to allow multiple methods of communication between systems. Subsequent documents will define profiles of this protocol that use specific, interoperable methods of communication between systems.

The iCalendar Transport-Independent Interoperability Protocol (iTIP) complements the iCalendar object specification by adding semantics for group scheduling methods commonly available in current calendaring systems. These scheduling methods permit two or more calendaring systems to perform transactions such as publishing, scheduling, rescheduling, responding to scheduling requests, negotiating changes, or canceling.

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Table of Contents

1. Introduction and Overview 5
   1.1. Formatting Conventions 5
   1.2. Related Documents 6
   1.3. Roles 6
   1.4. Methods 7
2. Interoperability Models 9
   2.1. Application Protocol 10
        2.1.1. Scheduling State 10
        2.1.2. Delegation 10
        2.1.3. Acting on Behalf of Other Calendar Users 11
        2.1.4. Component Revisions 11
        2.1.5. Message Sequencing 12
3. Application Protocol Elements 13
   3.1. Common Component Restriction Tables 15
        3.1.1. VCALENDAR 15
        3.1.2. VTIMEZONE 15
        3.1.3. VALARM 17
   3.2. Methods for VEVENT Calendar Components 17
        3.2.1. PUBLISH 18
        3.2.2. REQUEST 20
        3.2.3. REPLY 25
        3.2.4. ADD 27
        3.2.5. CANCEL 29
        3.2.6. REFRESH 31
        3.2.7. COUNTER 33
        3.2.8. DECLINECOUNTER 35
   3.3. Methods for VFREEBUSY Components 37
        3.3.1. PUBLISH 37
        3.3.2. REQUEST 40
        3.3.3. REPLY 42

3.4. Methods for VTODO Components 44
3.4.1. PUBLISH 44
3.4.2. REQUEST 46
3.4.3. REPLY 51
3.4.4. ADD 53
3.4.5. CANCEL 55
3.4.6. REFRESH 57
3.4.7. COUNTER 59
3.4.8. DECLINECOUNTER 61
3.5. Methods for VJOURNAL Components 62
3.5.1. PUBLISH 63
3.5.2. ADD 64
3.5.3. CANCEL 66
3.6. Status Replies 68
3.7. Implementation Considerations 77
3.7.1. Working With Recurrence Instances 77
3.7.2. Attendee Property Considerations 78
3.7.3. Extension Tokens 79
4. Examples 79
4.1. Published Event Examples 79
4.1.1. A Minimal Published Event 80
4.1.2. Changing a Published Event 80
4.1.3. Canceling a Published Event 81
4.1.4. A Rich Published Event 81
4.1.5. Anniversaries or Events Attached to Entire Days 83
4.2. Group Event Examples 83
4.2.1. A Group Event Request 84
4.2.2. Reply to a Group Event Request 85
4.2.3. Update an Event 85
4.2.4. Countering an Event Proposal 86
4.2.5. Delegating an Event 88
4.2.6. Delegate Accepts the Meeting 90
4.2.7. Delegate Declines the Meeting 91
4.2.8. Forwarding an Event Request 92
4.2.9. Cancel a Group Event 92
4.2.10. Removing Attendees 93
4.2.11. Replacing the Organizer 95
4.3. Busy Time Examples 96
4.3.1. Publish Busy Time 96
4.3.2. Request Busy Time 96
4.3.3. Reply to a Busy Time Request 97
4.4. Recurring Event and Time Zone Examples 98
4.4.1. A Recurring Event Spanning Time Zones 98
4.4.2. Modify a Recurring Instance 99
4.4.3. Cancel an Instance 101
4.4.4. Cancel a Recurring Event 101
4.4.5. Change All Future Instances 102
4.4.6. Add a New Instance to a Recurring Event 102

4.4.7. Add a New Series of Instances to a Recurring Event 103
4.4.8. Refreshing a Recurring Event 104
4.4.9. Counter an Instance of a Recurring Event 106
4.4.10. Error Reply to a Request 107
4.5. Group To-Do Examples 108
4.5.1. A VTODO Request 109
4.5.2. A VTODO Reply 110
4.5.3. A VTODO Request for Updated Status 110
4.5.4. A Reply: Percent-Complete 111
4.5.5. A Reply: Completed 111
4.5.6. An Updated VTODO Request 112
4.5.7. Recurring VTODOs 112
4.6. Journal Examples 113
4.7. Other Examples 114
4.7.1. Event Refresh 114
4.7.2. Bad RECURRENCE-ID 114
5. Application Protocol Fallbacks 116
5.1. Partial Implementation 116
5.1.1. Event-Related Fallbacks 117
5.1.2. Free/Busy-Related Fallbacks 119
5.1.3. To-Do-Related Fallbacks 120
5.1.4. Journal-Related Fallbacks 122
5.2. Latency Issues 123
5.2.1. Cancellation of an Unknown Calendar Component 123
5.2.2. Unexpected Reply from an Unknown Delegate 124
5.3. Sequence Number 124
6. Security Considerations 124
6.1. Security Threats 124
6.1.1. Spoofing the Organizer 124
6.1.2. Spoofing the Attendee 124
6.1.3. Unauthorized Replacement of the Organizer 125
6.1.4. Eavesdropping and Data Integrity 125
6.1.5. Flooding a Calendar 125
6.1.6. Unauthorized REFRESH Requests 125
6.2. Recommendations 125
6.2.1. Securing iTIP transactions 125
6.2.2. Implementation Controls 126
6.2.3. Access Controls and Filtering 126
6.3. Privacy Issues 126
7. IANA Considerations 127
7.1. Registration Template for REQUEST-STATUS Values 127
7.2. Additions to iCalendar METHOD Registry 127
7.3. REQUEST-STATUS Value Registry 129
8. Acknowledgments 130
9. References 131
9.1. Normative References 131
9.2. Informative References 131

Appendix A. Differences from RFC 2446 132
A.1. Changed Restrictions 132
A.2. Deprecated Features 133

1. Introduction and Overview

This document specifies how calendaring systems use iCalendar [RFC5545] objects to interoperate with other calendaring systems. In particular, it specifies how to schedule events, to-dos, or daily journal entries. It further specifies how to search for available busy time information. It does so in a general way, without specifying how communication between different systems actually takes place. Subsequent documents will specify transport bindings between systems that use this protocol.

This protocol is based on messages sent from an originator to one or more recipients. For certain types of messages, a recipient may reply in order to update their status and may also return transaction/request status information. The protocol supports the ability for the message originator to modify or cancel the original message. The protocol also supports the ability for recipients to suggest changes to the originator of a message. The elements of the protocol also define the user roles for its transactions.

This specification obsoletes RFC 2446 - a list of important changes is provided in Appendix A.

1.1. Formatting Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Calendaring and scheduling roles are referred to in quoted-strings of text with the first character of each word in upper case. For example, "Organizer" refers to a role of a "Calendar User" (CU) within the scheduling protocol.

Calendar components defined by [RFC5545] are referred to with capitalized, quoted-strings of text. All calendar components start with the letter "V". For example, "VEVENT" refers to the event calendar component, "VTODO" refers to the to-do calendar component, and "VJOURNAL" refers to the daily journal calendar component.

Scheduling methods are referred to with capitalized, quoted-strings of text. For example, "REQUEST" refers to the method for requesting a scheduling calendar component be created or modified; "REPLY" refers to the method a recipient of a request uses to update their status with the "Organizer" of the calendar component.

Properties defined by [RFC5545] are referred to with capitalized, quoted-strings of text, followed by the word "property". For example, "ATTENDEE" property refers to the iCalendar property used to convey the calendar address of a "Calendar User".

Property parameters defined by this specification are referred to with capitalized, quoted-strings of text, followed by the word "parameter". For example, "VALUE" parameter refers to the iCalendar property parameter used to override the default data type for a property value.

Enumerated values defined by this specification are referred to with capitalized text, either alone or followed by the word "value".

In tables, the quoted-string text is specified without quotes in order to minimize the table length.

1.2. Related Documents

Implementers will need to be familiar with several other specifications that, along with this one, describe the Internet calendaring and scheduling standards. The related documents are:

[RFC5545] - specifies the objects, data types, properties, and property parameters used in the protocols, along with the methods for representing and encoding them.

[iMIP] - specifies an Internet email binding for iTIP.

This specification does not attempt to repeat the concepts or definitions from these other specifications. Where possible, explicit references are made to the other specifications.

1.3. Roles

Exchanges of iCalendar objects for the purposes of group calendaring and scheduling occur between "Calendar Users" (CUs). CUs take on several roles in iTIP:

Role	Description
Organizer	The CU who initiates an exchange takes on the role of Organizer. For example, the CU who proposes a group meeting is the Organizer.
Attendee	CUs who are included in the scheduling message as possible recipients of that scheduling message. For example, the CUs asked to participate in a group meeting by the Organizer take on the role of Attendee.
Other CU	A CU that is not explicitly included in a scheduling message, i.e., not the Organizer or an Attendee.

Note that "ROLE" is also a descriptive parameter to the iCalendar "ATTENDEE" property. Its use is to convey descriptive context about an "Attendee" -- such as "chair", "required participant", or "non-required participant" -- and has nothing to do with the calendaring workflow.

1.4. Methods

The iTIP methods are listed below and their usage and semantics are defined in Section 3 of this document.

Method	Description
PUBLISH	Used to publish an iCalendar object to one or more "Calendar Users". There is no interactivity between the publisher and any other "Calendar User". An example might include a baseball team publishing its schedule to the public.
REQUEST	Used to schedule an iCalendar object with other "Calendar Users". Requests are interactive in that they require the receiver to respond using the reply methods. Meeting requests, busy-time requests, and the assignment of tasks to other "Calendar Users" are all examples. Requests are also used by the Organizer to update the status of an iCalendar object.
REPLY	A reply is used in response to a request to convey Attendee status to the Organizer. Replies are commonly used to respond to meeting and task requests.
ADD	Add one or more new instances to an existing recurring iCalendar object.
CANCEL	Cancel one or more instances of an existing iCalendar object.
REFRESH	Used by an Attendee to request the latest version of an iCalendar object.
COUNTER	Used by an Attendee to negotiate a change in an iCalendar object. Examples include the request to change a proposed event time or change the due date for a task.
DECLINECOUNTER	Used by the Organizer to decline the proposed counter proposal.

Group scheduling in iTIP is accomplished using the set of "request" and "response" methods described above. The following table shows the methods broken down by who can send them.

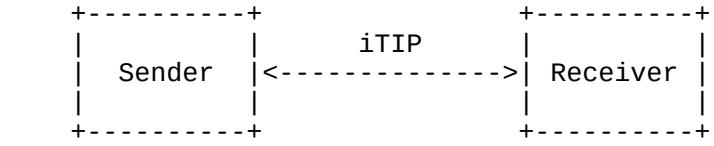
Originator	Methods
Organizer	PUBLISH, REQUEST, ADD, CANCEL, DECLINECOUNTER
Attendee	REPLY, REFRESH, COUNTER, REQUEST (only when delegating)

Note that for some calendar component types, the allowable methods are a subset of the above set. In addition, apart from "VTIMEZONE" iCalendar components, only one component type is allowed in a single iTIP message.

2. Interoperability Models

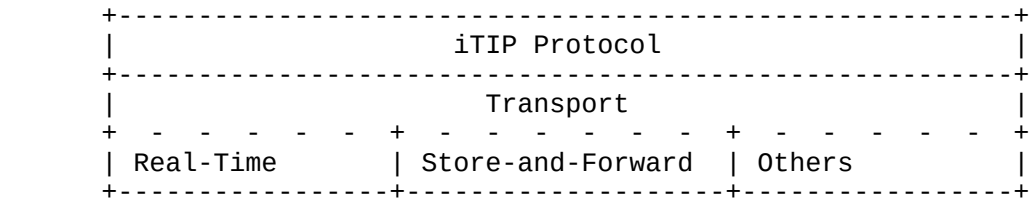
There are two distinct protocols relevant to interoperability: an "application protocol" and a "transport protocol". The application protocol defines the content of the iCalendar objects sent between sender and receiver to accomplish the scheduling transactions listed above. The transport protocol defines how the iCalendar objects are sent between the sender and receiver. This document focuses on the application protocol. Binding documents such as [iMIP] focus on the transport protocol.

The connection between sender and receiver in the diagram below refers to the application protocol. The iCalendar objects passed from the sender to the receiver are presented in Section 3, "Application Protocol Elements".



There are several variations of this diagram in which the sender and receiver take on various roles of a "Calendar User Agent" (CUA) or a "Calendar Service" (CS).

The architecture of iTIP is depicted in the diagram below. An application written to this specification may work with bindings for the store-and-forward transport, the real-time transport, or both. Also note that iTIP could be bound to other transports.



2.1. Application Protocol

In the iTIP model, an iCalendar object is created and managed by an "Organizer". The "Organizer" interacts with other CUs by sending one or more of the iTIP messages listed above. "Attendees" use the "REPLY" method to communicate their status. "Attendees" do not make direct changes to the master iCalendar object. They can, however, use the "COUNTER" method to suggest changes to the "Organizer". In any case, the "Organizer" has complete control over the master iCalendar object.

2.1.1. Scheduling State

There are two distinct states relevant to iCalendar objects used in scheduling: the overall state of the iCalendar object and the state associated with an "Attendee" in that iCalendar object.

The state of an iCalendar object is defined by the "STATUS" property and is controlled by the "Organizer." There is no default value for the "STATUS" property. The "Organizer" sets the "STATUS" property to the appropriate value for each iCalendar object.

The state of a particular "Attendee" relative to an iCalendar object used for scheduling is defined by the "PARTSTAT" parameter in the "ATTENDEE" property for each "Attendee". When an "Organizer" issues the initial iCalendar object, "Attendee" status is typically unknown. The "Organizer" specifies this by setting the "PARTSTAT" parameter to "NEEDS-ACTION". Each "Attendee" modifies their "ATTENDEE" property "PARTSTAT" parameter to an appropriate value as part of a "REPLY" message sent back to the "Organizer".

2.1.2. Delegation

Delegation is defined as the process by which an "Attendee" grants another CU (or several CUs) the right to attend on their behalf. The "Organizer" is made aware of this change because the delegating "Attendee" informs the "Organizer". These steps are detailed in the "REQUEST" method sections for the appropriate components.

2.1.3. Acting on Behalf of Other Calendar Users

In many organizations, one user will act on behalf of another to organize and/or respond to meeting requests. iTIP provides two mechanisms that support these activities.

First, the "Organizer" is treated as a special entity, separate from "Attendees". All responses from "Attendees" flow to the "Organizer", making it easy to separate a "Calendar User" organizing a meeting from "Calendar Users" attending the meeting. Additionally, iCalendar provides descriptive roles for each "Attendee". For instance, a role of "chair" may be ascribed to one or more "Attendees". The "chair" and the "Organizer" may or may not be the same "Calendar User". This maps well to scenarios where an assistant may manage meeting logistics for another individual who chairs a meeting.

Second, a "SENT-BY" parameter may be specified in either the "Organizer" or "Attendee" properties. When specified, the "SENT-BY" parameter indicates that the responding CU acted on behalf of the specified "Attendee" or "Organizer".

2.1.4. Component Revisions

The "SEQUENCE" property is used by the "Organizer" to indicate revisions to the calendar component. When the "Organizer" makes changes to one of the following properties, the sequence number MUST be incremented:

- "DTSTART"
- "DTEND"
- "DURATION"
- "DUE"
- "RRULE"
- "RDATE"
- "EXDATE"
- "STATUS"

In addition, changes made by the "Organizer" to other properties MAY also require the sequence number to be incremented. The "Organizer" CUA MUST increment the sequence number whenever it makes changes to properties in the calendar component that the "Organizer" deems will

jeopardize the validity of the participation status of the "Attendees". For example, changing the location of a meeting from one location to another distant location could effectively impact the participation status of the "Attendees".

Depending on the "METHOD", the "SEQUENCE" property MUST follow these rules in the context of iTIP:

- For the "PUBLISH" and "REQUEST" methods, the "SEQUENCE" property value is incremented according to the rules stated above.
- The "SEQUENCE" property value MUST be incremented each time the "Organizer" uses the "ADD" or "CANCEL" methods.
- The "SEQUENCE" property value MUST NOT be incremented when using "REPLY", "REFRESH", "COUNTER", "DECLINECOUNTER", or when sending a delegation "REQUEST".

In some circumstances, the "Organizer" may not have received responses to the final revision sent out. In this situation, the "Organizer" may wish to send an update "REQUEST" and set "RSVP=TRUE" for all "Attendees" so that current responses can be collected.

The value of the "SEQUENCE" property contained in a response from an "Attendee" may not always match the "Organizer's" revision. Implementations may choose to have the CUA indicate to the CU that the response is to an iCalendar object that has been revised, and allow the CU to decide whether or not to accept the response.

Whilst a change in sequence number is indicative of a significant change to a previously scheduled item, "Attendee" CUAs SHOULD NOT rely solely on a change in sequence number as a means of detecting a significant change. Instead, CUAs SHOULD compare the old and new versions of the calendar components, determine the exact nature of the changes, and make decisions -- possibly based on "Calendar User" preferences -- as to whether the user needs to be explicitly informed of the change.

2.1.5. Message Sequencing

CUAs that handle the iTIP application protocol must often correlate a component in a calendar store with a component received in the iTIP message. For example, an event may be updated with a later revision of the same event. To accomplish this, a CUA must correlate the version of the event already in its calendar store with the version sent in the iTIP message. In addition to this correlation, there are several factors that can cause iTIP messages to arrive in an unexpected order. That is, an "Organizer" could receive a reply to

an earlier revision of a component after receiving a reply to a later revision.

To maximize interoperability and to handle messages that arrive in an unexpected order, use the following rules:

1. The primary key for referencing a particular iCalendar component is the "UID" property value. To reference an instance of a recurring component, the primary key is composed of the "UID" and the "RECURRENCE-ID" properties.

2. The secondary key for referencing a component is the "SEQUENCE" property value. For components where the "UID" and "RECURRENCE-ID" property values are the same, the component with the highest numeric value for the "SEQUENCE" property obsoletes all other revisions of the component with lower values.

3. "Attendees" send "REPLY" messages to the "Organizer". For replies where the "UID" and "RECURRENCE-ID" property values are the same, the value of the "SEQUENCE" property indicates the revision of the component to which the "Attendee" is replying. The reply with the highest numeric value for the "SEQUENCE" property obsoletes all other replies with lower values.

4. In situations where the "UID", "RECURRENCE-ID", and "SEQUENCE" property values match, the "DTSTAMP" property is used as the tie-breaker. The component with the latest "DTSTAMP" overrides all others. Similarly, for "Attendee" responses where the "UID", "RECURRENCE-ID", and "SEQUENCE" property values match, the response with the latest "DTSTAMP" overrides all others.

Hence, CUAs will need to persist the following component properties in order to correctly process iTIP messages: "UID", "RECURRENCE-ID", "SEQUENCE", and "DTSTAMP". Furthermore, for each "ATTENDEE" property of a component, "Organizer" CUAs will need to persist the "SEQUENCE" and "DTSTAMP" property values associated with the "Attendee's" last response, so that any earlier responses from an "Attendee" that are received out of order (e.g., due to a delay in the transport) can be correctly discarded.

3. Application Protocol Elements

iTIP messages are "text/calendar" MIME entities that contain calendaring and scheduling information. The particular type of iCalendar message is referred to as the "method type". Each method type is identified by a "METHOD" property specified as part of the "text/calendar" content type. The table below shows various

combinations of calendar components and the method types that this specification supports.

	VEVENT	VTODO	VJOURNAL	VFREEBUSY
PUBLISH	Yes	Yes	Yes	Yes
REQUEST	Yes	Yes	No	Yes
REFRESH	Yes	Yes	No	No
CANCEL	Yes	Yes	Yes	No
ADD	Yes	Yes	Yes	No
REPLY	Yes	Yes	No	Yes
COUNTER	Yes	Yes	No	No
DECLINECOUNTER	Yes	Yes	No	No

Each method type is defined in terms of its associated components and properties. Some components and properties are required, some are optional, and others are excluded. The restrictions are expressed in this document using a simple "restriction table". The first column indicates the name of a component or property. Properties of the iCalendar object are not indented. Properties of a component are indented. The second column (the "Presence" column) indicates whether or not a component or property should be present and, if present, how many times it can occur. The third column contains comments for further clarification.

The presence column uses the following values to assert whether a property is required or optional, and the number of times it may appear in the iCalendar object.

Presence Value	Description
1	One instance MUST be present.
1+	At least one instance MUST be present.
0	Instances of this property MUST NOT be present.
0+	Multiple instances MAY be present.
0 or 1	Up to 1 instance of this property MAY be present.

The tables also call out "IANA-PROPERTY", "X-PROPERTY", "IANA-COMPONENT", and "X-COMPONENT" to show where registered and experimental property and component extensions can appear. The tables do not lay out the restrictions of property parameters. Those restrictions are defined in [RFC5545].

3.1. Common Component Restriction Tables

3.1.1. VCALENDAR

The restriction table below applies to properties of the iCalendar object. That is, the properties at the outermost scope.

Constraints for Properties in a VCALENDAR Component		
Component/Property	**Presence**	**Comment**
CALSCALE	0 or 1	
PRODID	1	
VERSION	1	Value MUST be 2.0.
IANA-PROPERTY	0+	
X-PROPERTY	0+	

3.1.2. VTIMEZONE

"VTIMEZONE" components may be referred to by other components via a "TZID" parameter on a "DATETIME" value type. The property restrictions in the table below apply to any "VTIMEZONE" component in an iTIP message.

Constraints for VTIMEZONE Components

Component/Property	Presence	Comment
VTIMEZONE	0+	MUST be present if any date/time refers to timezone.
DAYLIGHT	0+	MUST be one or more of either STANDARD or DAYLIGHT.
COMMENT	0+	
DTSTART	1	MUST be local time format.
RDATE	0+	
RRULE	0 or 1	
TZNAME	0+	
TZOFFSETFROM	1	
TZOFFSETTO	1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
LAST-MODIFIED	0 or 1	
STANDARD	0+	MUST be one or more of either STANDARD or DAYLIGHT.
COMMENT	0+	
DTSTART	1	MUST be local time format.
RDATE	0+	If present, RRULE MUST NOT be present.
RRULE	0 or 1	If present, RDATE MUST NOT be present.
TZNAME	0+	
TZOFFSETFROM	1	
TZOFFSETTO	1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
TZID	1	
TZURL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	

3.1.3. VALARM

The property restrictions in the table below apply to any "VALARM" component in an iTIP message.

Constraints for VALARM Components

Component/Property	Presence	Comment
VALARM	0+	
ACTION	1	
ATTACH	0+	
ATTENDEE	0+	
DESCRIPTION	0 or 1	
DURATION	0 or 1	If present, REPEAT MUST be present.
REPEAT	0 or 1	If present, DURATION MUST be present.
SUMMARY	0 or 1	
TRIGGER	1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	

3.2. Methods for VEVENT Calendar Components

This section defines the property set restrictions for the method types that are applicable to the "VEVENT" calendar component. Each method is defined using a table that clarifies the property constraints that define the particular method.

The following summarizes the methods that are defined for the "VEVENT" calendar component.

Method	Description
PUBLISH	Post notification of an event. Used primarily as a method of advertising the existence of an event.
REQUEST	Make a request for an event. This is an explicit invitation to one or more Attendees. Event requests are also used to update or change an existing event. Clients that cannot handle REQUEST MAY degrade the event to view it as a PUBLISH.
REPLY	Reply to an event request. Clients may set their status (PARTSTAT) to ACCEPTED, DECLINED, TENTATIVE, or DELEGATED.
ADD	Add one or more instances to an existing event.
CANCEL	Cancel one or more instances of an existing event.
REFRESH	A request is sent to an Organizer by an Attendee asking for the latest version of an event to be resent to the requester.
COUNTER	Counter a REQUEST with an alternative proposal. Sent by an Attendee to the Organizer.
DECLINECOUNTER	Decline a counter proposal. Sent to an Attendee by the Organizer.

3.2.1. PUBLISH

The "PUBLISH" method in a "VEVENT" calendar component is an unsolicited posting of an iCalendar object. Any CU may add published components to their calendar. The "Organizer" MUST be present in a published iCalendar component. "Attendees" MUST NOT be present. Its expected usage is for encapsulating an arbitrary event as an iCalendar object. The "Organizer" may subsequently update (with another "PUBLISH" method), add instances to (with an "ADD" method), or cancel (with a "CANCEL" method) a previously published "VEVENT" calendar component.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:PUBLISH of a VEVENT

Component/Property	Presence	Comment
METHOD	1	MUST equal PUBLISH.
VEVENT	1+	
DTSTAMP	1	
DTSTART	1	
ORGANIZER	1	
SUMMARY	1	Can be null.
UID	1	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
SEQUENCE	0 or 1	MUST be present if value is greater than 0; MAY be present if 0.
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0 or 1	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DTEND MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PRIORITY	0 or 1	
RDATE	0+	
RELATED-TO	0+	
RESOURCES	0+	
RRULE	0 or 1	
STATUS	0 or 1	MAY be one of TENTATIVE/CONFIRMED/CANCELLED.
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	

ATTENDEE	0	
REQUEST-STATUS	0	
VALARM	0+	
VFREEBUSY	0	
VJOURNAL	0	
VTODO	0	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	

3.2.2. REQUEST

The "REQUEST" method in a "VEVENT" component provides the following scheduling functions:

- Invite "Attendees" to an event.
- Reschedule an existing event.
- Response to a "REFRESH" request.
- Update the details of an existing event, without rescheduling it.
- Update the status of "Attendees" of an existing event, without rescheduling it.
- Reconfirm an existing event, without rescheduling it.
- Forward a "VEVENT" to another uninvited CU.
- For an existing "VEVENT" calendar component, delegate the role of "Attendee" to another CU.
- For an existing "VEVENT" calendar component, change the role of "Organizer" to another CU.

The "Organizer" originates the "REQUEST". The recipients of the "REQUEST" method are the CUs invited to the event, the "Attendees". "Attendees" use the "REPLY" method to convey attendance status to the "Organizer".

The "UID" and "SEQUENCE" properties are used to distinguish the various uses of the "REQUEST" method. If the "UID" property value in the "REQUEST" is not found on the recipient's calendar, then the "REQUEST" is for a new "VEVENT" calendar component. If the "UID" property value is found on the recipient's calendar, then the "REQUEST" is for a rescheduling, an update, or a reconfirmation of the "VEVENT" calendar component.

For the "REQUEST" method, multiple "VEVENT" components in a single iCalendar object are only permitted for components with the same "UID" property. That is, a series of recurring events may have instance-specific information. In this case, multiple "VEVENT" components are needed to express the entire series.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:REQUEST of a VEVENT

Component/Property	Presence	Comment
METHOD	1	MUST be REQUEST.
VEVENT	1+	All components MUST have the same UID.
ATTENDEE	1+	
DTSTAMP	1	
DTSTART	1	
ORGANIZER	1	
SEQUENCE	0 or 1	MUST be present if value is greater than 0; MAY be present if 0.
SUMMARY	1	Can be null.
UID	1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DTEND	0 or 1	If present, DURATION MUST NOT be present.

DURATION	0 or 1	If present, DTEND MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PRIORITY	0 or 1	
RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
REQUEST-STATUS	0	
RESOURCES	0+	
RRULE	0 or 1	
STATUS	0 or 1	MAY be one of TENTATIVE/CONFIRMED.
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
VALARM	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VFREEBUSY	0	
VJOURNAL	0	
VTODO	0	

3.2.2.1. Rescheduling an Event

The "REQUEST" method may be used to reschedule an event. A rescheduled event involves a change to the existing event in terms of its time or recurrence intervals and possibly the location or description. If the recipient CUA of a "REQUEST" method finds that the "UID" property value already exists on the calendar but that the "SEQUENCE" (or "DTSTAMP") property value in the "REQUEST" method is greater than the value for the existing event, then the "REQUEST" method describes a rescheduling of the event.

3.2.2.2. Updating or Reconfirmation of an Event

The "REQUEST" method may be used to update or reconfirm an event. An update to an existing event does not involve changes to the time or recurrence intervals, and might not involve a change to the location or description for the event. If the recipient CUA of a "REQUEST" method finds that the "UID" property value already exists on the calendar and that the "SEQUENCE" property value in the "REQUEST" is the same as the value for the existing event, then the "REQUEST" method describes an update of the event details, but not a rescheduling of the event.

The update "REQUEST" method is the appropriate response to a "REFRESH" method sent from an "Attendee" to the "Organizer" of an event.

The "Organizer" of an event may also send unsolicited "REQUEST" methods. The unsolicited "REQUEST" methods may be used to update the details of the event without rescheduling it, to update the "PARTSTAT" parameter of "Attendees", or to reconfirm the event.

3.2.2.3. Delegating an Event to Another CU

Some calendar and scheduling systems allow "Attendees" to delegate their presence at an event to another "Calendar User". iTIP supports this concept using the following workflow. Any "Attendee" may delegate their right to participate in a calendar "VEVENT" to another CU. The implication is that the delegate participates in lieu of the original "Attendee", NOT in addition to the "Attendee". The delegator MUST notify the "Organizer" of this action using the steps outlined below. Implementations may support or restrict delegation as they see fit. For instance, some implementations may restrict a delegate from delegating a "REQUEST" to another CU.

The "Delegator" of an event forwards the existing "REQUEST" to the "Delegate". The "REQUEST" method MUST include an "ATTENDEE" property with the calendar address of the "Delegate". The "Delegator" MUST also send a "REPLY" method to the "Organizer" with the "Delegator's" "ATTENDEE" property "PARTSTAT" parameter value set to "DELEGATED". In addition, the "DELEGATED-TO" parameter MUST be included with the calendar address of the "Delegate". Also, a new "ATTENDEE" property for the "Delegate" MUST be included and must specify the calendar user address set in the "DELEGATED-TO" parameter, as above.

In response to the request, the "Delegate" MUST send a "REPLY" method to the "Organizer", and optionally to the "Delegator". The "REPLY" method SHOULD include the "ATTENDEE" property with the "DELEGATED-FROM" parameter value of the "Delegator's" calendar address.

The "Delegator" may continue to receive updates to the event even though they will not be attending. This is accomplished by the "Delegator" setting their "role" attribute to "NON-PARTICIPANT" in the "REPLY" to the "Organizer".

3.2.2.4. Changing the Organizer

The situation may arise where the "Organizer" of a "VEVENT" is no longer able to perform the "Organizer" role and abdicates without passing on the "Organizer" role to someone else. When this occurs, the "Attendees" of the "VEVENT" may use out-of-band mechanisms to communicate the situation and agree upon a new "Organizer". The new "Organizer" should then send out a new "REQUEST" with a modified version of the "VEVENT" in which the "SEQUENCE" number has been incremented and the "ORGANIZER" property has been changed to the new "Organizer".

3.2.2.5. Sending on Behalf of the Organizer

There are a number of scenarios that support the need for a "Calendar User" to act on behalf of the "Organizer" without explicit role changing. This might be the case if the CU designated as "Organizer" is sick or unable to perform duties associated with that function. In these cases, iTIP supports the notion of one CU acting on behalf of another. Using the "SENT-BY" parameter, a "Calendar User" could send an updated "VEVENT" "REQUEST". In the case where one CU sends on behalf of another CU, the "Attendee" responses are still directed back towards the CU designated as "Organizer".

3.2.2.6. Forwarding to an Uninvited CU

An "Attendee" invited to a "VEVENT" calendar component may send the "VEVENT" calendar component to another new CU not previously associated with the "VEVENT" calendar component. The current "Attendee" invited to the "VEVENT" calendar component does this by forwarding the original "REQUEST" method to the new CU. The new CU can send a "REPLY" to the "Organizer" of the "VEVENT" calendar component. The reply contains an "ATTENDEE" property for the new CU.

The "Organizer" ultimately decides whether or not the new CU becomes part of the event and is not obligated to do anything with a "REPLY" from a new (uninvited) CU. If the "Organizer" does not want the new CU to be part of the event, the new "ATTENDEE" property is not added to the "VEVENT" calendar component. The "Organizer" MAY send the CU a "CANCEL" message to indicate that they will not be added to the event. If the "Organizer" decides to add the new CU, the new "ATTENDEE" property is added to the "VEVENT" calendar component. Furthermore, the "Organizer" is free to change any "ATTENDEE"

property parameter from the values supplied by the new CU to something the "Organizer" considers appropriate. The "Organizer" SHOULD send the new CU a "REQUEST" message to inform them that they have been added.

When forwarding a "REQUEST" to another CU, the forwarding "Attendee" MUST NOT make changes to the original message.

3.2.2.7. Updating Attendee Status

The "Organizer" of an event may also request updated status from one or more "Attendees". The "Organizer" sends a "REQUEST" method to the "Attendee" and sets the "ATTENDEE;RSVP=TRUE" property parameter. The "SEQUENCE" property for the event is not changed from its previous value. A recipient will determine that the only change in the "REQUEST" is that their "RSVP" property parameter indicates a request for updated status. The recipient SHOULD respond with a "REPLY" method indicating their current status with respect to the "REQUEST".

3.2.3. REPLY

The "REPLY" method in a "VEVENT" calendar component is used to respond (e.g., accept or decline) to a "REQUEST" or to reply to a delegation "REQUEST". When used to provide a delegation response, the "Delegator" SHOULD include the calendar address of the "Delegate" on the "DELEGATED-TO" property parameter of the "Delegator's" "ATTENDEE" property. The "Delegate" SHOULD include the calendar address of the "Delegator" on the "DELEGATED-FROM" property parameter of the "Delegate's" "ATTENDEE" property.

The "REPLY" method is also used when processing of a "REQUEST" fails. Depending on the value of the "REQUEST-STATUS" property, no scheduling action may have been performed.

The "Organizer" of an event may receive the "REPLY" method from a CU not in the original "REQUEST". For example, a "REPLY" may be received from a "Delegate" to an event. In addition, the "REPLY" method may be received from an unknown CU (a "Party Crasher"). This uninvited "Attendee" may be accepted, or the "Organizer" may cancel the event for the uninvited "Attendee" by sending a "CANCEL" method to the uninvited "Attendee".

An "Attendee" MAY include a message to the "Organizer" using the "COMMENT" property. For example, if the user indicates tentative acceptance and wants to let the "Organizer" know why, the reason can be expressed in the "COMMENT" property value.

The "Organizer" may also receive a "REPLY" from one CU on behalf of another. Like the scenario enumerated above for the "Organizer", "Attendees" may have another CU respond on their behalf. This is done using the "SENT-BY" parameter.

The optional properties listed in the table below (those listed as "0+" or "0 or 1") MUST NOT be changed from those of the original request. If property changes are desired, the "COUNTER" message must be used.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:REPLY of a VEVENT

Component/Property	Presence	Comment
METHOD	1	MUST be REPLY.
VEVENT	1+	All components MUST have the same UID.
ATTENDEE	1	MUST be the address of the Attendee replying.
DTSTAMP	1	
ORGANIZER	1	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
UID	1	MUST be the UID of the original REQUEST.
SEQUENCE	0 or 1	If non-zero, MUST be the sequence number of the original REQUEST. MAY be present if 0.
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DTSTART	0 or 1	

DURATION	0 or 1	If present, DTEND MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PRIORITY	0 or 1	
RDATE	0+	
RELATED-TO	0+	
RESOURCES	0+	
REQUEST-STATUS	0+	
RRULE	0 or 1	
STATUS	0 or 1	
SUMMARY	0 or 1	
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
VALARM	0	
VTIMEZONE	0 or 1	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VFREEBUSY	0	
VJOURNAL	0	
VTODO	0	

3.2.4. ADD

The "ADD" method allows the "Organizer" to add one or more new instances to an existing "VEVENT" using a single iTIP message without having to send the entire "VEVENT" with all the existing instance data, as it would have to do if the "REQUEST" method were used.

The "UID" must be that of the existing event. If the "UID" property value in the "ADD" is not found on the recipient's calendar, then the recipient SHOULD send a "REFRESH" to the "Organizer" in order to be updated with the latest version of the "VEVENT". If an "Attendee" implementation does not support the "ADD" method, it should respond with a "REQUEST-STATUS" value of 3.14 and ask for a "REFRESH".

When handling an "ADD" message, the "Attendee" treats each component in the "ADD" message as if it were referenced via an "RDATE" in the main component.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:ADD of a VEVENT

Component/Property	Presence	Comment
METHOD	1	MUST be ADD.
VEVENT	1	
DTSTAMP	1	
DTSTART	1	
ORGANIZER	1	
SEQUENCE	1	MUST be greater than 0.
SUMMARY	1	Can be null.
UID	1	MUST match that of the original event.
ATTACH	0+	
ATTENDEE	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DTEND MUST NOT be present.
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PRIORITY	0 or 1	
RELATED-TO	0+	
RESOURCES	0+	
STATUS	0 or 1	MAY be one of TENTATIVE/CONFIRMED.
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	

EXDATE	0	
RECURRENCE-ID	0	
REQUEST-STATUS	0	
RDATE	0	
RRULE	0	
VALARM	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VFREEBUSY	0	
VTODO	0	
VJOURNAL	0	

3.2.5. CANCEL

The "CANCEL" method in a "VEVENT" calendar component is used to send a cancellation notice of an existing event request to the affected "Attendees". The message is sent by the "Organizer" of the event. For a recurring event, either the whole event or instances of an event may be cancelled. To cancel the complete range of a recurring event, the "UID" property value for the event MUST be specified and a "RECURRENCE-ID" MUST NOT be specified in the "CANCEL" method. In order to cancel an individual instance of the event, the "RECURRENCE-ID" property value for the event MUST be specified in the "CANCEL" method.

There are two options for canceling a sequence of instances of a recurring "VEVENT" calendar component:

a. The "RECURRENCE-ID" property for an instance in the sequence MUST be specified with the "RANGE" property parameter value of "THISANDFUTURE" to indicate cancellation of the specified "VEVENT" calendar component and all instances after.

b. Individual recurrence instances may be cancelled by specifying multiple "VEVENT" components each with a "RECURRENCE-ID" property corresponding to one of the instances to be cancelled.

The "Organizer" MUST send a "CANCEL" message to each "Attendee" affected by the cancellation. This can be done using a single "CANCEL" message for all "Attendees" or by using multiple messages with different subsets of the affected "Attendees" in each.

When a "VEVENT" is cancelled, the "SEQUENCE" property value MUST be incremented as described in Section 2.1.4.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:CANCEL of a VEVENT

Component/Property	Presence	Comment
METHOD	1	MUST be CANCEL.
VEVENT	1+	All must have the same UID.
ATTENDEE	0+	MUST include some or all Attendees being removed from the event. MUST include some or all Attendees if the entire event is cancelled.
DTSTAMP	1	
ORGANIZER	1	
SEQUENCE	1	
UID	1	MUST be the UID of the original REQUEST.
COMMENT	0+	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DTSTART	0 or 1	
DURATION	0 or 1	If present, DTEND MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PRIORITY	0 or 1	

RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
RESOURCES	0+	
RRULE	0 or 1	
STATUS	0 or 1	MUST be set to CANCELLED to cancel the entire event. If uninviting specific Attendees, then MUST NOT be included.
SUMMARY	0 or 1	
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
REQUEST-STATUS	0	
VALARM	0	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VTODO	0	
VJOURNAL	0	
VFREEBUSY	0	

3.2.6. REFRESH

The "REFRESH" method in a "VEVENT" calendar component is used by "Attendees" of an existing event to request an updated description from the event "Organizer". The "REFRESH" method must specify the "UID" property of the event to update. A recurrence instance of an event may be requested by specifying the "RECURRENCE-ID" property corresponding to the associated event. The "Organizer" responds with the latest description and version of the event.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:REFRESH of a VEVENT

Component/Property	Presence	Comment
METHOD	1	MUST be REFRESH.
VEVENT	1	
ATTENDEE	1	MUST be the address of requester.
DTSTAMP	1	
ORGANIZER	1	
UID	1	MUST be the UID associated with original REQUEST.
COMMENT	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
IANA-PROPERTY	0+	
X-PROPERTY	0+	
ATTACH	0	
CATEGORIES	0	
CLASS	0	
CONTACT	0	
CREATED	0	
DESCRIPTION	0	
DTEND	0	
DTSTART	0	
DURATION	0	
EXDATE	0	
GEO	0	
LAST-MODIFIED	0	
LOCATION	0	
PRIORITY	0	
RDATE	0	
RELATED-TO	0	
REQUEST-STATUS	0	
RESOURCES	0	
RRULE	0	
SEQUENCE	0	
STATUS	0	
SUMMARY	0	
TRANSP	0	
URL	0	

VALARM	0	
VTIMEZONE	0+	
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VTODO	0	
VJOURNAL	0	
VFREEBUSY	0	

3.2.7. COUNTER

The "COUNTER" method for a "VEVENT" calendar component is used by an "Attendee" of an existing event to submit to the "Organizer" a counter proposal to the event. The "Attendee" sends this message to the "Organizer" of the event.

The counter proposal is an iCalendar object consisting of a "VEVENT" calendar component that provides the complete description of the alternate event.

The "Organizer" rejects the counter proposal by sending the "Attendee" a "DECLINECOUNTER" method. The "Organizer" accepts the counter proposal by rescheduling the event as described in Section 3.2.2.1, "Rescheduling an Event". The "Organizer's" CUA SHOULD send a "REQUEST" message to all "Attendees" affected by any change triggered by an accepted "COUNTER".

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:COUNTER of a VEVENT

Component/Property	Presence	Comment
METHOD	1	MUST be COUNTER.
VEVENT	1	
DTSTAMP	1	
DTSTART	1	

ORGANIZER	1	MUST be the Organizer of the original event.
SEQUENCE	1	MUST echo the original SEQUENCE number. MUST be present if non-zero. MAY be present if zero.
SUMMARY	1	Can be null.
UID	1	MUST be the UID associated with the REQUEST being countered.
ATTACH	0+	
ATTENDEE	0+	Can also be used to propose other Attendees.
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTEND	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DTEND MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PRIORITY	0 or 1	
RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
REQUEST-STATUS	0+	
RESOURCES	0+	
RRULE	0 or 1	
STATUS	0 or 1	Value must be one of CONFIRMED/TENATIVE/CANCELLED.
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
VALARM	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.

IANA-COMPONENT	0+	
X-COMPONENT	0+	
VTODO	0	
VJOURNAL	0	
VFREEBUSY	0	

3.2.8. DECLINECOUNTER

The "DECLINECOUNTER" method in a "VEVENT" calendar component is used by the "Organizer" of an event to reject a counter proposal submitted by an "Attendee". The "Organizer" must send the "DECLINECOUNTER" message to the "Attendee" that sent the "COUNTER" method to the "Organizer".

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:DECLINECOUNTER of a VEVENT

Component/Property	Presence	Comment
METHOD	1	MUST be DECLINECOUNTER.
VEVENT	1+	All components MUST have the same UID.
ATTENDEE	1+	MUST for all Attendees.
DTSTAMP	1	
ORGANIZER	1	
SEQUENCE	1	MUST echo the original SEQUENCE number.
UID	1	MUST echo original UID.
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DTSTART	0 or 1	
DTEND	0 or 1	If present, DURATION MUST NOT be present.

DURATION	0 or 1	If present, DTEND MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PRIORITY	0 or 1	
RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
REQUEST-STATUS	0+	
RESOURCES	0+	
RRULE	0 or 1	
STATUS	0 or 1	MAY be one of TENTATIVE/CONFIRMED.
SUMMARY	0 or 1	Can be null.
TRANSP	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VALARM	0	
VFREEBUSY	0	
VJOURNAL	0	
VTODO	0	

3.3. Methods for VFREEBUSY Components

This section defines the property set for the methods that are applicable to the "VFREEBUSY" calendar component. Each of the methods is defined using a restriction table.

This document only addresses the transfer of busy time information. Applications desiring free time information MUST infer this from available busy time information.

The "FREEBUSY" property value MAY include a list of values, separated by the COMMA character (US-ASCII decimal 44). Alternately, multiple busy time periods MAY be specified with multiple instances of the "FREEBUSY" property. Both forms MUST be supported by implementations conforming to this document. Duplicate busy time periods SHOULD NOT be specified in an iCalendar object. However, two different busy time periods MAY overlap.

"FREEBUSY" properties SHOULD be sorted such that their values are in ascending order, based on the start time and then the end time, with the earliest periods first. For example, today’s busy time information should appear after yesterday’s busy time information. And the busy time for this half-hour should appear after the busy time for earlier today. Busy time periods can also span a day boundary.

The following summarizes the methods that are defined for the "VFREEBUSY" calendar component.

Method	Description
PUBLISH	Publish unsolicited busy time data.
REQUEST	Request busy time data.
REPLY	Reply to a busy time request.

3.3.1. PUBLISH

The "PUBLISH" method in a "VFREEBUSY" calendar component is used to publish busy time data. The method may be sent from one CU to any other. The purpose of the method is to provide a way to send unsolicited busy time data. That is, the busy time data is not being sent as a "REPLY" to the receipt of a "REQUEST" method.

The "ORGANIZER" property MUST be specified in the busy time information. The value is the CU address of the originator of the busy time information.

The busy time information within the iCalendar object MAY be grouped into more than one "VFREEBUSY" calendar component. This capability allows busy time periods to be grouped according to some common periodicity, such as a calendar week, month, or year. In this case, each "VFREEBUSY" calendar component MUST include the "ORGANIZER", "DTSTART", and "DTEND" properties in order to specify the source of the busy time information and the date and time interval over which the busy time information covers.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:PUBLISH of a VFREEBUSY

Component/Property	Presence	Comment
METHOD	1	MUST be PUBLISH.
VFREEBUSY	1+	
DTSTAMP	1	
DTSTART	1	DateTime values must be in UTC.
DTEND	1	DateTime values must be in UTC.
FREEBUSY	0+	MUST be BUSYTIME. Multiple instances are allowed. Multiple instances SHOULD be sorted in ascending order.
ORGANIZER	1	MUST contain the address of originator of busy time data.
UID	1	
COMMENT	0+	
CONTACT	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
URL	0 or 1	Specifies busy time URL.
ATTENDEE	0	
DURATION	0	
REQUEST-STATUS	0	
VALARM	0	
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VTODO	0	
VJOURNAL	0	
VTIMEZONE	0	

3.3.2. REQUEST

The "REQUEST" method in a "VFREEBUSY" calendar component is used to ask a "Calendar User" for their busy time information. The request may be for a busy time information bounded by a specific date and time interval.

This message only permits requests for busy time information. The message is sent from a "Calendar User" requesting the busy time information of one or more intended recipients.

If the originator of the "REQUEST" method is not authorized to make a busy time request on the recipient's calendar system, then an exception message SHOULD be returned in a "REPLY" method, but no busy time data need be returned.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:REQUEST of a VFREEBUSY

Component/Property	Presence	Comment
METHOD	1	MUST be REQUEST.
VFREEBUSY	1	
ATTENDEE	1+	Contains the calendar user addresses of the "Calendar Users" whose freebusy is being requested.
DTEND	1	DateTime values must be in UTC.
DTSTAMP	1	
DTSTART	1	DateTime values must be in UTC.
ORGANIZER	1	MUST be the request originator's address.
UID	1	
COMMENT	0+	
CONTACT	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
FREEBUSY	0	
DURATION	0	
REQUEST-STATUS	0	
URL	0	
VALARM	0	
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VTODO	0	
VJOURNAL	0	
VTIMEZONE	0	

3.3.3. REPLY

The "REPLY" method in a "VFREEBUSY" calendar component is used to respond to a busy time request. The method is sent by the recipient of a busy time request to the originator of the request.

The "REPLY" method may also be used to respond to an unsuccessful "REQUEST" method. Depending on the "REQUEST-STATUS" value, no busy time information may be returned.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:REPLY of a VFREEBUSY

Component/Property	Presence	Comment
METHOD	1	MUST be REPLY.
VFREEBUSY	1	
ATTENDEE	1	MUST be the address of the Attendee replying.
DTSTAMP	1	
DTEND	1	DateTime values must be in UTC.
DTSTART	1	DateTime values must be in UTC.
FREEBUSY	0+	MUST be BUSYTIME. Multiple instances are allowed. Multiple instances SHOULD be sorted in ascending order.
ORGANIZER	1	MUST be the request originator's address.
UID	1	MUST be the UID of the original REQUEST.
COMMENT	0+	
CONTACT	0 or 1	
REQUEST-STATUS	0+	
URL	0 or 1	Specifies busy time URL.
IANA-PROPERTY	0+	
X-PROPERTY	0+	
DURATION	0	
SEQUENCE	0	
VALARM	0	
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VTODO	0	
VJOURNAL	0	
VTIMEZONE	0	

3.4. Methods for VTODO Components

This section defines the property set for the methods that are applicable to the "VTODO" calendar component. Each of the methods is defined using a restriction table that specifies the property constraints that define the particular method.

The following summarizes the methods that are defined for the "VTODO" calendar component.

Method	Description
PUBLISH	Post notification of a VTODO. Used primarily as a method of advertising the existence of a VTODO.
REQUEST	Assign a VTODO. This is an explicit assignment to one or more Calendar Users. The REQUEST method is also used to update or change an existing VTODO. Clients that cannot handle REQUEST MAY degrade the method to treat it as a PUBLISH.
REPLY	Reply to a VTODO request. Attendees MAY set PARTSTAT to ACCEPTED, DECLINED, TENTATIVE, DELEGATED, PARTIAL, and COMPLETED.
ADD	Add one or more instances to an existing to-do.
CANCEL	Cancel one or more instances of an existing to-do.
REFRESH	A request sent to a VTODO Organizer asking for the latest version of a VTODO.
COUNTER	Counter a REQUEST with an alternative proposal.
DECLINECOUNTER	Decline a counter proposal by an Attendee.

3.4.1. PUBLISH

The "PUBLISH" method in a "VTODO" calendar component has no associated response. It is simply a posting of an iCalendar object that may be added to a calendar. It MUST have an "Organizer". It MUST NOT have "Attendees". Its expected usage is for encapsulating an arbitrary "VTODO" calendar component as an iCalendar object. The

"Organizer" MAY subsequently update (with another "PUBLISH" method), add instances to (with an "ADD" method), or cancel (with a "CANCEL" method) a previously published "VTODO" calendar component.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:PUBLISH of a VTODO

Component/Property	Presence	Comment
METHOD	1	MUST be PUBLISH.
VTODO	1+	
DTSTAMP	1	
DTSTART	1	
ORGANIZER	1	
PRIORITY	1	
SEQUENCE	0 or 1	MUST be present if value is greater than 0; MAY be present if 0.
SUMMARY	1	Can be null.
UID	1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DUE	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DUE MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PERCENT-COMPLETE	0 or 1	
RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.

RELATED-TO	0+	
RESOURCES	0+	
RRULE	0 or 1	
STATUS	0 or 1	MAY be one of COMPLETED/NEEDS-ACTION/ IN-PROCESS/CANCELLED.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
ATTENDEE	0	
REQUEST-STATUS	0	
VALARM	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VFREEBUSY	0	
VEVENT	0	
VJOURNAL	0	

3.4.2. REQUEST

The "REQUEST" method in a "VTODO" calendar component provides the following scheduling functions:

- Assign a to-do to one or more "Calendar Users".
- Reschedule an existing to-do.
- Update the details of an existing to-do, without rescheduling it.
- Update the completion status of "Attendees" of an existing to-do, without rescheduling it.
- Reconfirm an existing to-do, without rescheduling it.
- Delegate/reassign an existing to-do to another "Calendar User".

The assigned "Calendar Users" are identified in the "VTODO" calendar component by individual "ATTENDEE;ROLE=REQ-PARTICIPANT" property value sequences.

Typically, the originator of a "REQUEST" is the "Organizer" of the to-do, and the recipient of a "REQUEST" is the "Calendar User" assigned the to-do. The "Attendee" uses the "REPLY" method to convey their acceptance and completion status to the "Organizer" of the "REQUEST".

The "UID", "SEQUENCE", and "DTSTAMP" properties are used to distinguish the various uses of the "REQUEST" method. If the "UID" property value in the "REQUEST" is not found on the recipient's calendar, then the "REQUEST" is for a new to-do. If the "UID" property value is found on the recipient's calendar, then the "REQUEST" is a rescheduling, an update, or a reconfirmation of the "VTODO" calendar object.

If the "Organizer" of the "REQUEST" method is not authorized to make a to-do request on the "Attendee's" calendar system, then an exception is returned in the "REQUEST-STATUS" property of a subsequent "REPLY" method, but no scheduling action is performed.

For the "REQUEST" method, multiple "VTODO" components in a single iCalendar object are only permitted for components with the same "UID" property. That is, a series of recurring events may have instance-specific information. In this case, multiple "VTODO" components are needed to express the entire series.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:REQUEST of a VTODO

Component/Property	Presence	Comment
METHOD	1	MUST be REQUEST.
VTODO	1+	All components must have the same UID.
ATTENDEE	1+	
DTSTAMP	1	
DTSTART	1	
ORGANIZER	1	
PRIORITY	1	
SEQUENCE	0 or 1	MUST be present if value is greater than 0; MAY be present if 0.
SUMMARY	1	Can be null.

UID	1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null
DUE	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DUE MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PERCENT-COMPLETE	0 or 1	
RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
RESOURCES	0+	
RRULE	0 or 1	
STATUS	0 or 1	MAY be one of COMPLETED/NEEDS-ACTION/ IN-PROCESS.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
REQUEST-STATUS	0	
VALARM	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VFREEBUSY	0	
VJOURNAL	0	

3.4.2.1. REQUEST for Rescheduling a VTODO

The "REQUEST" method may be used to reschedule a "VTODO" calendar component.

Rescheduling a "VTODO" calendar component involves a change to the existing "VTODO" calendar component in terms of its start or due time, recurrence intervals, and possibly the description. If the recipient CUA of a "REQUEST" method finds that the "UID" property value already exists on the calendar but that the "SEQUENCE" property value in the "REQUEST" is greater than the value for the existing "VTODO", then the "REQUEST" method describes a rescheduling of the "VTODO" calendar component.

3.4.2.2. REQUEST for Update or Reconfirmation of a VTODO

The "REQUEST" method may be used to update or reconfirm a "VTODO" calendar component. Reconfirmation is merely an update of "Attendee" completion status or overall "VTODO" calendar component status.

An update to an existing "VTODO" calendar component does not involve changes to the start or due time, recurrence intervals, or (generally) the description for the "VTODO" calendar component. If the recipient CUA of a "REQUEST" method finds that the "UID" property value already exists on the calendar and that the "SEQUENCE" property value in the "REQUEST" is the same as the value for the existing event, then the "REQUEST" method describes an update of the "VTODO" calendar component details, but not a rescheduling of the "VTODO" calendar component.

The update "REQUEST" is the appropriate response to a "REFRESH" method sent from an "Attendee" to the "Organizer" of a "VTODO" calendar component.

Unsolicited "REQUEST" methods MAY be sent by the "Organizer" of a "VTODO" calendar component. The unsolicited "REQUEST" methods are used to update the details of the "VTODO" (without rescheduling it or updating the completion status of "Attendees") or the "VTODO" calendar component itself (i.e., reconfirm the "VTODO").

3.4.2.3. REQUEST for Delegating a VTODO

The "REQUEST" method is also used to delegate or reassign ownership of a "VTODO" calendar component to another "Calendar User". For example, it may be used to delegate an "Attendee's" role (i.e., "chair" or "participant") for a "VTODO" calendar component. The "REQUEST" method is sent by one of the "Attendees" of an existing "VTODO" calendar component to some other individual.

For the purposes of this description, the "Attendee" delegating the "VTODO" calendar component is referred to as the "Delegator". The "Attendee" receiving the delegation request is referred to as the "Delegate".

The "Delegator" of a "VTODO" calendar component MUST forward the existing "REQUEST" method for a "VTODO" calendar component to the "Delegate". The "VTODO" calendar component description MUST include the "Delegator's" up-to-date "VTODO" calendar component definition. The "REQUEST" method MUST also include an "ATTENDEE" property with the calendar address of the "Delegate". The "Delegator" MUST also send a "REPLY" method back to the "Organizer" with the "Delegator's" "Attendee" property "PARTSTAT" parameter value set to "DELEGATED". In addition, the "DELEGATED-TO" parameter MUST be included with the calendar address of the "Delegate". A response to the delegation "REQUEST" is sent from the "Delegate" to the "Organizer", and optionally to the "Delegator". The "REPLY" method from the "Delegate" SHOULD include the "ATTENDEE" property with their calendar address and the "DELEGATED-FROM" parameter with the value of the "Delegator's" calendar address.

The delegation "REQUEST" method MUST assign a value for the "RSVP" property parameter associated with the "Delegator's" "Attendee" property to that of the "Delegate's" "ATTENDEE" property. For example, if the "Delegator's" "ATTENDEE" property specifies "RSVP=TRUE", then the "Delegate's" "ATTENDEE" property MUST specify "RSVP=TRUE".

3.4.2.4. REQUEST Forwarded to an Uninvited Calendar User

An "Attendee" assigned a "VTODO" calendar component may send the "VTODO" calendar component to another new CU not previously associated with the "VTODO" calendar component. The current "Attendee" assigned the "VTODO" calendar component does this by forwarding the original "REQUEST" method to the new CU. The new CU can send a "REPLY" to the "Organizer" of the "VTODO" calendar component. The reply contains an "ATTENDEE" property for the new CU.

The "Organizer" ultimately decides whether or not the new CU becomes part of the to-do and is not obligated to do anything with a "REPLY" from a new (uninvited) CU. If the "Organizer" does not want the new CU to be part of the to-do, the new "ATTENDEE" property is not added to the "VTODO" calendar component. The "Organizer" MAY send the CU a "CANCEL" message to indicate that they will not be added to the to-do. If the "Organizer" decides to add the new CU, the new "ATTENDEE" property is added to the "VTODO" calendar component. Furthermore, the "Organizer" is free to change any "ATTENDEE" property parameter from the values supplied by the new CU to something the "Organizer"

considers appropriate. The "Organizer" SHOULD send the new "Attendee" a "REQUEST" message to inform them that they have been added.

When forwarding a "REQUEST" to another CU, the forwarding "Attendee" MUST NOT make changes to the original message.

3.4.2.5. REQUEST Updated Attendee Status

An "Organizer" of a "VTODO" may request an updated status from one or more "Attendees". The "Organizer" sends a "REQUEST" method to the "Attendee" with the "ATTENDEE;RSVP=TRUE" property sequence. The "SEQUENCE" property for the "VTODO" is not changed from its previous value. A recipient determines that the only change in the "REQUEST" is that their "RSVP" property parameter indicates a request for an updated status. The recipient SHOULD respond with a "REPLY" method indicating their current status with respect to the "REQUEST".

3.4.3. REPLY

The "REPLY" method in a "VTODO" calendar component is used to respond (e.g., accept or decline) to a request or to reply to a delegation request. It is also used by an "Attendee" to update their completion status. When used to provide a delegation response, the "Delegator" MUST include the calendar address of the "Delegate" in the "DELEGATED-TO" parameter of the "Delegator's" "ATTENDEE" property. The "Delegate" MUST include the calendar address of the "Delegator" on the "DELEGATED-FROM" parameter of the "Delegate's" "ATTENDEE" property.

The "REPLY" method MAY also be used to respond to an unsuccessful "VTODO" calendar component "REQUEST" method. Depending on the "REQUEST-STATUS" value, no scheduling action may have been performed.

The "Organizer" of a "VTODO" calendar component MAY receive a "REPLY" method from a "Calendar User" not in the original "REQUEST". For example, a "REPLY" method MAY be received from a "Delegate" of a "VTODO" calendar component. In addition, the "REPLY" method MAY be received from an unknown "Calendar User" who has been forwarded the "REQUEST" by an original "Attendee" of the "VTODO" calendar component. This uninvited "Attendee" MAY be accepted or the "Organizer" MAY cancel the "VTODO" calendar component for the uninvited "Attendee" by sending them a "CANCEL" method.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:REPLY of a VTODO

Component/Property	Presence	Comment
METHOD	1	MUST be REPLY.
VTODO	1+	All components MUST have the same UID.
ATTENDEE	1	MUST be the address of the Attendee replying.
DTSTAMP	1	
ORGANIZER	1	
REQUEST-STATUS	0+	
UID	1	MUST be the UID of the original REQUEST.
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTSTART	0 or 1	
DUE	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DUE MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PERCENT-COMPLETE	0 or 1	
PRIORITY	0 or 1	
RDATE	0+	
RELATED-TO	0+	
RESOURCES	0+	
RRULE	0 or 1	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
SEQUENCE	0 or 1	MUST be the sequence number of the original REQUEST if greater than 0. MAY be present if 0.

STATUS	0 or 1	
SUMMARY	0 or 1	Can be null.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
VALARM	0	
VTIMEZONE	0 or 1	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VFREEBUSY	0	

3.4.4. ADD

The "ADD" method allows the "Organizer" to add one or more new instances to an existing "VTODO" using a single iTIP message without having to send the entire "VTODO" with all the existing instance data, as it would have to do if the "REQUEST" method were used.

The "UID" must be that of the existing to-do. If the "UID" property value in the "ADD" is not found on the recipient's calendar, then the recipient SHOULD send a "REFRESH" to the "Organizer" in order to be updated with the latest version of the "VTODO". If an "Attendee" implementation does not support the "ADD" method, it should respond with a "REQUEST-STATUS" value of 3.14 and ask for a "REFRESH".

When handling an "ADD" message, the "Attendee" treats each component in the "ADD" message as if it were referenced via an "RDATE" in the main component.

The "SEQUENCE" property value is incremented since the sequence of to-dos has changed.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:ADD of a VTODO

Component/Property	Presence	Comment
METHOD	1	MUST be ADD.
VTODO	1	
DTSTAMP	1	
ORGANIZER	1	
PRIORITY	1	
SEQUENCE	1	MUST be greater than 0.
SUMMARY	1	Can be null.
UID	1	MUST match that of the original to-do.
ATTACH	0+	
ATTENDEE	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DTSTART	0 or 1	
DUE	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DUE MUST NOT be present.
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PERCENT-COMPLETE	0 or 1	
RELATED-TO	0+	
RESOURCES	0+	
STATUS	0 or 1	MAY be one of COMPLETED/NEEDS-ACTION/ IN-PROCESS.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
EXDATE	0	
RECURRENCE-ID	0	
REQUEST-STATUS	0	
RDATE	0	
RRULE	0	

VALARM	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VJOURNAL	0	
VFREEBUSY	0	

3.4.5. CANCEL

The "CANCEL" method in a "VTODO" calendar component is used to send a cancellation notice of an existing "VTODO" calendar request to the affected "Attendees". The message is sent by the "Organizer" of a "VTODO" calendar component to the "Attendees" of the "VTODO" calendar component. For a recurring "VTODO" calendar component, either the whole "VTODO" calendar component or instances of a "VTODO" calendar component may be cancelled. To cancel the complete range of a recurring "VTODO" calendar component, the "UID" property value for the "VTODO" calendar component MUST be specified and a "RECURRENCE-ID" MUST NOT be specified in the "CANCEL" method. In order to cancel an individual instance of a recurring "VTODO" calendar component, the "RECURRENCE-ID" property value for the "VTODO" calendar component MUST be specified in the "CANCEL" method.

There are two options for canceling a sequence of instances of a recurring "VTODO" calendar component:

a. The "RECURRENCE-ID" property for an instance in the sequence MUST be specified with the "RANGE" property parameter value of "THISANDFUTURE" to indicate cancellation of the specified "VTODO" calendar component and all instances after.

b. Individual recurrence instances may be cancelled by specifying multiple "VTODO" components each with a "RECURRENCE-ID" property corresponding to one of the instances to be cancelled.

The "Organizer" MUST send a "CANCEL" message to each "Attendee" affected by the cancellation. This can be done by using either a single "CANCEL" message for all "Attendees" or multiple messages with different subsets of the affected "Attendees" in each.

When a "VTODO" is cancelled, the "SEQUENCE" property value MUST be incremented as described in Section 2.1.4.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:CANCEL of a VTODO

Component/Property	Presence	Comment
METHOD	1	MUST be CANCEL.
VTODO	1+	
ATTENDEE	0+	MUST include some or all Attendees being removed from the to-do. MUST include some or all Attendees if the entire to-do is cancelled.
UID	1	MUST echo original UID.
DTSTAMP	1	
ORGANIZER	1	
SEQUENCE	1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTSTART	0 or 1	
DUE	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DUE MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PERCENT-COMPLETE	0 or 1	
RDATE	0+	

RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
RESOURCES	0+	
RRULE	0 or 1	
PRIORITY	0 or 1	
STATUS	0 or 1	MUST be set to CANCELLED to cancel the entire VTODO. If removing specific Attendees, then MUST NOT be included.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
REQUEST-STATUS	0	
VALARM	0	
VTIMEZONE	0 or 1	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VFREEBUSY	0	

3.4.6. REFRESH

The "REFRESH" method in a "VTODO" calendar component is used by "Attendees" of an existing "VTODO" calendar component to request an updated description from the "Organizer" of the "VTODO" calendar component. The "Organizer" of the "VTODO" calendar component MAY use this method to request an updated status from the "Attendees". The "REFRESH" method MUST specify the "UID" property corresponding to the "VTODO" calendar component needing update.

A refresh of a recurrence instance of a "VTODO" calendar component may be requested by specifying the "RECURRENCE-ID" property corresponding to the associated "VTODO" calendar component. The "Organizer" responds with the latest description and rendition of the "VTODO" calendar component. In most cases, this will be a "REQUEST" unless the "VTODO" has been cancelled, in which case the "Organizer" MUST send a "CANCEL". This method is intended to facilitate machine processing of requests for updates to a "VTODO" calendar component.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:REFRESH of a VTODO

Component/Property	Presence	Comment
METHOD	1	MUST be REFRESH.
VTODO	1	
ATTENDEE	1	
DTSTAMP	1	
UID	1	MUST echo original UID.
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
IANA-PROPERTY	0+	
X-PROPERTY	0+	
ATTACH	0	
CATEGORIES	0	
CLASS	0	
COMMENT	0	
COMPLETED	0	
CONTACT	0	
CREATED	0	
DESCRIPTION	0	
DTSTART	0	
DUE	0	
DURATION	0	
EXDATE	0	
GEO	0	
LAST-MODIFIED	0	
LOCATION	0	
ORGANIZER	0	
PERCENT-COMPLETE	0	
PRIORITY	0	
RDATE	0	
RELATED-TO	0	
REQUEST-STATUS	0	
RESOURCES	0	
RRULE	0	
SEQUENCE	0	
STATUS	0	
URL	0	

VALARM	0	
VTIMEZONE	0+	
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VFREEBUSY	0	

3.4.7. COUNTER

The "COUNTER" method in a "VTODO" calendar component is used by an "Attendee" of an existing "VTODO" calendar component to submit to the "Organizer" a counter proposal for the "VTODO" calendar component.

The counter proposal is an iCalendar object consisting of a "VTODO" calendar component that provides the complete description of the alternate "VTODO" calendar component.

The "Organizer" rejects the counter proposal by sending the "Attendee" a "DECLINECOUNTER" method. The "Organizer" accepts the counter proposal by rescheduling the to-do as described in Section 3.4.2.1, "REQUEST for Rescheduling a To-Do". The "Organizer's" CUA SHOULD send a "REQUEST" message to all "Attendees" affected by any change triggered by an accepted "COUNTER".

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:COUNTER of a VTODO

Component/Property	Presence	Comment
METHOD	1	MUST be COUNTER.
VTODO	1	
ATTENDEE	1+	
DTSTAMP	1	
ORGANIZER	1	
PRIORITY	1	
SUMMARY	1	Can be null.

UID	1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	Can be null.
DTSTART	0 or 1	
DUE	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DUE MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	
LOCATION	0 or 1	
PERCENT-COMPLETE	0 or 1	
RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
REQUEST-STATUS	0+	
RESOURCES	0+	
RRULE	0 or 1	
SEQUENCE	0 or 1	MUST echo the original SEQUENCE number. MUST be present if non-zero. MAY be present if zero.
STATUS	0 or 1	MAY be one of COMPLETED/NEEDS-ACTION/ IN-PROCESS/CANCELLED.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
VALARM	0+	
VTIMEZONE	0 or 1	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	

VEVENT	0	
VFREEBUSY	0	

3.4.8. DECLINECOUNTER

The "DECLINECOUNTER" method in a "VTODO" calendar component is used by an "Organizer" of the "VTODO" calendar component to reject a counter proposal offered by one of the "Attendees". The "Organizer" sends the message to the "Attendee" that sent the "COUNTER" method to the "Organizer".

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:DECLINECOUNTER of a VTODO

Component/Property	Presence	Comment
METHOD	1	MUST be DECLINECOUNTER.
VTODO	1	
ATTENDEE	1+	MUST for all ATTENDEEs.
DTSTAMP	1	
ORGANIZER	1	
SEQUENCE	1	MUST echo the original SEQUENCE number.
UID	1	MUST echo original UID.
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
COMPLETED	0 or 1	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTSTART	0 or 1	
DUE	0 or 1	If present, DURATION MUST NOT be present.
DURATION	0 or 1	If present, DUE MUST NOT be present.
EXDATE	0+	
GEO	0 or 1	
LAST-MODIFIED	0 or 1	

LOCATION	0 or 1	
PERCENT-COMPLETE	0 or 1	
PRIORITY	0 or 1	
RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
REQUEST-STATUS	0+	
RESOURCES	0+	
RRULE	0 or 1	
STATUS	0 or 1	MAY be one of COMPLETED/NEEDS-ACTION/ IN-PROCESS.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
VALARM	0	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VFREEBUSY	0	

3.5. Methods for VJOURNAL Components

This section defines the property set for the methods that are applicable to the "VJOURNAL" calendar component.

The following summarizes the methods that are defined for the "VJOURNAL" calendar component.

Method	Description
PUBLISH	Post a journal entry. Used primarily as a method of advertising the existence of a journal entry.
ADD	Add one or more instances to an existing journal entry.
CANCEL	Cancel one or more instances of an existing journal entry.

3.5.1. PUBLISH

The "PUBLISH" method in a "VJOURNAL" calendar component has no associated response. It is simply a posting of an iCalendar object that may be added to a calendar. It MUST have an "Organizer". It MUST NOT have "Attendees". The expected usage is for encapsulating an arbitrary journal entry as an iCalendar object. The "Organizer" MAY subsequently update (with another "PUBLISH" method) or cancel (with a "CANCEL" method) a previously published journal entry.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:PUBLISH of a VJOURNAL

Component/Property	Presence	Comment
METHOD	1	MUST be PUBLISH.
VJOURNAL	1+	
DESCRIPTION	1	Can be null.
DTSTAMP	1	
DTSTART	1	
ORGANIZER	1	
UID	1	
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0+	
CREATED	0 or 1	
EXDATE	0+	
LAST-MODIFIED	0 or 1	

RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
RRULE	0 or 1	
SEQUENCE	0 or 1	MUST be present if non-zero. MAY be present if zero.
STATUS	0 or 1	MAY be one of DRAFT/FINAL/CANCELLED.
SUMMARY	0 or 1	Can be null.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
ATTENDEE	0	
REQUEST-STATUS	0	
VALARM	0+	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	
VEVENT	0	
VFREEBUSY	0	
VTODO	0	

3.5.2. ADD

The "ADD" method allows the "Organizer" to add one or more new instances to an existing "VJOURNAL" using a single iTIP message without having to send the entire "VJOURNAL" with all the existing instance data, as it would have to do if the "REQUEST" method were used.

The "UID" must be that of the existing journal entry. If the "UID" property value in the "ADD" is not found on the recipient's calendar, then the recipient MAY treat the "ADD" as a "PUBLISH".

When handling an "ADD" message, the "Attendee" treats each component in the "ADD" message as if it were referenced via an "RDATE" in the main component. There is no response to the "Organizer".

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:ADD of a VJOURNAL

Component/Property	Presence	Comment
METHOD	1	MUST be ADD.
VJOURNAL	1	
DESCRIPTION	1	Can be null.
DTSTAMP	1	
DTSTART	1	
ORGANIZER	1	
SEQUENCE	1	MUST be greater than 0.
UID	1	MUST match that of the original journal.
ATTACH	0+	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0+	
CREATED	0 or 1	
LAST-MODIFIED	0 or 1	
RELATED-TO	0+	
STATUS	0 or 1	MAY be one of DRAFT/FINAL/CANCELLED.
SUMMARY	0 or 1	Can be null.
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
ATTENDEE	0	
EXDATE	0	
RECURRENCE-ID	0	
REQUEST-STATUS	0	
RDATE	0	
RRULE	0	
VALARM	0+	
VTIMEZONE	0 or 1	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	

VEVENT	0	
VFREEBUSY	0	
VTODO	0	

3.5.3. CANCEL

The "CANCEL" method in a "VJOURNAL" calendar component is used to send a cancellation notice of an existing journal entry. The message is sent by the "Organizer" of a journal entry. For a recurring journal entry, either the whole journal entry or instances of a journal entry may be cancelled. To cancel the complete range of a recurring journal entry, the "UID" property value for the journal entry MUST be specified and a "RECURRENCE-ID" property MUST NOT be specified in the "CANCEL" method. In order to cancel an individual instance of the journal entry, the "RECURRENCE-ID" property value for the journal entry MUST be specified in the "CANCEL" method.

There are two options for canceling a sequence of instances of a recurring "VJOURNAL" calendar component:

a. The "RECURRENCE-ID" property for an instance in the sequence MUST be specified with the "RANGE" property parameter value of "THISANDFUTURE" to indicate cancellation of the specified "VJOURNAL" calendar component and all instances after.

b. Individual recurrence instances may be cancelled by specifying multiple "VJOURNAL" components each with a "RECURRENCE-ID" property corresponding to one of the instances to be cancelled.

When a "VJOURNAL" is cancelled, the "SEQUENCE" property value MUST be incremented as described in Section 2.1.4.

This method type is an iCalendar object that conforms to the following property constraints:

Constraints for a METHOD:CANCEL of a VJOURNAL

Component/Property	Presence	Comment
METHOD	1	MUST be CANCEL.
VJOURNAL	1+	All MUST have the same UID.
DTSTAMP	1	
ORGANIZER	1	
SEQUENCE	1	
UID	1	MUST be the UID of the original REQUEST.
ATTACH	0+	
ATTENDEE	0	
CATEGORIES	0+	
CLASS	0 or 1	
COMMENT	0+	
CONTACT	0+	
CREATED	0 or 1	
DESCRIPTION	0 or 1	
DTSTART	0 or 1	
EXDATE	0+	
LAST-MODIFIED	0 or 1	
RDATE	0+	
RECURRENCE-ID	0 or 1	Only if referring to an instance of a recurring calendar component. Otherwise, it MUST NOT be present.
RELATED-TO	0+	
RRULE	0 or 1	
STATUS	0 or 1	MAY be present; MUST be CANCELLED if present.
SUMMARY	0 or 1	
URL	0 or 1	
IANA-PROPERTY	0+	
X-PROPERTY	0+	
REQUEST-STATUS	0	
VALARM	0	
VTIMEZONE	0+	MUST be present if any date/time refers to a timezone.
IANA-COMPONENT	0+	
X-COMPONENT	0+	

VEVENT	0	
VFREEBUSY	0	
VTODO	0	

3.6. Status Replies

The "REQUEST-STATUS" property is used to convey status information about a "REPLY", "COUNTER", or "DECLINECOUNTER" iTIP message. The codes listed in the table below SHOULD be used. If the "REQUEST-STATUS" property is not present in one of these iTIP messages, then a status code of "2.0" (success) MUST be assumed.

This specification adds a new IANA registry for "REQUEST-STATUS" property values, as defined in Section 7, which includes a new registration template for defining the specific components of the "REQUEST-STATUS" property value. Additional codes MAY be used, provided the process described in Section 8.2.1 of [RFC5545] is used to register them.

This specification allows for multiple "REQUEST-STATUS" properties to be returned in iCalendar components in the appropriate iTIP messages. When multiple "REQUEST-STATUS" properties are present, the following restrictions apply:

1. Within any one component, the "top-level" numeric value of the "short return status code" MUST be the same for all "REQUEST-STATUS" properties, i.e., there cannot be a mixture of, e.g., 2.xx and 5.xx codes within a single component.

2. Across all components in the iTIP message, the following applies:

   A. If any one component would have a 5.xx code, then either all components MUST have a code in that range or "REQUEST-STATUS" MUST NOT be present in the other components if a 5.xx code is not appropriate for those components.

   B. Otherwise, if any one component would have a 3.xx code, then either all components MUST have a code in that range or "REQUEST-STATUS" MUST NOT be present in the other components if a 3.xx code is not appropriate for those components.

   C. 2.xx and 4.xx codes can be used in different components, provided that each component follows the restriction in (1) above.

The following "REQUEST-STATUS" codes are defined (any "Offending Data" MAY be specified in the "REQUEST-STATUS" value as the extdata field):

3.6.1. Status Code 2.0

Status Code: 2.0

Status Description: Success.

Status Exception Data: None.

Description: iTIP operation succeeded.

3.6.2. Status Code 2.1

Status Code: 2.1

Status Description: Success, but fallback taken on one or more property values.

Status Exception Data: Property name and value MAY be specified.

Description: iTIP operation succeeded with fallback on one or more property values.

3.6.3. Status Code 2.2

Status Code: 2.2

Status Description: Success; invalid property ignored.

Status Exception Data: Property name MAY be specified.

Description: iTIP operation succeeded but a property was ignored.

3.6.4. Status Code 2.3

Status Code: 2.3

Status Description: Success; invalid property parameter ignored.

Status Exception Data: Property parameter name and value MAY be specified.

Description: iTIP operation succeeded but a property parameter was ignored because it was invalid.

3.6.5. Status Code 2.4

Status Code: 2.4

Status Description: Success; unknown, non-standard property ignored.

Status Exception Data: Non-standard property name MAY be specified.

Description: iTIP operation succeeded but a property parameter was ignored because it was unknown.

3.6.6. Status Code 2.5

Status Code: 2.5

Status Description: Success; unknown, non-standard property value ignored.

Status Exception Data: Property and non-standard value MAY be specified.

Description: iTIP operation succeeded but a property was ignored because its value was unknown.

3.6.7. Status Code 2.6

Status Code: 2.6

Status Description: Success; invalid calendar component ignored.

Status Exception Data: Calendar component sentinel (e.g., BEGIN: ALARM) MAY be specified.

Description: iTIP operation succeeded but a component was ignored because it was invalid.

3.6.8. Status Code 2.7

Status Code: 2.7

Status Description: Success; request forwarded to Calendar User.

Status Exception Data: Original and forwarded calendar user addresses MAY be specified.

Description: iTIP operation succeeded, and the request was forwarded to another Calendar User.

3.6.9. Status Code 2.8

Status Code: 2.8

Status Description: Success; repeating event ignored. Scheduled as a single component.

Status Exception Data: RRULE or RDATE property name and value MAY be specified.

Description: iTIP operation succeeded but a repeating event was truncated to a single instance.

3.6.10. Status Code 2.9

Status Code: 2.9

Status Description: Success; truncated end date time to date boundary.

Status Exception Data: DTEND property value MAY be specified.

Description: iTIP operation succeeded but the end time was truncated to a date boundary.

3.6.11. Status Code 2.10

Status Code: 2.10

Status Description: Success; repeating VTODO ignored. Scheduled as a single VTODO.

Status Exception Data: RRULE or RDATE property name and value MAY be specified.

Description: iTIP operation succeeded but a repeating to-do was truncated to a single instance.

3.6.12. Status Code 2.11

Status Code: 2.11

Status Description: Success; unbounded RRULE clipped at some finite number of instances.

Status Exception Data: RRULE property name and value MAY be specified. Number of instances MAY also be specified.

Description: iTIP operation succeeded but an unbounded repeating object was clipped to a finite number of instances.

3.6.13. Status Code 3.0

Status Code: 3.0

Status Description: Invalid property name.

Status Exception Data: Property name MAY be specified.

Description: iTIP operation failed because of an invalid property name.

3.6.14. Status Code 3.1

Status Code: 3.1

Status Description: Invalid property value.

Status Exception Data: Property name and value MAY be specified.

Description: iTIP operation failed because of an invalid property value.

3.6.15. Status Code 3.2

Status Code: 3.2

Status Description: Invalid property parameter.

Status Exception Data: Property parameter name and value MAY be specified.

Description: iTIP operation failed because of an invalid property parameter.

3.6.16. Status Code 3.3

Status Code: 3.3

Status Description: Invalid property parameter value.

Status Exception Data: Property parameter name and value MAY be specified.

Description: iTIP operation failed because of an invalid property parameter value.

3.6.17. Status Code 3.4

Status Code: 3.4

Status Description: Invalid calendar component sequence.

Status Exception Data: Calendar component sentinel MAY be specified (e.g., BEGIN:VTIMEZONE).

Description: iTIP operation failed because of an invalid component.

3.6.18. Status Code 3.5

Status Code: 3.5

Status Description: Invalid date or time.

Status Exception Data: Date/time value(s) MAY be specified.

Description: iTIP operation failed because of an invalid date or time property.

3.6.19. Status Code 3.6

Status Code: 3.6

Status Description: Invalid rule.

Status Exception Data: RRULE property value MAY be specified.

Description: iTIP operation failed because of an invalid rule property.

3.6.20. Status Code 3.7

Status Code: 3.7

Status Description: Invalid Calendar User.

Status Exception Data: ATTENDEE property value MAY be specified.

Description: iTIP operation failed because of an invalid ATTENDEE property.

3.6.21. Status Code 3.8

Status Code: 3.8

Status Description: No authority.

Status Exception Data: METHOD and ATTENDEE property values MAY be specified.

Description: iTIP operation failed because an Attendee does not have suitable privileges for the operation.

3.6.22. Status Code 3.9

Status Code: 3.9

Status Description: Unsupported version.

Status Exception Data: VERSION property name and value MAY be specified.

Description: iTIP operation failed because the calendar data version is not supported.

3.6.23. Status Code 3.10

Status Code: 3.10

Status Description: Request entity too large.

Status Exception Data: None.

Description: iTIP operation failed because the calendar data was too large.

3.6.24. Status Code 3.11

Status Code: 3.11

Status Description: Required component or property missing.

Status Exception Data: Component or property name MAY be specified.

Description: iTIP operation failed because the calendar data did not contain a required property or component.

3.6.25. Status Code 3.12

Status Code: 3.12

Status Description: Unknown component or property found.

Status Exception Data: Component or property name MAY be specified.

Description: iTIP operation failed because the calendar data contained an unknown property or component.

3.6.26. Status Code 3.13

Status Code: 3.13

Status Description: Unsupported component or property found.

Status Exception Data: Component or property name MAY be specified.

Description: iTIP operation failed because the calendar data contained an unsupported property or component.

3.6.27. Status Code 3.14

Status Code: 3.14

Status Description: Unsupported capability.

Status Exception Data: METHOD or action MAY be specified.

Description: iTIP operation failed because the operation is not supported.

3.6.28. Status Code 4.0

Status Code: 4.0

Status Description: Event conflict. Date/time is busy.

Status Exception Data: DTSTART and DTEND property names and values MAY be specified.

Description: iTIP operation failed because the event overlaps the date and time of another event.

3.6.29. Status Code 5.0

Status Code: 5.0

Status Description: Request not supported.

Status Exception Data: METHOD property value MAY be specified.

Description: iTIP operation failed because the operation is not supported.

3.6.30. Status Code 5.1

Status Code: 5.1

Status Description: Service unavailable.

Status Exception Data: ATTENDEE property value MAY be specified.

Description: iTIP operation failed because scheduling is not active.

3.6.31. Status Code 5.2

Status Code: 5.2

Status Description: Invalid calendar service.

Status Exception Data: ATTENDEE property value MAY be specified.

Description: iTIP operation failed because there is no scheduling capability.

3.6.32. Status Code 5.3

Status Code: 5.3

Status Description: No scheduling support for user.

Status Exception Data: ATTENDEE property value MAY be specified.

Description: iTIP operation failed because scheduling is not enabled for an Attendee.

3.7. Implementation Considerations

3.7.1. Working With Recurrence Instances

iCalendar includes a recurrence grammar to represent recurring events. The benefit of such a grammar is the ability to represent a number of events in a single object. However, while this simplifies creation of a recurring event, meeting instances still need to be referenced. For instance, an "Attendee" may decline the third instance of a recurring Friday event. Similarly, the "Organizer" may change the time or location to a single instance of the recurring event.

Since implementations may elect to store recurring events as either a single event object or a collection of discrete, related event objects, the protocol is designed so that each recurring instance may be both referenced and versioned. Hence, implementations that choose to maintain per-instance properties (such as "ATTENDEE" property "PARTSTAT" parameter) may do so. However, the protocol does not require per-instance recognition unless the instance itself must be renegotiated.

The scenarios for recurrence instance referencing are listed below. For purposes of simplification, a change to an event refers to a "trigger property." That is, a property that has a substantive effect on the meeting itself, such as start time, location, due date (for "VTODO" calendar components), and possibly description.

"Organizer"-initiated actions:

- deletes or changes a single instance of a recurring event
- makes changes that affect all future instances
- makes changes that affect all previous instances
- deletes or modifies a previously changed instance

"Attendee"-initiated actions:

- changes status for a particular recurrence instance
- sends a "COUNTER" for a particular recurrence instance

An instance of a recurring event is assigned a unique identification, "RECURRENCE-ID" property, when that instance is renegotiated. Negotiation may be necessary when a substantive change to the event or to-do has been made (such as changing the start time, end time, due date, or location). The "Organizer" can identify a specific recurrence instance using the "RECURRENCE-ID" property. The property value is equal to the date/time of the instance. If the "Organizer" wishes to change the "DTSTART", the original, unmodified "DTSTART" value of the instance is used as the value "RECURRENCE-ID" property, and the new "DTSTART" and "DTEND" values reflect the change.

3.7.2. Attendee Property Considerations

The "ORGANIZER" property is required on published events, to-dos, and journal entries for two reasons. First, only the "Organizer" is allowed to update and redistribute an event or to-do component. It follows that the "ORGANIZER" property MUST be present in the event, to-do, or journal entry component so that the CUA has a basis for authorizing an update. Second, it is prudent to provide a point of contact for anyone who receives a published component, in case of problems.

Email addresses that correspond to groups of "Calendar Users" could be specified as a mailto: URI [RFC2368] calendar user address. Sending email to such an address results in email being sent to multiple recipients. Such an address may be used as the value of an "ATTENDEE" property. Thus, it is possible that the recipient of a "REQUEST" does not appear explicitly in the list.

It is recommended that the general approach to finding a "Calendar User" in an "Attendee" list be as follows:

1. Search for the "Calendar User" in the "Attendee" list where "CUTYPE=INDIVIDUAL"

2. Failing (1), look for "Attendees" where "CUTYPE=GROUP" or "CUTYPE=UNKNOWN". The CUA then determines if the "Calendar User" is a member of one of these groups. If so, the "REPLY" method sent to the "Organizer" MUST contain a new "ATTENDEE" property in which:

   * the "TYPE" property parameter is set to INDIVIDUAL

* the "MEMBER" property parameter is set to the name of the group

3. Failing (2), the CUA MAY ignore or accept the request as the "Calendar User" wishes.

3.7.3. Extension Tokens

To make iCalendar objects extensible, new component, property, or property parameters can be used. Two types of extensions are defined by [RFC5545]: IANA-registered tokens ("iana-token") and experimental use tokens ("x-name"). A client SHOULD save "iana-token's" and MAY use them in replies. A client MAY save "x-name's" and MAY use them in replies. When delegating or forwarding messages to other CUs, a client SHOULD include "iana-token's" and "x-names's".

4. Examples

4.1. Published Event Examples

In the calendaring and scheduling context, publication refers to the one-way transfer of event information. Consumers of published events simply incorporate the event into a calendar. No reply is expected. Individual "A" publishes an event. Individual "B" reads the event and incorporates it into their calendar. Events are published in several ways, including embedding the event as an object in a web page, emailing the event to a distribution list, or posting the event to a newsgroup.

The table below illustrates the sequence of events between the publisher and the consumers of a published event.

Action	Organizer	Receiver
Publish an event	"A" sends or posts a PUBLISH message.	"B" reads a published event.
Publish an updated event	"A" sends or posts a PUBLISH message.	"B" reads the updated event.
Cancel a published event	"A" sends or posts a CANCEL message.	"B" reads the canceled event publication.

4.1.1. A Minimal Published Event

The iCalendar object below describes a single event that begins on July 1, 1997 at 20:00 UTC. This event contains the minimum set of properties for a "PUBLISH" for a "VEVENT" calendar component.

```
BEGIN:VCALENDAR
METHOD:PUBLISH
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
DTSTART:19970701T200000Z
DTSTAMP:19970611T190000Z
SUMMARY:ST. PAUL SAINTS -VS- DULUTH-SUPERIOR DUKES
UID:0981234-1234234-23@example.com
END:VEVENT
END:VCALENDAR
```

4.1.2. Changing a Published Event

The iCalendar object below describes an update to the event described in Section 4.1.1; the time has been changed, an end time has been added, and the sequence number has been adjusted.

```
BEGIN:VCALENDAR
METHOD:PUBLISH
VERSION:2.0
PRODID:-//Example/ExampleCalendarClient//EN
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
DTSTAMP:19970612T190000Z
DTSTART:19970701T210000Z
DTEND:19970701T230000Z
SEQUENCE:1
UID:0981234-1234234-23@example.com
SUMMARY:ST. PAUL SAINTS -VS- DULUTH-SUPERIOR DUKES
END:VEVENT
END:VCALENDAR
```

The "UID" property is used by the client to identify the event. The "SEQUENCE" property indicates that this is a change to the event. The event with a matching "UID" and sequence number 0 is superseded by this event.

The "SEQUENCE" property provides a reliable way to distinguish different versions of the same event. Each time an event is published, its sequence number is incremented. If a client receives

an event with a sequence number 5 and finds it has the same event with sequence number 2, the event SHOULD be updated. However, if the client received an event with sequence number 2 and finds it already has sequence number 5 of the same event, the event MUST NOT be updated.

4.1.3. Canceling a Published Event

The iCalendar object below cancels the event described in Section 4.1.1. This cancels the event with "SEQUENCE" property of 0, 1, and 2.

```
BEGIN:VCALENDAR
METHOD:CANCEL
VERSION:2.0
PRODID:-//Example/ExampleCalendarClient//EN
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
COMMENT:DUKES forfeit the game
SEQUENCE:2
UID:0981234-1234234-23@example.com
DTSTAMP:19970613T190000Z
END:VEVENT
END:VCALENDAR
```

4.1.4. A Rich Published Event

This example describes the same event as in Section 4.1.1, but in much greater detail.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:PUBLISH
SCALE:GREGORIAN
VERSION:2.0
BEGIN:VTIMEZONE
TZID:America-Chicago
TZURL:http://example.com/tz/America-Chicago
BEGIN:STANDARD
DTSTART:19671029T020000
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
TZOFFSETFROM:-0500
TZOFFSETTO:-0600
TZNAME:CST
END:STANDARD
BEGIN:DAYLIGHT
DTSTART:19870405T020000
RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
```

```
TZOFFSETFROM:-0600
TZOFFSETTO:-0500
TZNAME:CDT
END:DAYLIGHT
END:VTIMEZONE
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTACH:http://www.example.com/
CATEGORIES:SPORTS EVENT,ENTERTAINMENT
CLASS:PRIVATE
DESCRIPTION:MIDWAY STADIUM\n
 Big time game.  MUST see.\n
 Expected duration:2 hours\n
DTEND;TZID=America-Chicago:19970701T180000
DTSTART;TZID=America-Chicago:19970702T160000
DTSTAMP:19970614T190000Z
STATUS:CONFIRMED
LOCATION;VALUE=URI:http://stadium.example.com/
PRIORITY:2
RESOURCES:SCOREBOARD
SEQUENCE:3
SUMMARY:ST. PAUL SAINTS -VS- DULUTH-SUPERIOR DUKES
UID:0981234-1234234-23@example.com
RELATED-TO:0981234-1234234-14@example.com
BEGIN:VALARM
TRIGGER:-PT2H
ACTION:DISPLAY
DESCRIPTION:You should be leaving for the game now.
END:VALARM
BEGIN:VALARM
TRIGGER:-PT30M
ACTION:AUDIO
END:VALARM
END:VEVENT
END:VCALENDAR
```

The "RELATED-TO" field contains the "UID" property of a related calendar event. The "SEQUENCE" property 3 indicates that this event supersedes versions 0, 1, and 2.

4.1.5. Anniversaries or Events Attached to Entire Days

This example demonstrates the use of the "VALUE" parameter to tie a "VEVENT" to a day rather than a specific time.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:PUBLISH
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
DTSTAMP:19970614T190000Z
UID:0981234-1234234-23@example.com
DTSTART;VALUE=DATE:19970714
RRULE:FREQ=YEARLY;INTERVAL=1
SUMMARY: Bastille Day
END:VEVENT
END:VCALENDAR
```

4.2. Group Event Examples

Group events are distinguished from published events in that they have "Attendees" and there is interaction between the "Attendees" and the "Organizer" with respect to the event. Individual "A" requests a meeting between individuals "A", "B", "C", and "D". Individual "B" confirms attendance to the meeting. Individual "C" declines attendance. Individual "D" tentatively confirms attendance. The following table illustrates the message flow between these individuals. "A", the CU scheduling the meeting, is referenced as the "Organizer".

Action	"Organizer"	Attendee
Initiate a meeting request	"A" sends a REQUEST message to "B", "C", and "D".	
Accept the meeting request		"B" sends a REPLY message to "A" with its ATTENDEE PARTSTAT parameter set to ACCEPTED.
Decline the meeting request		"C" sends a REPLY message to "A" with its ATTENDEE PARTSTAT parameter set to DECLINED.
Tentatively accept the meeting request		"D" sends a REPLY message to "A" with its ATTENDEE PARTSTAT parameter set to TENTATIVE.
Confirm meeting status with Attendees	"A" sends a REQUEST message to "B" and "D" with updated information.	

4.2.1. A Group Event Request

A sample meeting request is sent from "A" to "B", "C", and "D". "E" is also sent a copy of the request but is not expected to attend and need not reply. "E" illustrates how CUAs might implement an "FYI"-type feature. Note the use of the "ROLE" parameter. The default value for the "ROLE" parameter is "REQ-PARTICIPANT" and it need not be enumerated. In this case, we are using the value "NON-PARTICIPANT" to indicate "E" is a non-attending CU. The parameter is not needed on other "Attendees" since "PARTICIPANT" is the default value.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED;CN=A:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL;CN=B:mailto:b@example.com
```

```
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL;CN=C:mailto:c@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL;CN=Hal:mailto:d@example.com
ATTENDEE;RSVP=FALSE;CUTYPE=ROOM:conf_big@example.com
ATTENDEE;ROLE=NON-PARTICIPANT;RSVP=FALSE:mailto:e@example.com
DTSTAMP:19970611T190000Z
DTSTART:19970701T200000Z
DTEND:19970701T2100000Z
SUMMARY:Conference
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:0
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.2.2. Reply to a Group Event Request

"Attendee" "B" accepts the meeting.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VEVENT
ATTENDEE;PARTSTAT=ACCEPTED:mailto:b@example.com
ORGANIZER:mailto:a@example.com
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:0
REQUEST-STATUS:2.0;Success
DTSTAMP:19970612T190000Z
END:VEVENT
END:VCALENDAR
```

"B" could have declined the meeting or indicated tentative acceptance by setting the "ATTENDEE" "PARTSTAT" parameter to "DECLINED" or "TENTATIVE", respectively. Also, "REQUEST-STATUS" is not required in successful transactions.

4.2.3. Update an Event

The event is moved to a different time. The combination of the "UID" property (unchanged) and the "SEQUENCE" (bumped to 1) properties indicate the update.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
```

```
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:b@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:c@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL;CN=Hal:mailto:d@example.com
ATTENDEE;ROLE=NON-PARTICIPANT;RSVP=FALSE;
 CUTYPE=ROOM:mailto:conf@example.com
ATTENDEE;ROLE=NON-PARTICIPANT;RSVP=FALSE:mailto:e@example.com
DTSTART:19970701T180000Z
DTEND:19970701T190000Z
SUMMARY:Phone Conference
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:1
DTSTAMP:19970613T190000Z
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.2.4. Countering an Event Proposal

"A" sends a "REQUEST" to "B" and "C". "B" makes a counter proposal to "A" to change the time and location.

"A" sends the following "REQUEST":

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:b@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:c@example.com
DTSTART:19970701T190000Z
DTEND:19970701T200000Z
SUMMARY:Discuss the Merits of the election results
LOCATION:Green Conference Room
UID:calsrv.example.com-873970198738777a@example.com
SEQUENCE:0
DTSTAMP:19970611T190000Z
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

"B" sends "COUNTER" to "A", requesting changes to time and place. "B" uses the "COMMENT" property to communicate a rationale for the change. Note that the "SEQUENCE" property is not incremented on a "COUNTER".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:COUNTER
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:b@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:c@example.com
DTSTART:19970701T160000Z
DTEND:19970701T170000Z
DTSTAMP:19970612T190000Z
SUMMARY:Discuss the Merits of the election results
LOCATION:Blue Conference Room
COMMENT:This time works much better and I think the big conference
 room is too big
UID:calsrv.example.com-873970198738777a@example.com
SEQUENCE:0
END:VEVENT
END:VCALENDAR
```

"A" accepts the changes from "B". To accept a counter proposal, the "Organizer" sends a new event "REQUEST" with an incremented sequence number.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:b@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:c@example.com
DTSTAMP:19970613T190000Z
DTSTART:19970701T160000Z
DTEND:19970701T170000Z
SUMMARY:Discuss the Merits of the election results - changed to
 meet B's schedule
LOCATION:Blue Conference Room
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:1
STATUS:CONFIRMED
```

```
END:VEVENT
END:VCALENDAR
```

Instead, "A" rejects "B's" counter proposal.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:DECLINECOUNTER
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:b@example.com
COMMENT:Sorry, I cannot change this meeting time
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:0
DTSTAMP:19970614T190000Z
END:VEVENT
END:VCALENDAR
```

4.2.5. Delegating an Event

When delegating an event request to another "Calendar User", the "Delegator" must both update the "Organizer" with a "REPLY" and send a request to the "Delegate". There is currently no protocol limitation to delegation depth. It is possible for the original delegate to delegate the meeting to someone else, and so on. When a request is delegated from one CUA to another, there are a number of responsibilities required of the "Delegator". The "Delegator" MUST:

- Send a "REPLY" to the "Organizer" with the following updates:

  A. The "Delegator's" "ATTENDEE" property "PARTSTAT" parameter is set to "DELEGATED" and the "DELEGATED-TO" parameter is set to the address of the "Delegate".

  B. Add an additional "ATTENDEE" property for the "Delegate" with the "DELEGATED-FROM" property parameter set to the "Delegator".

  C. Indicate whether they want to continue to receive updates when the "Organizer" sends out updated versions of the event. Setting the "RSVP" property parameter to "TRUE" will cause the updates to be sent; setting it to "FALSE" causes no further updates to be sent. Note that in either case, if the "Delegate" declines the invitation, the "Delegator" will be notified.

o The "Delegator" MUST also send a copy of the original "REQUEST" method to the "Delegate", with changes (A) and (B), as detailed above applied.

If the "Delegate" declines the meeting, the "Organizer" MUST send an update "REQUEST" to the "Delegator" so that the "Delegator" may elect to delegate the "REQUEST" to another CUA.

Action	"Organizer"	Attendee
Initiate a meeting request	"A" sends a REQUEST message to "B" and "C".	
Delegate: "C" delegates to "E"		"C" sends a REPLY to "A" with the ATTENDEE PARTSTAT parameter set to DELEGATED and with a new ATTENDEE property for "E". "E's" ATTENDEE DELEGATED-FROM parameter is set to "C". "C's" ATTENDEE DELEGATED-TO parameter is set to "E". "C" sends REQUEST message to "E" with the original meeting request information. The PARTSTAT property parameter for "C" is set to DELEGATED and the DELEGATED-TO parameter is set to the address of "E". An ATTENDEE property is added for "E" and the DELEGATED-FROM parameter is set to the address of "C".
Confirm meeting attendance		"E" sends REPLY message to "A", and optionally to "C", with its PARTSTAT property parameter set to ACCEPTED.
Optional: Redistribute meeting to Attendees	"A" sends REQUEST message to "B", "C", and "E".	

"C" responds to the "Organizer".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=DELEGATED;DELEGATED-
 TO="mailto:e@example.com":mailto:c@example.com
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:0
REQUEST-STATUS:2.0;Success
DTSTAMP:19970611T190000Z
END:VEVENT
END:VCALENDAR
```

"Attendee" "C" delegates presence at the meeting to "E".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=DELEGATED;DELEGATED-
 TO="mailto:e@example.com":mailto:c@example.com
ATTENDEE;RSVP=TRUE;
 DELEGATED-FROM="mailto:c@example.com":mailto:e@example.com
DTSTART:19970701T180000Z
DTEND:19970701T200000Z
SUMMARY:Phone Conference
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:0
STATUS:CONFIRMED
DTSTAMP:19970611T190000Z
END:VEVENT
END:VCALENDAR
```

4.2.6. Delegate Accepts the Meeting

To accept a delegated meeting, the delegate, "E", sends the following message to "A" and "C".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
```

```
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=ACCEPTED;DELEGATED-
 FROM="mailto:c@example.com":mailto:e@example.com
ATTENDEE;PARTSTAT=DELEGATED;
 DELEGATED-TO="mailto:e@example.com":mailto:c@example.com
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:0
REQUEST-STATUS:2.0;Success
DTSTAMP:19970614T190000Z
END:VEVENT
END:VCALENDAR
```

4.2.7. Delegate Declines the Meeting

In this example, the "Delegate" declines the meeting request and sets the "ATTENDEE" property "PARTSTAT" parameter to "DECLINED". The "Organizer" SHOULD resend the "REQUEST" to "C" with the "PARTSTAT" parameter of the "Delegate" set to "DECLINED". This lets the "Delegator" know that the "Delegate" has declined and provides an opportunity to the "Delegator" to either accept the request or delegate it to another CU.

"E" responds to "A" and "C". Note the use of the "COMMENT" property "E" uses to indicate why the delegation was declined.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=DELEGATED;
 DELEGATED-TO="mailto:e@example.com":mailto:c@example.com
ATTENDEE;PARTSTAT=DECLINED;
 DELEGATED-FROM="mailto:c@example.com":mailto:e@example.com
COMMENT:Sorry, I will be out of town at that time.
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:0
REQUEST-STATUS:2.0;Success
DTSTAMP:19970614T190000Z
END:VEVENT
END:VCALENDAR
```

"A" resends the "REQUEST" method to "C". "A" may also wish to express the fact that the item was delegated in the "COMMENT" property.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=DECLINED;
 DELEGATED-FROM="mailto:c@example.com":mailto:e@example.com
ATTENDEE;RSVP=TRUE:mailto:c@example.com
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:0
SUMMARY:Phone Conference
DTSTART:19970701T180000Z
DTEND:19970701T200000Z
DTSTAMP:19970614T200000Z
COMMENT:DELEGATE (ATTENDEE mailto:e@example.com) DECLINED YOUR
 INVITATION
END:VEVENT
END:VCALENDAR
```

4.2.8. Forwarding an Event Request

The protocol does not prevent an "Attendee" from "forwarding" a "VEVENT" calendar component to other "Calendar Users". Forwarding differs from delegation in that the forwarded "Calendar User" (often referred to as a "Party Crasher") does not replace the forwarding "Calendar User". Implementations are not required to add the "Party Crasher" to the "Attendee" list, and hence there is no guarantee that a "Party Crasher" will receive additional updates to the event. The forwarding "Calendar User" SHOULD NOT add the "Party Crasher" to the "Attendee" list. The "Organizer" MAY add the forwarded "Calendar User" to the "Attendee" list.

4.2.9. Cancel a Group Event

Individual "A" requests a meeting between individuals "A", "B", "C", and "D". Individual "B" declines attendance to the meeting. Individual "A" decides to cancel the meeting. The following table illustrates the sequence of messages that would be exchanged between these individuals.

Messages related to a previously canceled event ("SEQUENCE" property value is less than the "SEQUENCE" property value of the "CANCEL" message) MUST be ignored.

Action	Organizer	Attendee
Initiate a meeting request	"A" sends a REQUEST message to "B", "C", and "D".	
Decline the meeting request		"B" sends a REPLY message to "A" with its PARTSTAT parameter set to DECLINED.
Cancel the meeting	"A" sends a CANCEL message to "B", "C", and "D".	

This example shows how "A" cancels the event.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:CANCEL
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;CUTYPE=INDIVIDUAL;mailto:a@example.com
ATTENDEE;CUTYPE=INDIVIDUAL:mailto:b@example.com
ATTENDEE;CUTYPE=INDIVIDUAL:mailto:c@example.com
ATTENDEE;CUTYPE=INDIVIDUAL:mailto:d@example.com
COMMENT:Mr. B cannot attend.  It’s raining.  Lets cancel.
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:1
STATUS:CANCELLED
DTSTAMP:19970613T190000Z
END:VEVENT
END:VCALENDAR
```

4.2.10. Removing Attendees

"A" wants to remove "B" from a meeting. This is done by sending a "CANCEL" to "B" and removing "B" from the "Attendee" list in the master copy of the event.

Action	Organizer	Attendee
Remove "B" as an Attendee Update the master copy of the event	"A" sends a CANCEL message to "B". "A" optionally sends the updated event to the remaining Attendees.	

The original meeting includes "A", "B", "C", and "D". The example below shows the "CANCEL" that "A" sends to "B". Note that in the example below, the "STATUS" property is omitted. This is used when the meeting itself is cancelled and not when the intent is to remove an "Attendee" from the event.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:CANCEL
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE:mailto:b@example.com
COMMENT:You're off the hook for this meeting
UID:calsrv.example.com-873970198738777@example.com
DTSTAMP:19970613T193000Z
SEQUENCE:1
END:VEVENT
END:VCALENDAR
```

The updated master copy of the event is shown below. The "Organizer" MAY resend the updated event to the remaining "Attendees". Note that "B" has been removed.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;CUTYPE=INDIVIDUAL:mailto:c@example.com
ATTENDEE;CUTYPE=INDIVIDUAL:mailto:d@example.com
ATTENDEE;CUTYPE=ROOM:mailto:cr_big@example.com
ATTENDEE;ROLE=NON-PARTICIPANT;
 RSVP=FALSE:mailto:e@example.com
DTSTAMP:19970611T190000Z
DTSTART:19970701T200000Z
```

```
DTEND:19970701T203000Z
SUMMARY:Phone Conference
UID:calsrv.example.com-873970198738777@example.com
SEQUENCE:2
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.2.11. Replacing the Organizer

The scenario for this example begins with "A" as the "Organizer" for a recurring meeting with "B", "C", and "D". "A" receives a new job offer in another country and drops out of touch. "A" left no forwarding address or way to be reached. Using out-of-band communication, the other "Attendees" eventually learn what has happened and reach an agreement that "B" should become the new "Organizer" for the meeting. To do this, "B" sends out a new version of the event and the other "Attendees" agree to accept "B" as the new "Organizer". "B" also removes "A" from the event.

When the "Organizer" is replaced, the "SEQUENCE" property value MUST be incremented.

This is the message "B" sends to "C" and "D".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:b@example.com
ATTENDEE;ROLE=CHAIR;STATUS=ACCEPTED:mailto:b@example.com
ATTENDEE;CUTYPE=INDIVIDUAL:mailto:c@example.com
ATTENDEE;CUTYPE=INDIVIDUAL:mailto:d@example.com
DTSTAMP:19970611T190000Z
DTSTART:19970701T200000Z
DTEND:19970701T203000Z
RRULE:FREQ=WEEKLY
SUMMARY:Phone Conference
UID:123456@example.com
SEQUENCE:1
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.3. Busy Time Examples

Busy time objects can be used in several ways. First, a CU may request busy time from another CU for a specific range of time. That request can be answered with a busy time "REPLY". Additionally, a CU may simply publish their busy time for a given interval and point other CUs to the published location. The following examples outline both scenarios.

4.3.1. Publish Busy Time

Individual "A" publishes busy time for one week.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
METHOD:PUBLISH
BEGIN:VFREEBUSY
DTSTAMP:19980101T124100Z
ORGANIZER:mailto:a@example.com
DTSTART:19980101T124200Z
DTEND:19980108T124200Z
FREEBUSY:19980101T180000Z/19980101T190000Z
FREEBUSY:19980103T020000Z/19980103T050000Z
FREEBUSY:19980107T020000Z/19980107T050000Z
FREEBUSY:19980113T000000Z/19980113T010000Z
FREEBUSY:19980115T190000Z/19980115T200000Z
FREEBUSY:19980115T220000Z/19980115T230000Z
FREEBUSY:19980116T013000Z/19980116T043000Z
END:VFREEBUSY
END:VCALENDAR
```

4.3.2. Request Busy Time

Individual "A" requests busy time from individuals "B" and "C". Individuals "B" and "C" reply with busy time data to individual "A". The following table illustrates the sequence of messages that would be exchanged between these individuals.

Action	Organizer	Attendee
Initiate a busy time request	"A" sends REQUEST message to "B" and "C".	
Reply to the BUSY request with BUSY time data		"B" sends a REPLY message to "A" with busy time data.

"A" sends a "REQUEST" to "B" and "C" for busy time.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VFREEBUSY
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR:mailto:a@example.com
ATTENDEE:mailto:b@example.com
ATTENDEE:mailto:c@example.com
DTSTAMP:19970613T190000Z
DTSTART:19970701T080000Z
DTEND:19970701T200000
UID:calsrv.example.com-873970198738777@example.com
END:VFREEBUSY
END:VCALENDAR
```

4.3.3. Reply to a Busy Time Request

"B" sends a "REPLY" method type of a "VFREEBUSY" calendar component to "A".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VFREEBUSY
ORGANIZER:mailto:a@example.com
ATTENDEE:mailto:b@example.com
DTSTART:19970701T080000Z
DTEND:19970701T200000Z
UID:calsrv.example.com-873970198738777@example.com
FREEBUSY:19970701T090000Z/PT1H,19970701T140000Z/PT30M
DTSTAMP:19970613T190030Z
END:VFREEBUSY
```

```
END:VCALENDAR
```

"B" is busy from 09:00 to 10:00 and from 14:00 to 14:30.

4.4. Recurring Event and Time Zone Examples

4.4.1. A Recurring Event Spanning Time Zones

This event describes a weekly phone conference. The "Attendees" are each in a different time zone.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VTIMEZONE
TZID:America-SanJose
TZURL:http://example.com/tz/America-SanJose
BEGIN:STANDARD
DTSTART:19671029T020000
RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
TZOFFSETFROM:-0700
TZOFFSETTO:-0800
TZNAME:PST
END:STANDARD
BEGIN:DAYLIGHT
DTSTART:19870405T020000
RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
TZOFFSETFROM:-0800
TZOFFSETTO:-0700
TZNAME:PDT
END:DAYLIGHT
END:VTIMEZONE
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED;
 CUTYPE=INDIVIDUAL:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:b@example.fr
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:c@example.jp
DTSTAMP:19970613T190030Z
DTSTART;TZID=America-SanJose:19970701T140000
DTEND;TZID=America-SanJose:19970701T150000
RRULE:FREQ=WEEKLY;COUNT=20;WKST=SU;BYDAY=TU
RDATE;TZID=America-SanJose:19970910T140000
EXDATE;TZID=America-SanJose:19970909T140000
EXDATE;TZID=America-SanJose:19971028T140000
SUMMARY:Weekly Phone Conference
UID:calsrv.example.com-873970198738777@example.com
```

```
SEQUENCE:0
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

The first component of this iCalendar object is the time zone component. The "DTSTART" date coincides with the first instance of the "RRULE" property.

The recurring meeting is defined in a particular time zone, presumably that of the originator. The client for each "Attendee" has the responsibility of determining the recurrence time in the "Attendee's" time zone.

The repeating event starts on Tuesday, July 1, 1997 at 2:00pm PDT (UTC-7). "Attendee" B@example.fr is in France, where the local time on this date is 9 hours ahead of PDT, or 23:00 CEST (UTC+2). "Attendee" C@example.jp is in Japan, where local time is 16 hours ahead of PDT, or Wednesday, July 2 at 06:00 JST (UTC+9). The event repeats weekly on Tuesdays (in PST/PDT). The "RRULE" property results in 20 instances. The last instance falls on Tuesday, November 11, 1997 2:00pm PST. The "RDATE" property adds another instance: WED, 10-SEP-1997 2:00 PM PDT.

There are also two exception dates to the recurrence rule. The first one is:

- TUE, 09-SEP-1997 14:00 PDT (UTC-7)
- TUE, 09-SEP-1997 23:00 CEST (UTC+2)
- WED, 10-SEP-1997 06:00 JST (UTC+9)

and the second is:

- TUE, 28-OCT-1997 14:00 PST (UTC-8)
- TUE, 28-OCT-1997 23:00 CET (UTC+1)
- WED, 29-OCT-1997 07:00 JST (UTC+9)

4.4.2. Modify a Recurring Instance

In this example, the "Organizer" issues a recurring meeting. Later, the "Organizer" changes an instance of the event by changing the "DTSTART" property. Note the use of "RECURRENCE-ID" property and "SEQUENCE" property in the second request.

Original Request:

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:guid-1@example.com
SEQUENCE:0
RRULE:FREQ=MONTHLY;BYMONTHDAY=1;UNTIL=19980901T210000Z
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE:mailto:b@example.com
ATTENDEE:mailto:c@example.com
ATTENDEE:mailto:d@example.com
DESCRIPTION:IETF-C&S Conference Call
CLASS:PUBLIC
SUMMARY:IETF Calendaring Working Group Meeting
DTSTART:19970601T210000Z
DTEND:19970601T220000Z
LOCATION:Conference Call
DTSTAMP:19970526T083000Z
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

The event request below is to change the time of a specific instance. This changes the July 1st instance to July 3rd.

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:guid-1@example.com
RECURRENCE-ID:19970701T210000Z
SEQUENCE:1
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE:mailto:b@example.com
ATTENDEE:mailto:c@example.com
ATTENDEE:mailto:d@example.com
DESCRIPTION:IETF-C&S Conference Call
CLASS:PUBLIC
SUMMARY:IETF Calendaring Working Group Meeting
DTSTART:19970703T210000Z
DTEND:19970703T220000Z
LOCATION:Conference Call
```

```
DTSTAMP:19970626T093000Z
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.4.3. Cancel an Instance

In this example, the "Organizer" of a recurring event deletes the August 1st instance.

```
BEGIN:VCALENDAR
METHOD:CANCEL
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:guid-1@example.com
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE:mailto:b@example.com
ATTENDEE:mailto:c@example.com
ATTENDEE:mailto:d@example.com
RECURRENCE-ID:19970801T210000Z
SEQUENCE:2
STATUS:CANCELLED
DTSTAMP:19970721T093000Z
END:VEVENT
END:VCALENDAR
```

4.4.4. Cancel a Recurring Event

In this example, the "Organizer" wishes to cancel the entire recurring event and any exceptions.

```
BEGIN:VCALENDAR
METHOD:CANCEL
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:guid-1@example.com
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE:mailto:b@example.com
ATTENDEE:mailto:c@example.com
ATTENDEE:mailto:d@example.com
DTSTAMP:19970721T103000Z
STATUS:CANCELLED
SEQUENCE:3
END:VEVENT
```

```
END:VCALENDAR
```

4.4.5. Change All Future Instances

This example changes the meeting location from a conference call to Seattle, starting September 1 and extending to all future instances.

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:guid-1@example.com
RECURRENCE-ID;THISANDFUTURE:19970901T210000Z
SEQUENCE:3
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
ATTENDEE;RSVP=TRUE:mailto:c@example.com
ATTENDEE;RSVP=TRUE:mailto:d@example.com
DESCRIPTION:IETF-C&S Discussion
CLASS:PUBLIC
SUMMARY:IETF Calendaring Working Group Meeting
DTSTART:19970901T210000Z
DTEND:19970901T220000Z
LOCATION:Building 32, Microsoft, Seattle, WA
DTSTAMP:19970526T083000Z
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.4.6. Add a New Instance to a Recurring Event

This example adds a one-time additional instance to the recurring event. "Organizer" adds a second July meeting on the 15th.

```
BEGIN:VCALENDAR
METHOD:ADD
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:123456789@example.com
SEQUENCE:4
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
ATTENDEE;RSVP=TRUE:mailto:c@example.com
ATTENDEE;RSVP=TRUE:mailto:d@example.com
```

```
DESCRIPTION:IETF-C&S Conference Call
CLASS:PUBLIC
SUMMARY:IETF Calendaring Working Group Meeting
DTSTART:19970715T210000Z
DTEND:19970715T220000Z
LOCATION:Conference Call
DTSTAMP:19970629T093000Z
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.4.7. Add a New Series of Instances to a Recurring Event

The scenario for this example involves an ongoing meeting, originally set up to occur every Tuesday. The "Organizer" later decides that the meetings need to be on Tuesdays and Thursdays.

The original event:

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:123456789@example.com
SEQUENCE:0
RRULE:WKST=SU;BYDAY=TU;FREQ=WEEKLY
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
SUMMARY:Review Accounts
DTSTART:19980303T210000Z
DTEND:19980303T220000Z
LOCATION:The White Room
DTSTAMP:19980301T093000Z
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

The entire event can be rescheduled using a "REQUEST". This is done by using the "UID" of the event to reschedule and including the modified "RRULE". Note that since this is an entire rescheduling of the event, any instance-specific information will be lost, unless explicitly included with the update "REQUEST".

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
```

```
VERSION:2.0
BEGIN:VEVENT
UID:123456789@example.com
SEQUENCE:7
RRULE:WKST=SU;BYDAY=TU,TH;FREQ=WEEKLY
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
SUMMARY:Review Accounts
DTSTART:19980303T210000Z
DTEND:19980303T220000Z
DTSTAMP:19980303T193000Z
LOCATION:The White Room
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.4.8. Refreshing a Recurring Event

The next series of examples illustrate how an "Organizer" would respond to a "REFRESH" submitted by an "Attendee" after a series of instance-specific modifications. To convey all instance-specific changes, the "Organizer" must provide the latest event description and the relevant instances. The first three examples show the history, including the initial "VEVENT" request and subsequent instance changes, and finally the "REFRESH".

Original Request:

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:123456789@example.com
SEQUENCE:0
RDATE:19980304T180000Z
RDATE:19980311T180000Z
RDATE:19980318T180000Z
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
SUMMARY:Review Accounts
DTSTART:19980304T180000Z
DTEND:19980304T200000Z
DTSTAMP:19980303T193000Z
LOCATION:Conference Room A
STATUS:CONFIRMED
```

```
END:VEVENT
END:VCALENDAR
```

Organizer changes 2nd instance location and time:

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:123456789@example.com
SEQUENCE:1
RECURRENCE-ID:19980311T180000Z
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
SUMMARY:Review Accounts
DTSTART:19980311T160000Z
DTEND:19980311T180000Z
DTSTAMP:19980306T193000Z
LOCATION:The Small conference room
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

Organizer adds a 4th instance of the meeting using the "ADD" method.

```
BEGIN:VCALENDAR
METHOD:ADD
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:123456789@example.com
SEQUENCE:2
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
SUMMARY:Review Accounts
DTSTART:19980315T180000Z
DTEND:19980315T200000Z
DTSTAMP:19980307T193000Z
LOCATION:Conference Room A
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

If "B" requests a "REFRESH", "A" responds with the following to capture all instance-specific data. In this case, both the initial request and an additional "VEVENT" that specifies the instance-specific data are included. Because these are both of the same type (they are both "VEVENTS"), they can be conveyed in the same iCalendar object.

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:123456789@example.com
SEQUENCE:2
RDATE:19980304T180000Z
RDATE:19980311T160000Z
RDATE:19980315T180000Z
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
SUMMARY:Review Accounts
DTSTART:19980304T180000Z
DTEND:19980304T200000Z
DTSTAMP:19980303T193000Z
LOCATION:Conference Room A
STATUS:CONFIRMED
END:VEVENT
BEGIN:VEVENT
SEQUENCE:2
UID:123456789@example.com
RECURRENCE-ID:19980311T160000Z
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
SUMMARY:Review Accounts
DTSTART:19980311T160000Z
DTEND:19980304T180000Z
DTSTAMP:19980306T193000Z
LOCATION:The Small conference room
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

4.4.9. Counter an Instance of a Recurring Event

In this example, one of the "Attendees" counters the "DTSTART" property of the proposed second July meeting.

```
BEGIN:VCALENDAR
METHOD:COUNTER
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:guid-1@example.com
RECURRENCE-ID:19970715T210000Z
SEQUENCE:4
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;RSVP=TRUE:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
ATTENDEE;RSVP=TRUE:mailto:c@example.com
ATTENDEE;RSVP=TRUE:mailto:d@example.com
DESCRIPTION:IETF-C&S Conference Call
CLASS:PUBLIC
SUMMARY:IETF Calendaring Working Group Meeting
DTSTART:19970715T220000Z
DTEND:19970715T230000Z
LOCATION:Conference Call
COMMENT:May we bump this by an hour? I have a conflict
DTSTAMP:19970629T094000Z
END:VEVENT
END:VCALENDAR
```

4.4.10. Error Reply to a Request

The following example illustrates a scenario where a meeting is proposed containing an unsupported property and a bad property.

Original Request:

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:guid-1@example.com
SEQUENCE:0
RRULE:FREQ=MONTHLY;BYMONTHDAY=1
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
ATTENDEE;RSVP=TRUE:mailto:c@example.com
ATTENDEE;RSVP=TRUE:mailto:d@example.com
DESCRIPTION:IETF-C&S Conference Call
CLASS:PUBLIC
SUMMARY:IETF Calendaring Working Group Meeting
DTSTART:19970601T210000Z
```

```
DTEND:19970601T220000Z
DTSTAMP:19970602T094000Z
LOCATION:Conference Call
STATUS:CONFIRMED
FOO:BAR
END:VEVENT
END:VCALENDAR
```

"B" responds to indicate that "RRULE" is not supported and that an unrecognized property was encountered.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE:mailto:b@example.com
REQUEST-STATUS:3.0;Invalid Property Name;FOO
UID:guid-1@example.com
SEQUENCE:0
DTSTAMP:19970603T094000Z
END:VEVENT
END:VCALENDAR
```

4.5. Group To-Do Examples

Individual "A" creates a group task in which individuals "A", "B", "C", and "D" will participate. Individual "B" confirms acceptance of the task. Individual "C" declines the task. Individual "D" tentatively accepts the task. The following table illustrates the sequence of messages that would be exchanged between these individuals. Individual "A" then issues a "REQUEST" method to obtain the status of the to-do from each participant. The response indicates the individual "Attendee's" completion status. The table below illustrates the message flow.

Action	Organizer	Attendee
Initiate a to-do request	"A" sends a REQUEST message to "B", "C", and "D".	
Accept the to-do request		"B" sends a REPLY message to "A" with its PARTSTAT parameter set to ACCEPTED.
Decline the to-do request		"C" sends a REPLY message to "A" with its PARTSTAT parameter set to DECLINED.
Tentatively accept the to-do request		"D" sends a REPLY message to "A" with its PARTSTAT parameter set to TENTATIVE.
Check Attendee completion status	"A" sends a REQUEST message to "B" and "D" with current information.	
Attendee indicates percent complete		"B" sends a REPLY message indicating percent complete.
Attendee indicates completion		"D" sends a REPLY message indicating completion.

4.5.1. A VTODO Request

A sample "REQUEST" for a "VTODO" calendar component that "A" sends to "B", "C", and "D".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VTODO
ORGANIZER:mailto:a@example.com
```

```
ATTENDEE;ROLE=CHAIR:mailto:a@example.com
ATTENDEE;RSVP=TRUE:mailto:b@example.com
ATTENDEE;RSVP=TRUE:mailto:c@example.com
ATTENDEE;RSVP=TRUE:mailto:d@example.com
DTSTART:19970701T170000Z
DUE:19970722T170000Z
PRIORITY:1
SUMMARY:Create the requirements document
UID:calsrv.example.com-873970198738777-00@example.com
SEQUENCE:0
DTSTAMP:19970717T200000Z
STATUS:NEEDS-ACTION
END:VTODO
END:VCALENDAR
```

4.5.2. A VTODO Reply

"B" accepts the to-do.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VTODO
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=ACCEPTED:mailto:b@example.com
UID:calsrv.example.com-873970198738777-00@example.com
COMMENT:I'll send you my input by email
SEQUENCE:0
DTSTAMP:19970717T203000Z
REQUEST-STATUS:2.0;Success
END:VTODO
END:VCALENDAR
```

"B" could have declined the "VTODO" or indicated tentative acceptance by setting the "PARTSTAT" property parameter sequence to "DECLINED" or "TENTATIVE", respectively.

4.5.3. A VTODO Request for Updated Status

"A" requests status from all "Attendees".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VTODO
ORGANIZER:mailto:a@example.com
```

```
ATTENDEE;ROLE=CHAIR:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:b@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:d@example.com
UID:calsrv.example.com-873970198738777-00@example.com
SUMMARY:Create the requirements document
PRIORITY:1
SEQUENCE:0
STATUS:IN-PROCESS
DTSTART:19970701T170000Z
DTSTAMP:19970717T230000Z
END:VTODO
END:VCALENDAR
```

4.5.4. A Reply: Percent-Complete

A reply indicating the task being worked on and that "B" is 75% complete with "B's" part of the assignment.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VTODO
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=IN-PROCESS:mailto:b@example.com
PERCENT-COMPLETE:75
UID:calsrv.example.com-873970198738777-00@example.com
DTSTAMP:19970717T233000Z
SEQUENCE:0
END:VTODO
END:VCALENDAR
```

4.5.5. A Reply: Completed

A reply indicating that "D" completed "D's" part of the assignment.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VTODO
ORGANIZER:mailto:a@example.com
ATTENDEE;PARTSTAT=COMPLETED:mailto:d@example.com
UID:calsrv.example.com-873970198738777-00@example.com
DTSTAMP:19970717T233000Z
SEQUENCE:0
END:VTODO
END:VCALENDAR
```

4.5.6. An Updated VTODO Request

"Organizer" "A" resends the "VTODO" calendar component. "A" sets the overall completion for the to-do at 40%.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VTODO
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE;PARTSTAT=ACCEPTED;CUTYPE=INDIVIDUAL:mailto:b@example.com
ATTENDEE;PARTSTAT=COMPLETED;CUTYPE=INDIVIDUAL:mailto:d@example.com
DTSTART:19970701T170000Z
DUE:19970722T170000Z
PRIORITY:1
SUMMARY:Create the requirements document
UID:calsrv.example.com-873970198738777-00@example.com
SEQUENCE:1
DTSTAMP:19970718T100000Z
STATUS:IN-PROCESS
PERCENT-COMPLETE:40
END:VTODO
END:VCALENDAR
```

4.5.7. Recurring VTODOs

The following examples relate to recurring "VTODO" calendar components.

4.5.7.1. Request for a Recurring VTODO

In this example, "A" sends a recurring "VTODO" calendar component to "B" and "D".

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VTODO
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR:mailto:a@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:b@example.com
ATTENDEE;RSVP=TRUE;CUTYPE=INDIVIDUAL:mailto:d@example.com
RRULE:FREQ=MONTHLY;COUNT=10;BYDAY=1FR
DTSTART:19980101T100000Z
DUE:19980103T100000Z
```

```
SUMMARY:Send Status Reports to Area Managers
UID:calsrv.example.com-873970198738777-00@example.com
SEQUENCE:0
DTSTAMP:19970717T200000Z
STATUS:NEEDS-ACTION
PRIORITY:1
END:VTODO
END:VCALENDAR
```

4.5.7.2. Replying to an Instance of a Recurring VTODO

In this example, "B" updates "A" on a single instance of the "VTODO" calendar component.

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REPLY
VERSION:2.0
BEGIN:VTODO
ATTENDEE;PARTSTAT=IN-PROCESS:mailto:b@example.com
PERCENT-COMPLETE:75
UID:calsrv.example.com-873970198738777-00@example.com
DTSTAMP:19970717T233000Z
RECURRENCE-ID:19980101T170000Z
SEQUENCE:1
END:VTODO
END:VCALENDAR
```

4.6. Journal Examples

The iCalendar object below describes a single journal entry for October 2, 1997. The "RELATED-TO" property references the phone conference event for which minutes were taken.

```
BEGIN:VCALENDAR
METHOD:PUBLISH
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VJOURNAL
DTSTART:19971002T200000Z
DTSTAMP:19970717T233100Z
ORGANIZER:mailto:a@example.com
SUMMARY:Phone conference minutes
DESCRIPTION:The editors meeting was held on October 1, 1997.
 Details are in the attached document.
UID:0981234-1234234-2410@example.com
RELATED-TO:0981234-1234234-2402-35@example.com
ATTACH:ftp://ftp.example.com/pub/ed/minutes100197.txt
```

```
END:VJOURNAL
END:VCALENDAR
```

4.7. Other Examples

4.7.1. Event Refresh

Refresh the event with a "UID" property value of "guid-1-12345@example.com":

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REFRESH
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE:mailto:b@example.com
ATTENDEE:mailto:c@example.com
ATTENDEE:mailto:d@example.com
UID:guid-1-12345@example.com
DTSTAMP:19970603T094000
END:VEVENT
END:VCALENDAR
```

4.7.2. Bad RECURRENCE-ID

Component instances are identified by the combination of "UID", "RECURRENCE-ID", and "SEQUENCE". When an "Organizer" sends an iTIP message to an "Attendee", there are three cases in which an instance cannot be found. They are:

1. The component with the referenced "UID" and "RECURRENCE-ID" has been found but the "SEQUENCE" number in the calendar store does not match that of the iTIP message.

2. The component with the referenced "UID" has been found, the "SEQUENCE" numbers match, but the "RECURRENCE-ID" cannot be found.

3. The "UID" and "SEQUENCE" numbers are found but the CUA does not support recurrences.

In case (1), two things can happen. If the "SEQUENCE" number of the "Attendee's" instance is larger than that in the "Organizer's" message, then the "Attendee" is receiving an out-of-sequence message and MUST ignore it. If the "SEQUENCE" number of the "Attendee's" instance is smaller, then the "Organizer" is sending out a newer

version of the component and the "Attendee's" version needs to be updated. Since one or more updates have been missed, the "Attendee" SHOULD send a "REFRESH" message to the "Organizer" to get an updated version of the event.

In case (2), something has gone wrong. Both the "Organizer" and the "Attendee" should have the same instances, but the "Attendee" does not have the referenced instance. In this case, the "Attendee" SHOULD send a "REFRESH" to the "Organizer" to get an updated version of the event.

In case (3), the limitations of the "Attendee's" CUA makes it impossible to match an instance other than the single instance scheduled. In this case, the "Attendee" need not send a "REFRESH" to the "Organizer".

The example below shows a sequence in which an "Attendee" sends a "REFRESH" to the "Organizer".

Action	Organizer	Attendee
Update an instance request	"A" sends REQUEST message to "B".	
Attendee requests refresh because RECURRENCE-ID was not found		"B" sends a REFRESH message to "A".
Refresh the entire event	"A" sends the latest copy of the event to "B"	
Attendee handles the request and updates the instance		"B" updates to the latest copy of the meeting.

Request from "A":

```
BEGIN:VCALENDAR
METHOD:REQUEST
PRODID:-//Example/ExampleCalendarClient//EN
VERSION:2.0
BEGIN:VEVENT
UID:example-12345@example.com
SEQUENCE:3
```

```
RRULE:FREQ=WEEKLY
RDATE;VALUE=PERIOD:19970819T210000Z/199700819T220000Z
ORGANIZER:mailto:a@example.com
ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:mailto:a@example.com
ATTENDEE:mailto:b@example.com
DESCRIPTION:IETF-C&S Conference Call
SUMMARY:IETF Calendaring Working Group Meeting
DTSTART:19970801T210000Z
DTEND:19970801T220000Z
RECURRENCE-ID:19970809T210000Z
DTSTAMP:19970726T083000
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
```

"B" has the event with "UID" property "example-12345@example.com", but "B's" "SEQUENCE" property value is "1" and the event does not have an instance at the specified recurrence time. This means that "B" has missed at least one update and needs a new copy of the event. "B" requests the latest copy of the event with the following refresh message:

```
BEGIN:VCALENDAR
PRODID:-//Example/ExampleCalendarClient//EN
METHOD:REFRESH
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:a@example.com
ATTENDEE:mailto:b@example.com
UID:example-12345@example.com
DTSTAMP:19970603T094000
END:VEVENT
END:VCALENDAR
```

5. Application Protocol Fallbacks

5.1. Partial Implementation

Applications that support this specification are not required to support the entire protocol. The following describes how methods and properties SHOULD "fallback" in applications that do not support the complete protocol. If a method or property is not addressed in this section, it may be ignored.

5.1.1. Event-Related Fallbacks

Method	Fallback
PUBLISH	Required
REQUEST	PUBLISH
REPLY	Required
ADD	Required if recurrences supported; otherwise, reply with a REQUEST-STATUS "2.8; Success, repeating event ignored. Scheduled as a single component", and schedule as a single component.
CANCEL	Required
REFRESH	Required
COUNTER	Reply with "Not Supported".
DECLINECOUNTER	Required if COUNTER is implemented for VEVENTs; otherwise, reply with "Not Supported".

iCalendar Property	Fallback
CALSCALE	Ignore - assume GREGORIAN.
PRODID	Ignore
METHOD	Required as described in the Method list above.
VERSION	Ignore

Event-Related Components	Fallback
VALARM	Reply with "Not Supported".
VTIMEZONE	Required if any DateTime value refers to a time zone.

Component Property	Fallback
ATTACH	Ignore
ATTENDEE	Required if METHOD is REQUEST; otherwise, ignore.
CATEGORIES	Ignore
CLASS	Ignore
COMMENT	Ignore
COMPLETED	Ignore
CONTACT	Ignore
CREATED	Ignore
DESCRIPTION	Ignore
DURATION	Required
DTSTAMP	Required
DTSTART	Required
DTEND	Required
EXDATE	Ignore
GEO	Ignore
LAST-MODIFIED	Ignore
LOCATION	Required
ORGANIZER	Required if METHOD is REQUEST; otherwise, ignore.
PRIORITY	Ignore
RELATED-TO	Ignore
RDATE	Ignore
RRULE	Ignore - assume the first instance occurs on the DTSTART property. If implemented, VTIMEZONE MUST also be implemented.
RECURRENCE-ID	Required if RRULE is implemented; otherwise, ignore.
REQUEST-STATUS	Required
RESOURCES	Ignore
SEQUENCE	Required
STATUS	Ignore
SUMMARY	Ignore
TRANSP	Required if FREEBUSY is implemented; otherwise, ignore.
URL	Ignore
UID	Required
X-	Ignore

5.1.2. Free/Busy-Related Fallbacks

Method	Fallback
PUBLISH	Required if freebusy lookups are supported; otherwise, reply with a REQUEST-STATUS "3.14; Unsupported capability".
REQUEST	Required if freebusy lookups are supported; otherwise, reply with a REQUEST-STATUS "3.14; Unsupported capability".
REPLY	Required if freebusy lookups are supported; otherwise, reply with a REQUEST-STATUS "3.14; Unsupported capability".

iCalendar Property	Fallback
CALSCALE	Ignore - assume GREGORIAN.
PRODID	Ignore
METHOD	Required as described in the Method list above.
VERSION	Ignore

Component Property	Fallback
ATTENDEE	Required if METHOD is REQUEST; otherwise, ignore.
COMMENT	Ignore
CONTACT	Ignore
DTEND	Required
DTSTAMP	Required
DTSTART	Required
DURATION	Ignore
FREEBUSY	Required
ORGANIZER	Required if METHOD is REQUEST; otherwise, ignore.
REQUEST-STATUS	Ignore
UID	Required
URL	Ignore
X-	Ignore

5.1.3. To-Do-Related Fallbacks

Method	Fallback
PUBLISH	Required
REQUEST	PUBLISH
REPLY	Required
ADD	Required if recurrences supported; otherwise, reply with a REQUEST-STATUS "2.8; Success, repeating event ignored. Scheduled as a single component", and schedule as a single component.
CANCEL	Required
REFRESH	Required
COUNTER	Reply with "Not Supported".
DECLINECOUNTER	Required if COUNTER for VTODOs is implemented; otherwise, reply with "Not Supported".

iCalendar Property	Fallback
CALSCALE	Ignore - assume GREGORIAN.
PRODID	Ignore
METHOD	Required as described in the Method list above.
VERSION	Ignore

To-Do-Related Components	Fallback
VALARM	Reply with "Not Supported".
VTIMEZONE	Required if any DateTime value refers to a time zone.

Component Property	Fallback
ATTACH	Ignore
ATTENDEE	Required if METHOD is REQUEST; otherwise, ignore.
CATEGORIES	Ignore
CLASS	Ignore
COMMENT	Ignore
COMPLETED	Required
CONTACT	Ignore
CREATED	Ignore
DESCRIPTION	Required if METHOD is REQUEST; otherwise, ignore.
DUE	Required
DURATION	Required
DTSTAMP	Required
DTSTART	Required
EXDATE	Ignore - reply with "Not Supported".
LAST-MODIFIED	Ignore
LOCATION	Ignore
ORGANIZER	Required if METHOD is REQUEST; otherwise, ignore.
PERCENT-COMPLETE	Ignore
PRIORITY	Required
RECURRENCE-ID	Required if RRULE is implemented; otherwise, ignore.
RELATED-TO	Ignore
REQUEST-STATUS	Ignore
RDATE	Ignore
RRULE	Ignore - assume the first instance occurs on the DTSTART property. If implemented, VTIMEZONE MUST also be implemented.
RESOURCES	Ignore
SEQUENCE	Required
STATUS	Required
SUMMARY	Ignore
URL	Ignore
UID	Required
X-	Ignore

5.1.4. Journal-Related Fallbacks

Method	Fallback
PUBLISH	Implementations MAY ignore the METHOD type. The REQUEST-STATUS "3.14; Unsupported capability" MUST be returned.
ADD	Implementations MAY ignore the METHOD type. The REQUEST-STATUS "3.14; Unsupported capability" MUST be returned.
CANCEL	Implementations MAY ignore the METHOD type. The REQUEST-STATUS "3.14; Unsupported capability" MUST be returned.

iCalendar Property	Fallback
CALSCALE	Ignore - assume GREGORIAN.
PRODID	Ignore
METHOD	Required as described in the Method list above.
VERSION	Ignore

Journal-Related Components	Fallback
VTIMEZONE	Required if any DateTime value refers to a time zone.

Component Property	Fallback
ATTACH	Ignore
ATTENDEE	Ignore
CATEGORIES	Ignore
CLASS	Ignore
COMMENT	Ignore
CONTACT	Ignore
CREATED	Ignore
DESCRIPTION	Ignore
DTSTAMP	Required
DTSTART	Required
EXDATE	Ignore
LAST-MODIFIED	Ignore
ORGANIZER	Ignore
RECURRENCE-ID	Required if RRULE is implemented; otherwise, ignore.
RELATED-TO	Ignore
RDATE	Ignore
RRULE	Ignore - assume the first instance occurs on the DTSTART property. If implemented, VTIMEZONE MUST also be implemented.
SEQUENCE	Required
STATUS	Ignore
SUMMARY	Required
URL	Ignore
UID	Required
X-	Ignore

5.2. Latency Issues

With a store-and-forward transport, it is possible for events to arrive out of sequence. That is, a "CANCEL" method may be received prior to receiving the associated "REQUEST" for the calendar component. This section discusses a few of these scenarios.

5.2.1. Cancellation of an Unknown Calendar Component

When a "CANCEL" method is received before the original "REQUEST" method, the calendar will be unable to correlate the "UID" property of the cancellation with an existing calendar component. It is suggested that messages that cannot be correlated and that also contain non-zero sequence numbers be held and not discarded. Implementations MAY age them out if no other messages arrive with the same "UID" property value and a lower sequence number.

5.2.2. Unexpected Reply from an Unknown Delegate

When an "Attendee" delegates an item to another CU, they MUST send a "REPLY" method to the "Organizer" using the "ATTENDEE" properties to indicate that the request was delegated and to whom. Hence, it is possible for an "Organizer" to receive a "REPLY" from a CU not listed as one of the original "Attendees". The resolution is left to the implementation, but it is expected that the calendaring software will either accept the reply or hold it until the related "REPLY" method is received from the "Delegator". If the version of the "REPLY" method is out of date, the "Organizer" SHOULD treat the message as a "REFRESH" message and update the "Delegate" with the correct version, provided that delegation to that delegate is acceptable.

5.3. Sequence Number

Under some conditions, a CUA may receive requests and replies with the same "SEQUENCE" property value. The "DTSTAMP" property is utilized as a tie-breaker when two items with the same "SEQUENCE" property value are evaluated.

6. Security Considerations

iTIP is an abstract transport protocol that will be bound to a real-time transport, a store-and-forward transport, and perhaps other transports. The transport protocol will be responsible for providing facilities for authentication and encryption using standard Internet mechanisms that are mutually understood between the sender and receiver.

6.1. Security Threats

6.1.1. Spoofing the Organizer

In iTIP, the "Organizer" (or someone working on the "Organizer's" behalf) is the only person authorized to make changes to an existing "VEVENT", "VTODO", or "VJOURNAL" calendar component and republish it or redistribute updates to the "Attendees". An iCalendar object that maliciously changes or cancels an existing "VEVENT", "VTODO", or "VJOURNAL" calendar component may be constructed by someone other than the "Organizer" and republished or sent to the "Attendees".

6.1.2. Spoofing the Attendee

In iTIP, an "Attendee" of a "VEVENT" or "VTODO" calendar component (or someone working on the "Attendee's" behalf) is the only person authorized to update any parameter associated with their "ATTENDEE"

property and send it to the "Organizer". An iCalendar object that maliciously changes the "ATTENDEE" parameters may be constructed by someone other than the real "Attendee" and sent to the "Organizer".

6.1.3. Unauthorized Replacement of the Organizer

There will be circumstances when "Attendees" of an event or to-do decide, using out-of-band mechanisms, that the "Organizer" must be replaced. When the new "Organizer" sends out the updated "VEVENT" or "VTODO", the "Attendee's" CUA will detect that the "Organizer" has been changed, but it has no way of knowing whether or not the change was mutually agreed upon.

6.1.4. Eavesdropping and Data Integrity

The iCalendar object is constructed with human-readable clear text. Any information contained in an iCalendar object may be read and/or changed by unauthorized persons while the object is in transit.

6.1.5. Flooding a Calendar

Implementations could provide a means to automatically incorporate "REQUEST" methods into a calendar. This presents the opportunity for a calendar to be flooded with requests, which effectively blocks all the calendar's free time.

6.1.6. Unauthorized REFRESH Requests

It is possible for an "Organizer" to receive a "REFRESH" request from someone who is not an "Attendee" of an event or to-do. Only "Attendees" of an event or to-do are authorized to receive replies to "REFRESH" requests. Replying to such requests to anyone who is not an "Attendee" may be a security problem.

6.2. Recommendations

For an application where the information is sensitive or critical and the network is subject to a high probability of attack, iTIP transactions SHOULD be encrypted and authenticated. This helps mitigate the threats of spoofing, eavesdropping, and malicious changes in transit.

6.2.1. Securing iTIP transactions

iTIP transport bindings MUST provide a mechanism to enable authentication of the sender's identity as well as privacy and integrity of the data being transmitted. This allows the receiver of a signed iCalendar object to verify the identity of the sender. This

sender may then be correlated to an "ATTENDEE" property in the iCalendar object. If the correlation is made and the sender is authorized to make the requested change or update, then the operation may proceed. It also allows the message to be encrypted to prevent unauthorized reading of the message contents in transit. iTIP transport binding documents describe this process in detail.

6.2.2. Implementation Controls

The threat of unauthorized replacement of the "Organizer" SHOULD be mitigated by a calendar system that uses this protocol by providing controls or alerts that make "Calendar Users" aware of such "Organizer" changes and allowing them to decide whether or not the request should be honored.

The threat of flooding a calendar SHOULD be mitigated by a calendar system that uses this protocol by providing controls that may be used to limit the acceptable sources for iTIP transactions, and perhaps the size of messages and volume of traffic, by source.

The threat of unauthorized "REFRESH" requests SHOULD be mitigated by a calendar system that uses this protocol by providing controls or alerts that allow "Calendar Users" to decide whether or not the request should be honored. An implementation MAY decide to maintain, for audit or historical purposes, "Calendar Users" who were part of an "Attendee" list and who were subsequently uninvited. Similar controls or alerts should be provided when a "REFRESH" request is received from these "Calendar Users" as well.

6.2.3. Access Controls and Filtering

In many environments, there could be restrictions on who is allowed to schedule with whom and who the allowed delegates are for particular "Calendar Users".

iTIP transport bindings SHOULD provide mechanisms for implementing access controls or filtering to ensure iTIP transactions only take place between authorized "Calendar Users". That would include preventing one "Calendar User" from scheduling with another or one "Calendar User" delegating to another.

6.3. Privacy Issues

The "Organizer" might want to keep "Attendees" from knowing which other "Attendees" are participating in an event or to-do. The "Organizer" has the choice of sending single iTIP messages with a full list of "Attendees" or sending iTIP messages to each "Attendee" with only that "Attendee" listed.

7. IANA Considerations

7.1. Registration Template for REQUEST-STATUS Values

This specification updates [RFC5545] by adding a "REQUEST-STATUS" value registry to the iCalendar Elements registry.

A "REQUEST-STATUS" value is defined by completing the following template.

Status Code: Hierarchical, numeric return status code, following the rules defined in Section 3.8.8.3 of [RFC5545].

Status Description: Textual status description. A short but clear description of the error.

Status Exception Data: Textual exception data. A short but clear description of what might appear in this field.

Description: Describe the underlying cause for this status code value.

7.2. Additions to iCalendar METHOD Registry

This document defines the following values for the iCalendar "METHOD" property, using the values template from Section 8.2.6 of [RFC5545]. These should be added to the Methods Registry defined in Section 8.3.12 of [RFC5545]:

7.2.1. METHOD:PUBLISH

Value: PUBLISH

Purpose: Standard iTIP "METHOD" value.

Conformance: Only used with the "METHOD" property.

Examples: See this RFC.

7.2.2. METHOD:REQUEST

Value: REQUEST

Purpose: Standard iTIP "METHOD" value.

Conformance: Only used with the "METHOD" property.

Examples: See this RFC.

7.2.3. METHOD:REPLY

Value: REPLY

Purpose: Standard iTIP "METHOD" value.

Conformance: Only used with the "METHOD" property.

Examples: See this RFC.

7.2.4. METHOD:ADD

Value: ADD

Purpose: Standard iTIP "METHOD" value.

Conformance: Only used with the "METHOD" property.

Examples: See this RFC.

7.2.5. METHOD:CANCEL

Value: CANCEL

Purpose: Standard iTIP "METHOD" value.

Conformance: Only used with the "METHOD" property.

Examples: See this RFC.

7.2.6. METHOD:REFRESH

Value: REFRESH

Purpose: Standard iTIP "METHOD" value.

Conformance: Only used with the "METHOD" property.

Examples: See this RFC.

7.2.7. METHOD:COUNTER

Value: COUNTER

Purpose: Standard iTIP "METHOD" value.

Conformance: Only used with the "METHOD" property.

Examples: See this RFC.

7.2.8. METHOD:DECLINECOUNTER

Value: DECLINECOUNTER

Purpose: Standard iTIP "METHOD" value.

Conformance: Only used with the "METHOD" property.

Examples: See this RFC.

7.3. REQUEST-STATUS Value Registry

New "REQUEST-STATUS" values can be registered using the process described in Section 8.2.1 of [RFC5545].

The following table is to be used to initialize the "REQUEST-STATUS" value registry.

Status Code	Status	Reference
2.0	Current	RFC 5546, Section 3.6.1
2.1	Current	RFC 5546, Section 3.6.2
2.2	Current	RFC 5546, Section 3.6.3
2.3	Current	RFC 5546, Section 3.6.4
2.4	Current	RFC 5546, Section 3.6.5
2.5	Current	RFC 5546, Section 3.6.6
2.6	Current	RFC 5546, Section 3.6.7
2.7	Current	RFC 5546, Section 3.6.8
2.8	Current	RFC 5546, Section 3.6.9
2.9	Current	RFC 5546, Section 3.6.10
2.10	Current	RFC 5546, Section 3.6.11
2.11	Current	RFC 5546, Section 3.6.12
3.0	Current	RFC 5546, Section 3.6.13
3.1	Current	RFC 5546, Section 3.6.14
3.2	Current	RFC 5546, Section 3.6.15
3.3	Current	RFC 5546, Section 3.6.16
3.4	Current	RFC 5546, Section 3.6.17
3.5	Current	RFC 5546, Section 3.6.18
3.6	Current	RFC 5546, Section 3.6.19
3.7	Current	RFC 5546, Section 3.6.20
3.8	Current	RFC 5546, Section 3.6.21
3.9	Current	RFC 5546, Section 3.6.22
3.10	Current	RFC 5546, Section 3.6.23
3.11	Current	RFC 5546, Section 3.6.24
3.12	Current	RFC 5546, Section 3.6.25
3.13	Current	RFC 5546, Section 3.6.26
3.14	Current	RFC 5546, Section 3.6.27
4.0	Current	RFC 5546, Section 3.6.28
5.0	Current	RFC 5546, Section 3.6.29
5.1	Current	RFC 5546, Section 3.6.30
5.2	Current	RFC 5546, Section 3.6.31
5.3	Current	RFC 5546, Section 3.6.32

8. Acknowledgments

This is an update to the original iTIP document authored by S. Silverberg, S. Mansour, F. Dawson, and R. Hopson.

This revision is the product of the Calsify IETF Working Group, and several participants have made important contributions to this specification, including Oliver Block, Bernard Desruisseaux, Mike Douglass, Tim Hare, Ciny Joy, Bruce Kahn, Reinhold Kainhofer, Eliot Lear, Jonathan Lennox, Andy Mabbett, Aki Niemi, John W. Noerenberg II, Robert Ransdell, and Caleb Richardson.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2368] Hoffman, P., Masinter, L., and J. Zawinski, "The mailto URL scheme", RFC 2368, July 1998.

[RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, September 2009.

9.2. Informative References

[iMIP] Melnikov, A., Ed., "iCalendar Message-Based Interoperability Protocol (iMIP)", Work in Progress, October 2009.

Appendix A. Differences from RFC 2446

A.1. Changed Restrictions

This specification now defines an allowed combination of "REQUEST-STATUS" codes when multiple iCalendar components are included in an iTIP message.

This specification now restricts "RECURRENCE-ID" to only a single occurrence in any one iCalendar component in an iTIP message, as required by [RFC5545].

Changed the "RECURRENCE-ID" entry in the component restriction table to "0 or 1" from "0+", to fall in line with [RFC5545].

Changed the "FREEBUSY" entry in the "VFREEBUSY", "PUBLISH", and "REPLY" restriction tables to "0+" from "1+", to fall in line with [RFC5545].

Changed the "FREEBUSY" description in the "VFREEBUSY" and "REPLY" restriction tables to indicate that different "FBTYPE" ranges MUST NOT overlap.

Changed the "TZNAME" entry in the "VTIMEZONE" restriction table to "0+" from "0 or 1", to fall in line with [RFC5545].

Changed the "COMMENT" entry in the component restriction tables to "0+" from "0 or 1", to fall in line with [RFC5545].

Added the "ATTENDEE" entry in the "VALARM" restriction table to match the email alarm type in [RFC5545].

Changed the "CATEGORIES" entry in the component restriction tables to "0+" from "0 or 1", to fall in line with [RFC5545].

Changed the "RESOURCES" entry in the component restriction tables to "0+" from "0 or 1", to fall in line with [RFC5545].

Changed the "CONTACT" entry in the "VFREEBUSY" restriction table to "0 or 1" from "0+", to fall in line with [RFC5545].

Changed the "UID" entry in the "VFREEBUSY" and "PUBLISH" restriction tables to "1" from "0", to fall in line with [RFC5545].

Added the "COMPLETED" entry in the "VTODO" restriction tables to fall in line with [RFC5545].

Added the "REQUEST-STATUS" entry in the "VJOURNAL" restriction tables to fall in line with [RFC5545].

A.2. Deprecated Features

The "EXRULE" property was removed in [RFC5545] and references to that have been removed in this document too.

The "PROCEDURE" value for the "ACTION" property was removed in [RFC5545] and references to that have been removed in this document too.

The "THISANDPRIOR" option for the "RANGE" parameter was removed in [RFC5545] and references to that have been removed in this document too.

Author's Address

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

EMail: cyrus@daboo.name
URI: http://www.apple.com/