Network Working Group
Internet-Draft
Expires: May 21, 2002

S. Mansour
AOL/Netscape
D. Royer
INET-Consulting LLC
G. Babics
Steltor
P. Hill
Massachusetts Institute of
Technology

November 20, 2001

Calendar Access Protocol (CAP)
draft-ietf-calsch-cap-06

Status of this Memo

Copyright Notice

Abstract

The Calendar Access Protocol (CAP) is an Internet protocol that permits a Calendar User (CU) to utilize a Calendar User Agent (CUA) to access an [iCAL] based Calendar Store (CS). This memo defines the CAP specification.

The CAP definition is based on requirements identified by the Internet Engineering Task Force (IETF) Calendaring and Scheduling (CALSCH) Working Group. More information about the IETF CALSCH Working Group activities can be found on the IMC web site at http://www.imc.org/ietf-calendar and at the IETF web site at http://www.ietf.org/html.charters/calsch-charter.html[1]. Refer to the references within this memo for further information on how to access these various documents.

Table of Contents

1. Introduction

This document specifies how a Calendar User Agent (CUA) interacts with a Calendar Store (CS) to manage calendar information. In particular, it specifies how to query, create, modify, and delete iCalendar components (e.g., events, to-dos, or daily journal entries). It further specifies how to search for available busy time information.

CAP is specified as a BEEP "profile". As such many aspects of the protocol (e.g., authentication and privacy) are provided within the BEEP core [BEEP]. The protocol data units leverage the standard iCalendar format [iCAL] to convey calendar related information.

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 protocol defined by this memo. Calendar components defined by [iCAL] 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 defined by [iTIP], are referred to with capitalized, quoted-strings of text. For example, "REPLY" refers to the method for replying to a "REQUEST".

Calendar commands are referred by lower-case, quotes-strings of text, followed by the word "command". For example, "create" command refers to the command for creating a calendar entry, "search" command refers to the command for reading calendar components.

Properties defined by this memo 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 memo are referred to with capitalized, quoted-strings of text, followed by the word "parameter". For example, "PARTSTAT" parameter refers to the iCalendar property parameter used to specify the participation status of an attendee. Enumerated values defined by this memo 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 memos that, along with this one, describe the Internet calendaring and scheduling standards. These documents are:

[iCAL] (RFC2445) which specifies the objects, data types, properties and property parameters used in the protocols, along with the methods for representing and encoding them,

[iTIP] (RFC2446) which specifies an interoperability protocol for scheduling between different implementations. The related documents are:

[iMIP] (RFC2447) which specifies an Internet email binding for [iTIP].

[GUIDE] (draft/rfc...) which is a guide to implementers and describes the elements of a calendaring system, how they interact with each other, how they interact with end users, and how the standards and protocols are used.

This memo does not attempt to repeat the specification of concepts and definitions from these other memos. Where possible, references are made to the memo that provides for the specification of these concepts and definitions.

1.3 Definitions

Booked

An entry in a calendar has one of two conceptual states. It is scheduled or it is booked. A scheduled entry has been stored in the calendar store but has not been acted on by a calendar user (CU) or calendar user agent (CUA). A scheduled entry contains a METHOD property set to an [iTIP] method. A booked entry is a component does not have a METHOD property.

Calendar

A collection of logically related objects or entities each of which may be associated with a calendar date and possibly time of day. These entities can include other calendar properties or

calendar components. In addition, a calendar might be hierarchically related to other sub-calendars. A calendar is identified by its unique calendar identifier. The [iCAL] defines calendar properties, calendar components and component properties that make up the content of a calendar.

Calendar Access Protocol (CAP)

The standard Internet protocol that permits a Calendar User Agent to access and manipulate calendars residing on a Calendar Store.

Calendar Access Rights (CAR)

The mechanism for specifying the CAP operations ("ACTION") that a particular calendar user ("UPN") are granted or denied permission to perform on a given calendar object ("OBJECT"). The calendar access rights are specified with the "VCAR" calendar components within a CS and calendar.

Calendar Component

An object within a calendar or a calendar store (CS). Some types of calendar components include calendars, events, to-dos, journals, alarms, time zones and freebusy data. A calendar component consists of component properties and possibly other sub-components. For example, an event may contain an alarm component.

Calendar Component Properties

An attribute of a particular calendar component. Some calendar component properties are applicable to different types of calendar components. For example, DTSTART is applicable to VEVENT, VTODO, VJOURNAL calendar components. Other calendar components are applicable only to an individual type of calendar component. For example, TZURL is only applicable to VTIMEZONE calendar components.

Calendar Identifier (CalID)

A globally unique identifier associated with a calendar. Calendars reside within a CS. See Qualified Calendar Identifier and Relative Calendar Identifier.

Calendar Policy

A CAP operational restriction on the access or manipulation of a calendar. For example, "events MUST be scheduled in unit intervals of one hour".

Calendar Property

An attribute of a calendar (VAGENDA). The attribute applies to the calendar, as a whole. For example, CALSCALE specifies the calendar scale (e.g., GREGORIAN) for the whole calendar.

Calendar Service

An implementation of a Calendar Store that manages one or more calendars.

Calendar Store (CS)

The data and service model definition for a Calendar Service.

Calendar Store Identifier (CSID)

The globally unique identifier for an individual CS. A CSID consists of the host and port portions of a "Common Internet Scheme Syntax" part of a URL, as defined by [URL].

Calendar Store Components

Components maintained in a CS specify a grouping of calendar store-wide information.

Calendar Store Properties

Properties maintained in a Calendar Store calendar store-wide information.

Calendar User (CU)

An entity (often biological) that uses a calendaring system.

Calendar User Agent (CUA)

The CUA is the client application that a CU utilizes to access and manipulate a calendar.

CAP Session

An open communication channel between a CUA and a Calendar Service.

Delegate

A calendar user (sometimes called the delegatee) who has been

assigned participation in a scheduled calendar component (e.g., VEVENT) by one of the attendees in the scheduled calendar component (sometimes called the delegator). An example of a delegate is a team member told to go to a particular meeting.

Designate

A calendar user who is authorized to act on behalf of another calendar user. An example of a designate is an assistant.

Fan Out

The calendaring and scheduling process by which a calendar operation on one calendar is also performed on every other calendar specified in the operation.

Hierarchical Calendars

A CS feature where a calendar has a hierarchical relationship with another calendar in the CS. The top-most calendars in the hierarchical relationship have the CS as their parent. There may be multiple top-most calendars in a given CS. Within a given hierarchical relationship, all sub-calendars have a calendar with a "parent" relationship. In addition, sub-calendars may have a relationship with another calendar that has a "child" relationship. The hierarchical calendar feature is not a storage relationship of the calendars within the CS. Instead it is a feature that relates access control rights to calendar content between different calendars in the CS. The hierarchical relationship of a calendar is specified in the "PARENT" and "CHILDREN" calendar properties.

Overlapped Booking

A policy which indicates whether or not OPAQUE events can overlap one another. When the policy is applied to a calendar it indicates whether or not the time span of any entry (VEVENT, VTODO, ...) in the calendar can overlap the time span of any other entry in the same calendar. When applied to an individual entry, it indicates whether or not any other entry's time span can overlap that individual entry.

Owner

One or more CUs or UGs that have "OWNER" calendar access rights for a calendar. The owner is specified in the "OWNER" calendar property.

Qualified Calendar Identifier (Qualified CalID)

A CalID where both the <scheme> and <csid> are present.

Realm

A collection of calendar user accounts, identified by a string. The name of the Realm is only used in UPNs. In order to avoid namespace conflict, the Realm SHOULD be postfixed with an appropriate DNS domain name. (e.g., the foobar Realm could be called foobar.example.com).

Relative Calendar Identifier (Relative CalID)

An identifier for an individual calendar in a calendar store. It is unique within a calendar store. It is recommended to be globally unique. A Relative CalID consists of the portion of the "scheme part" of a Qualified CalID following the Calendar Store Identifier. This is the same as the "URL path" of the "Common Internet Scheme Syntax" portion of a URL, as defined by [URL].

Session Identity

A UPN associated with a CAP session. A session gains an identity after successful authentication. The identity is used in combination with CAR to determine access to data in the CS.

Sub-calendars

Calendars that have a "child" hierarchical relationship with another calendar, its "parent".

User Group (UG)

A collection of Calendar Users and/or User Groups. These groups are expanded by the CS and may reside either locally or in an external database or directory. The group membership may be fixed or dynamic over time.

Username

A name which denotes a Calendar User within a Realm. This is part of a UPN.

User Principal Name (UPN)

A unique identifier that denotes a CU or a group of CU. A UPN is a RFC 822 compliant email address, with exceptions listed below,

and in most cases it is deliverable to the CU. In some cases it is identical to the CU's well known email address. A CU's UPN MUST never be an e-mail address that is deliverable to a different person as there is no requirement that a person's UPN must be his e-mail address. It consists of a Realm in the form of a valid, and unique, DNS domain name and a unique Username. In it's simplest form it looks like "user@example.com".

In certain cases a UPN will not be RFC 822 compliant. When anonymous authentication is used, or anonymous authorization is being defined, the special UPN "@" will be used. When authentication must be used, but unique identity must be obscured, a UPN of the form @DNS-domain-name may be used. For example, "@example.com". Usage of these special cases is further discussed in the authentication and authorization sections of this document.

2. CAP Design

2.1 System Model

The system model describes the high level components of a calendar system and how they interact with each other.

CAP is used by a "Calendar User Agent" (CUA) to send commands to and receive responses from a "Calendar Service".

The CUA prepares a [MIME] encapsulated command, sends it to the CS, and receives a [MIME] encapsulated response. The calendaring related information within these messages are represented by iCalendar objects.

There are two distinct protocols in operation to accomplish this exchange. [BEEP] is used to move these encapsulations between a CUA and a CS. The CAP profile defines the content and semantics of the messages sent between the CUA and the Calendar Service.

2.2 Calendar Store Object Model

The conceptual model for a calendar store is shown below. The calendar store contains VCARs, VQUERYs, VTIMEZONEs, VAGENDAs and calendar store properties.

Calendars (VAGENDAs) contain VEVENTs, VTODOs, VJOURNALs, VCARs, VTIMEZONEs, VQUERYs and calendar properties. Calendars may also contain other calendars (VAGENDAs).

```
   Calendar Store
     |
     +-- VCARs
     +-- VQUERYs
     +-- VTIMEZONEs
     +-- VAGENDA
     |     |
     |     +--VEVENTs
     |     |    |
     |     |    +--VALARMs
     |     +--VTODOs
     |     |    |
     |     |    +--VALARMs
     |     +--VJOURNALs
     |     +--VCARs
     |     +--VTIMEZONEs
     |     +--VQUERYs
     |     +--VAGENDAs
     |     |    |
     |     |    +--VEVENTs
     |     |    |    |
     |     |    |    +--VALARMs
     |     |    +--VTODOs
     |     |    |    |
     |     |    |    +--VALARMs
     |     |    +--VJOURNALs
     |     |    +--VCARs
     |     |    +--VTIMEZONEs
     |     |    +--VQUERYs
     |     |    +--VFREEBUSY
     |     |    +--VAGENDAs
     |     |    |   |
     |     |    |   ...
```

Calendars within a Calendar Store are identified by their Relative CALID.

In this model, VSCHEDULE is a set of scheduling messages that have not yet been applied to the calendar. Components in VSCHEDULE are discussed in more detail below.

2.3 Protocol Model

The commands listed below are used to manipulate the data on the calendar store. Their usage and semantics are defined in Section 6.

CAP Commands

Command	Description
create	Create a new calendar component.
delete	Delete calendar components.
generate-uid	Generate one or more unique ids.
get-capability	Query the capabilities of the CS.
identify	Set a new identity for calendar access.
modify	Modify calendar components.
move	Move calendar components to another container.
noop	Do nothing.
schedule	Add an [iTIP] object to the VSCHEDULE set.
search	Search for calendar components.

2.4 Security Model

2.4.1 Calendar User and UPNs

A Calendar User (CU) is an entity that can be authenticated. It is represented in CAP as a UPN, which is the subject of access rights. The UPN representation is independent of the authentication mechanism used during a particular CUA/CS interaction. This is because UPNs are used within VCARs. If the UPN were dependent on the authentication mechanism, a VCAR could not be consistently evaluated. A CU may use one mechanism while using one CUA but the same CU may use a different authentication mechanism when using a different CUA, or while connecting from a different location.

The user may also have multiple UPNs for various purposes.

Note that the immutability of the user's UPN may be achieved by using SASL's authorization identity feature. (The transmitted authorization identity may be different than the identity in the client's authentication credentials.) [SASL, section 3]. This also permits a CU to authenticate using their own credentials, yet request the access privileges of the identity for which they are proxying SASL. Also, the form of authentication identity supplied by a service like TLS may not correspond to the UPNs used to express a server's access rights, requiring a server specific mapping to be done. The method by which a server determines a UPN, based on the authentication credentials supplied by a client, is implementation specific.

2.4.1.1 UPNs and Certificates

When using X.509 certificates for purposes of CAP authentication, the UPN should appear in the certificate. Unfortunately there is no

single correct guideline for which field should contain the UPN.

From RFC-2459, section 4.1.2.6 (Subject):

> If subject naming information is present only in the subjectAlt-Name extension (e.g., a key bound only to an email address or URI), then the subject name MUST be an empty sequence and the subjectAltName extension MUST be critical.
>
> Implementations of this specification MAY use these comparison rules to process unfamiliar attribute types (i.e., for name chaining). This allows implementations to process certificates with unfamiliar attributes in the subject name.
>
> In addition, legacy implementations exist where an RFC 822 name is embedded in the subject distinguished name as an EmailAddress attribute. The attribute value for EmailAddress is of type IA5String to permit inclusion of the character '@', which is not part of the PrintableString character set. EmailAddress attribute values are not case sensitive (e.g., "fanfeedback@redsox.com" is the same as "FANFEEDBACK@REDSOX.COM").
>
> Conforming implementations generating new certificates with electronic mail addresses MUST use the rfc822Name in the subject alternative name field (see sec. 4.2.1.7 of [RFC 2459]) to describe such identities. Simultaneous inclusion of the EmailAddress attribute in the subject distinguished name to support legacy implementations is deprecated but permitted.

Since no single method of including the UPN in the certificate will work in all cases, CAP implementations MUST support the ability to configure what the mapping will be by the CS administrator. Implementations MAY support multiple mapping definitions, for example, the UPN may be found in either the subject alternative name field, or the UPN may be embedded in the subject distinguished name as an EmailAddress attribute.

Note: If a CS or CUA is validating data received via iMIP, if the "ORGANIZER" or "ATTENDEE" property said (e.g.) "ATTENDEE;CN=Joe Random User:MAILTO:juser@example.com" then the email address should be checked against the UPN, and the CN should also be checked. This is so the "ATTENDEE" property cannot be changed to something misleading like "ATTENDEE;CN=Joe Rictus User:MAILTO:juser@example.com" and have it pass validation. This validation will also defeat other attempts at confusion.

2.4.1.2 Anonymous Users and Authentication

Anonymous access is often desirable. For example an organization may publish calendar information that does not require any access control for viewing or login. Conversely, a user may wish to view unrestricted calendar information without revealing their identity.

2.4.1.3 User Groups

A User Group is used to represent a collection of CUs or other UGs that can be referenced in VCARs. A UG is represented in CAP as a UPN. The CUA cannot distinguish between a UPN that represents a CU or a UG.

UGs are expanded as necessary by the CS. The CS MAY expand a UG (including nested UGs) to obtain a list of unique CUs. Duplicate UPNs are filtered during expansion.

The CS should not preserve UG expansions across operations. A UG may reference a static list of members, or it may represent a dynamic list. Each operation SHOULD generate its own expansion in order to recognize changes to UG membership.

CAP does not define commands or methods for managing UGs.

2.4.2 Access Rights - Summary

Access rights are used to grant or deny access to a calendar for a CU. CAP defines a new component type called a Calendar Access Right (VCAR). Specifically, a VCAR grants, or denies, UPNs the right to read and write components, properties, and parameters on calendars within a CS.

The VCAR model does not put any restriction on the sequence in which the object and access rights are created. That is, an event associated with a particular VCAR might be created before or after the actual VCAR is defined. In addition, the VCAR and VEVENT definition might be created in the same iCalendar object and passed together in a single command.

All rights MUST be denied unless specifically granted; individual VCARs MUST be specifically granted to an authenticated CU.

The access for a particular UPN is the union of all grants for that UPN minus the union of its denies.

2.4.2.1 Calendar Access Right (VCAR)

Access rights within CAP are specified with the "VCAR" calendar component, "RIGHTS" value type and the "GRANT", "DENY" and "CARID" component properties.

Properties within an iCalendar object are unordered. This also is the case for the "GRANT", "DENY" and "CARID" properties. Likewise, there is no implied ordering required for components of a "RIGHTS" value type other than that specified by the ABNF. [EDITOR'S NOTE, this requires a lot of review. We think that this paragraph may be incorrect.]

For details on the VCAR syntax please see section <forward ref>

2.4.2.2 Decreed VCARs

A CS MAY choose to implement and allow persistent immutable VCARs, that are configured by the CS administrator, which apply to all calendars on the server.

When a user attempts to modify or override a decreed VCAR an error will be returned, indicating that the user has insufficient authorization to perform the operation.

The CAP protocol does not define the semantics used to initially create a decreed VCAR. This administrative task is outside the scope of the CAP protocol.

For example an implementation or a CS administrator may wish to define a VCAR that will always allow the calendar owners to have full access to their own calendars. The GRANT property allows the OWNERs all (OBJECT=*) access to their own calendar objects. The DENY property disallows anyone (UPN=*) from being able to delete or modify this VCAR.

```
BEGIN:VCAR
CARID:Users Default Access
GRANT:UPN=OWNER;OBJECT=*;OBJECT=OBJECT=METHOD;VALUE=*
DENY:UPN=*;OBJECT=VCAR;OBJECT=CARID;
 VALUE="Users Default Access"
 ;OBJECT=METHOD,VALUE=DELETE,MODIFY
END:VCAR
```

Decreed VCARs MUST be readable by the calendar owner in standard VCAR format.

2.4.3 Inheritance

Calendars inherit VCARs from their parent calendar. Calendars whose parent is the Calendar Store inherit VCARs from the Calendar Store.

VCARs specified in a calendar or a sub-calendar override all inherited VCARs.

2.4.4 CAP Session Identity

A BEEP session has an associated set of authentication credentials, from which is derived a UPN. This UPN is the identity of the CAP session, and is used to determine access rights for the session.

The CUA may change the identity of a CAP session by calling the "identify" command. The Calendar Service only permits the operation if the session's authentication credentials are good for the requested identity. The method of checking this permission is implementation dependent, but may be thought of as a mapping from authentication credentials to UPNs. The "identify" command allows a single set of authentication credentials to choose from multiple identities, and allows multiple sets of authentication credentials to assume the same identity.

For anonymous access the identity of the session is "@", a UPN with a null Username and null Realm. A UPN with a null Username, but non-null Realm, such as "@foo.com" may be used to mean any identity from that Realm, which is useful to grant access rights to all users in a given Realm. A UPN with a non-null Username and null Realm, such as "bob@" could be a security risk and MUST NOT be used.

Since the UPN includes Realm information it may be used to govern calendar store access rights across Realms. However, governing access rights across Realms is only useful if login access is available. This could be done through a trusted server relationship or a temporary account.

The "identify" command provides for a weak group implementation. By allowing multiple sets of authentication credentials belonging to different users to identify as the same UPN, that UPN essentially identifies a group of people, and may be used for group calendar ownership, or the granting of access rights to a group.

2.5 Roles

CAP defines methods for managing [iCAL] objects in a Calendar Store and exchanging [iCAL] objects for the purposes of group calendaring and scheduling between "Calendar Users" (CUs) or "User Groups" (UGs).

There are two distinct roles taken on by CUs in CAP. The CU who creates an initial event or to-do and invites other CUs as attendees takes on the role of "Organizer". The CUs asked to participate in the event or to-do take on the role of "Attendee". Note that "role" is also a descriptive parameter to the "ATTENDEE" property. Its use is to convey descriptive context to an "Attendee" such as "chair", "REQ-PARTICIPANT" or "NON-PARTICIPANT" and has nothing to do with the scheduling workflow.

2.6 Calendar Addresses

Calendar addresses are URIs that are modeled after URLs [URL]. CAP uses the following forms of URI.

```
[[<scheme>]://<csid>[:<port>]/]<relativeCALID>
```

where:

<scheme> is "cap", the protocol described in this memo.

<csid> is the Calendar Store ID. It is the network address of the computer on which the CAP server is running.

<port> is optional. The port must be present in the URL if the CAP server does not listen on the default port number.

<relativeCALID> is an identifier that uniquely identifies the calendar on a particular calendar store. There is no implied structure in a Relative CALID. It is an arbitrary string of printable 7 bit ASCII characters. It may refer to the calendar of a user or of a resource such as a conference room. It MUST be unique within the calendar store. It is recommended that the Relative CALID be globally unique.

If the <scheme> and <csid> are present the calendar address is said to be "qualified". Senders are required to supply the <relativeCALID> portion of the address. A qualified calendar address is required when the <csid> of the target calendar address differs from that of the CAP server receiving the command.

Examples of CAP URIs:

```
cap://calendar.example.com/user1
://calendar.example.com/user1
user1
cap://calendar.example.com/conferenceRoomA
cap://calendar.example.com/89798-098-zytytasd
```

For a user currently authenticated to a CAP server on calendar.example.com, the first three addresses refer to the same calendar.

2.7 Extensions to iCalendar

In mapping the calendar query feature, and access rights onto the iCalendar format, several extended iCalendar properties and components are defined by this memo.

The search operation makes use of a new component, called VQUERY. The component consists of a set of new properties: QUERY, EXPAND and QUERYNAME, that define a search filter. VQUERY is used by the following CAP commands: "search", "move", "modify" and "delete".

Access rights are specified in the new iCalendar VCAR component.

Calendar are specified by the new VAGENDA component.

2.8 Relationship of RFC 2446 (ITIP) to CAP

[iTIP] describes scheduling methods which result in indirect manipulation of calendar components. In CAP, the "schedule" command is used to submit scheduling requests. Other CAP commands such as "create", "delete", "modify" and "move" provide direct manipulation of calendar components. In the CAP calendar store model, scheduling messages are conceptually kept separate from other calendar components. This is modeled with the VSCHEDULE set. Note that this is a conceptual model, the actual storage details are left to implementations.

When scheduling is used, the METHOD is saved along with components. A scheduled component becomes a booked component when its METHOD property is removed. For example, a component whose METHOD is "REQUEST" is scheduled. The component becomes booked when the METHOD is removed.

Several scheduled entries can be in the CS for the same UID. They are consolidated when booked, or they are removed from the CS.

For example, if you were on vacation, you could have a REQUEST to attend a meeting and several updates to that meeting. Your CUA would have to "search" them out of the CS using CAP, process them, determine what the final state of the object from a possible combination of user input and programmed logic. Then the CUA would instruct the CS to "create" a new booked entry or "modify" an existing entry. Finally, the CUA can do a "delete" of all of these now old scheduling requests in the CS. See [iTIP] for details on

resolving multiple [iTIP] scheduling entries.

3. Protocol Framework

CAP uses the BEEP application protocol kernel mapped onto TCP (refer to [BEEP] and [BEEPTCP] for more information). The default port that the Calendar Service listens for connections on is port 5229.

3.1 BEEP Exchange Styles

[BEEP] defines three styles of message exchange:

MSG/ANS,ANS,...,NUL: for one-to-many exchanges.

MSG/RPY: for one-to-one exchanges.

MSG/ERR: for requests the cannot be processed due to an error.

A CAP request, targeted at more than one containers, MUST use a one-to-many exchange, with a distinct answer associated with each target. CAP request targeted at a single container MAY use a one-to-one exchange or a one-to-many exchange. "MSG/ERR" MAY only be used when an error condition prevents the execution of the request on all the targeted calendars.

3.2 Use of XML, MIME and iCalendar

Each BEEP payload exchanged via CAP consists of an XML document and possibly an arbitrary MIME content. The XML document defines the action to be performed. When needed, the calendaring related data is included in a related MIME part containing an iCalendar object.

If only an XML document is sent in the BEEP payload, then the mapping to a BEEP payload is straight-forward, e.g.,

```
C: MSG 1 2 . 432 62
C: Content-Type: application/beep+xml
C:
C: <generate-uid num=10/>
C: END
```

Otherwise, arbitrary MIME content is included in the BEEP payload by using a "multipart/related" (see [RFC 3087]), identified using a "cid" URL (see [RFC 2392]), and the XML control document occurs as the starting body part, e.g.,

```
C: MSG 1 3 . 1023 951
C: Content-Type: multipart/related; boundary="boundary-asdf123";
C:               start="<1@cal.example.com>";
C:               type="application/beep+xml"
```

```
   C:
   C: --boundary-asdf123
   C: Content-Type: application/beep+xml
   C: Content-ID: <1@cal.example.com>
   C:
   C:
   C: <schedule id="abcd12346">
   C:   <target relcalid="john-relcalid"/>
   C:   <target relcalid="bob-relcalid"/>
   C:   <data content="cid:2@cal.example.com"/>
   C: </schedule>
   C: --boundary-asdf123
   C: Content-Type: text/calendar
   C: Content-ID: 2@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: METHOD:REQUEST
   C: BEGIN:VEVENT
   C: UID:abcd12345
   C: ORGAGNIZER:cap://cal.example.com/mary-relcalid
   C: ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.example.com/mary-relcalid
   C: ATTENDEE;PARTSTAT=NEEDS-ACTION;RSVP=TRUE:cap://cal.example.com/john-
relcalid
   C: ATTENDEE;PARTSTAT=NEEDS-ACTION;RSVP=TRUE:cap://cal.example.com/bob-
relcalid
   C: DTSTART:20010920T180000Z
   C: DTEND:20010920T190000Z
   C: SUMMARY:Mary invites John and Robert
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-asdf123--
   C: END
```

The MIME content-type "application/beep+xml" is defined in Section 6.4 of [BEEP].

3.3 Bounded Latency

A CUA can associate a maximum latency time to a command with the "max-time" element. If the CS is unable to complete the request in the specified amount of time, then the server sends a "timeout" request to which the CUA MUST respond with a "abort" or "continue" reply.

Upon receiving an "abort" reply, the CS MUST terminate the command in progress and return a request-status code 2.0.3. When receiving a "continue" reply the server resumes its work in progress. Note that a new latency time MAY be included in a "continue" reply.

The timeout element takes two arguments "latency" and "action". The "latency" argument MUST be set to the maximum latency time in seconds. The "action" argument accepts the following values: "ask" and "abort". If the maximum latency time is exceeded and the "action" argument is set to "ask", then CS MUST send a "timeout" message to inform the CUA, otherwise if the argument "action" is set to "abort" the CS can directly terminate the request and return a request-status code 2.0.3.

Example:

In this example bill@cal.example.com attempts to read a calendar but the latency time he supplies is not sufficient for the server to complete the command.

```
C: MSG 1 4 . 2043 680
C: Content-Type: multipart/related; boundary="boundary-zxy123";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-zxy123
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <search id="xyz12346">
C:   <max-time latency=3 action=ask/>
C:   <select>
C:     <source relcalid='opaqueid101'/>
C:     <data content="cid:2@cal.example.com"/>
C:   </select>
C: </search>
C: --boundary-zxy123
C: Content-Type: text/calendar
C: Content-ID: 2@cal.example.com
C:
C: BEGIN:VCALENDAR
C: BEGIN:VQUERY
C: QUERY:SELECT DTSTART,DTEND,SUMMARY,UID FROM VEVENT
C:   WHERE DTEND >= '19990714T080000Z' AND
C:   DTSTART <= '19990715T080000Z'
C: END:VQUERY
C: END:VCALENDAR
C: --boundary-zxy123--
C: END
# After 3 seconds
S: MSG 1 2 . 102 64
S: Content-Type: application/beep+xml
S:
S: <timeout id="xyz12346"/>
```

```
S: END
```

If Bill wants to continue and give the server more time he would issue a "continue" reply:

```
C: RPY 1 2 . 166 113
C: Content-Type: application/beep+xml
C:
C: <continue id="xyz12346">
C:   <max-time latency=3 action=ask/>
C: </continue>
C: END
```

If Bill wants to abort the command and not wait any further he would issue an "abort" reply:

```
C: RPY 1 2 . 166 62
C: Content-Type: application/beep+xml
C:
C: <abort id="xyz12346"/>
C: END
S: RPY 1 4 . 2723 114
S:
S: <result>
S:    <request-status code="2.0.3">
S:       Request Aborted by the CUA.
S:    </request-status>
S: </result>
S: END
```

4. Formal Command Syntax

4.1 Searching and Filtering

This section describes CAP's selecting and filtering entities within a calendar store. It is based on the Standard Query Language (SQL) defined in [SQL].

4.1.1 Grammar For Search Mechanism

```
    search     = "BEGIN:VQUERY" CRLF
                 [expand] querycomp
                 "END:VQUERY" CRLF

    expand     = "EXPAND" ":" ( "TRUE" / "FALSE") CRLF
               # the default is EXPAND:FALSE

    comp-name  = "VEVENT" / "VTODO" / "VJOURNAL"
            / "VTIMEZONE" / "VALARM" / "VFREEBUSY" / "VAGENDA"
               / "VCAR" / iana-name / x-name

    querycomp  = ( query ) / ( queryname query ) / queryname

    queryname  = "QUERYNAME:" text

    query      = "QUERY:" ( query-min / query-92 )
    #
    # NOTE: query-min MUST be implemented in CSs.
    #
    #   query-92 is ONLY used if CAPABILITY returns SQL-92
    #   as the QUERYLEVEL value or if QUERYLEVEL is not
    #    specified.
    #

    query-min      = capselect-min

    capselect-min  = "SELECT" capmin-cols "FROM" capmin-comps
                     "WHERE" capmin-cmp



    capmin-col     = # Any property name found in any of the
                     components.

    capmin-cols    = ( capmin-col / capmin-col ","
                     capmin-cols )
```

```
    capmin-comps   = ( comp-name / comp-name ","
                     compmin-comps )



    capmin-cmp     = ( colname capmin-cmp-rhs
                   / colname capmin-cmp-rhs
                     capmin-logical capmin-cmp )

    capmin-cmp-rhs = ( capmin-oper colvalue
                   /   "IS" ["NOT"] "NULL" )



    colname        = ( # Any valid component property name.
                   / "*" )



    cmpmin-oper    = ( " = "  / " != " / " < " / " > " / " <= "
                   / " >= " )

    capmin-logical = ( " AND " / " OR " )

    query-92       = capselect-92 capfrom-92 capwhere-92
                     caporderby-92

    capselect-92   = # Any valid [SQL] string that goes into
                     a SELECT clause.

    capfrom-92     = # Like capmin-comps except embedded spaces
                     # are allowed between commas - per [SQL].

    capwhere-92    = # Any valid [SQL] string that goes into a
                     # WHERE clause.

    caporderby-92  = # Any valid [SQL] string that goes into a
                     # ORDERBY clause.
```

4.1.2 SQL-MIN notes

(1) No inlined spaces are allowed if not in the grammar above.

(2) Note that cmpmin-oper and capmin-logical elements are surrounded by exactly one space.

```
Use 'VEVENT,VTODO', not 'VEVENT, VTODO'
Use 'DTSTART <= 20000605T131313Z', not
    'DTSTART <= 20000605T131313Z'.
Use ' AND ' and ' OR ', not 'AND' and not 'OR'.
```

(3) There is no ORDERBY. Sorting will take place in the order the columns are supplied in the command.

(4) The CS MUST sort at least the first column. The CS MAY sort additional columns.

(5) If EXPAND=FALSE and if colname is "*" sorting will be by the DTSTART value ascending. If EXPAND=TRUE and if colname is "*" sorting will be by the RECURRENCE-ID value ascending.

If colname is "*" and capmin-coms is VALARM only then sorting will be by TRIGGER time in UTC ascending.

(6) SQL-MIN MUST be implemented.

4.1.3 Querying Experminental Properties

4.1.4 Example, Query by UID

The following example would match the entire content of the component with the UID property equal to "uid123" and not expand any multiple instances of the component. If the CUA does not know if "uid123" was a VEVENT, VTODO, VJOURNAL, or any other component, then all components that the CUA supports MUST be supplied on the QUERY property. This example assumes the CUA only supports VTODO and VEVENT.

If the results were empty it could also mean that "uid123" was a property in a component other than a VTODO or VEVENT.

```
BEGIN:VQUERY
QUERY:SELECT * FROM VEVENT,VTODO WHERE UID = 'uid123'
END:VQUERY
```

The following example would match the entire content of the component with the UID property equal to "uid123" and would expand any instances of the component after applying any recurrence rules. This query could select multiple instances of components each with the same UID. Each instance would have a unique RECURRENCE-ID of the expanded component.

```
BEGIN:VQUERY
EXPAND:TRUE
QUERY:SELECT * FROM VEVENT,VTODO WHERE UID = 'uid123'
END:VQUERY
```

4.1.5 Query by Date-Time range

This query selects the entire content of every booked VEVENT that has an instance greater than or equal to July 1st, 2000 00:00:00 UTC and less than or equal to July 31st, 2000 23:59:59 UTC

```
BEGIN:VQUERY
EXPAND:TRUE
QUERY:SELECT * FROM VEVENT
  WHERE RECURRENCE-ID >= '20000801T000000Z'
  AND RECURRENCE-ID <= '20000831T235959Z'
  AND METHOD = 'CREATE'
END:VQUERY
```

4.1.6 Query for all Non-Booked Entries

The following example selects the entire content of all scheduling VEVENTS in the CS. The default for EXPAND is FALSE, so the recurrence rules will not be expanded.

```
BEGIN:VQUERY
QUERY:SELECT * FROM VEVENT,VTODO WHERE METHOD IS NOT NULL
END:VQUERY
```

The following example fetches the UIDs of all non-booked VEVENTs and VTODOs.

```
BEGIN:VQUERY
QUERY:SELECT UID FROM VEVENT,VTODO WHERE METHOD IS NOT NULL
END:VQUERY
```

4.1.7 Query with Subset of Properties by Date/Time

In this example only the named properties will be selected and all booked and non-booked components will be selected that have a DTSTART from February 1st to February 10th 2000.

```
BEGIN:VQUERY
QUERY:SELECT UID,DTSTART,DESCRIPTION,SUMMARY FROM VEVENT
```

```
     WHERE DTSTART >= '20000201T000000Z'
     AND DTSTART <= '20000210T235959Z'
   END:VQUERY
```

4.1.8 Components With Alarms In A Range

This example fetches all components with an alarm that triggers within the specified time range. In this case only the UID, SUMMARY, and DESCRIPTION will be selected for all booked VEVENTS that have an alarm between the two date-times.

```
   BEGIN:VQUERY
   EXPAND:TRUE
   QUERY:SELECT UID,SUMMARY,DESCRIPTION FROM VEVENT
     WHERE VALARM.TRIGGER >= '20000101T030405Z'
     AND VALARM.TRIGGER <= '20001231T235959Z'
     AND METHOD = 'CREATE'
   END:VQUERY
```

5. Access Rights

Access rights within CAP are specified with the "VCAR" calendar component, "RIGHTS" value type and the "GRANT", "DENY" and "CARID" component properties.

Properties within a VCAR must be evaluated in the order provided.

5.1 VCAR Inheritance

Calendar access rights specified in a calendar store are inherited as default calendar access rights for any calendar in the parent calendar store. Likewise, any calendar access rights specified in a root calendar are inherited as default calendar access rights for any sub-calendar to the root calendar. Furthermore, calendar access rights specified in a sub-calendar are inherited as default calendar access rights for any calendars that are hierarchically below the sub-calendar.

Calendar access rights specified in a calendar override any default calendar access rights. Calendar access rights specified within a sub-calendar override any default calendar access rights.

5.2 Access Control and NOCONFLICT

The TRANSP property can take on values (TRANSPARENT-NOCONFLICT, OPAQUE-NOCONFLICT) that prohibit other events from overlapping it. This setting overrides access. The ALLOW-CONFLICT Calendar or component setting may also prevent overlap, returning an error code "6.3"

6. Commands and Responses

CAP commands and responses are described in this section.

As mentioned in Section 3.2, CAP commands are defined by XML documents. The syntax of the commands is defined in Section 9, this section describes their semantic.

The attributes of a command are described in the "Attributes:" section in the command descriptions below. Similarly the "Elements:" section describes the elements that compose the command. The "Response:" section, identifies the responses that may be returned by the server.

In the examples below, lines preceded with "S:" refer to the server and lines preceded with "C:" refer to the client. Lines in which the first non-whitespace character is a "#" are editorial comments and are not part of the protocol.

6.1 Session Commands

6.1.1 "generate-uid" Command

Attributes:

num: Number of UIDs to generate (1 if omitted).

Response:

"uid-list"

The "generate-uid" command returns one or more unique identifiers which MUST be unique on the server's calendar store. It is recommended that the return values be globally unique ids.

Example:

```
C: MSG 1 5 . 2837 60
C: Content-Type: application/beep+xml
C:
C: <generateuid num=5/>
C: END
S: RPY 1 5 . 2897 328
S: Content-Type: application/beep+xml
S:
S: <uid-list>
S:   <uid>20011121T120000Z-12340@cal.example.com</uid>
S:   <uid>20011121T120000Z-12341@cal.example.com</uid>
```

```
S:   <uid>20011121T120000Z-12342@cal.example.com</uid>
S:   <uid>20011121T120000Z-12343@cal.example.com</uid>
S:   <uid>20011121T120000Z-12344@cal.example.com</uid>
S: </uid-list>
S: END
```

6.1.2 "get-capability" Command

Attributes:

None

Elements:

None

Response:

"capability"

The "get-capability" command returns information about the Calendar Service given the current state of the connection with the client. The values returned may differ depending on current user identify and the security level of the connection.

Client implementations SHOULD NOT require any capability element beyond those defined in this specification, and MAY ignore any non-standard, experimental capability elements. Non-standard experimental capability elements MUST be prefixed with the text "x-". The prefix SHOULD also include a vendor identifier. For example, "x-foo-barcapability", for the non-standard "barcapability" capability of the vendor "foo". It may return different results depending on the UPN.

Capability	Occurs	Description
cap	1	Container for CAP related elements.
version	1+	Version(s) of CAP, MUST include at least "1.0".
query-level	1+	Indicates level of SQL support. SQL-MIN or SQL-92. MUST include at least SQL-MIN.
car	1+	Indicates level of CAR support. CAR-MIN or CAR-FULL-1. CAR-MIN MUST be present.

```
         date          0 or 1
            max        0 or 1     The datetime value in UTC beyond
                                  which the server cannot accept. If
                                  not specified the default is
                                  99991231T235959Z.
            min        0 or 1     The datetime value prior to which
                                  the server cannot accept. If not
                                  specified the default is
                                  00000101T000000Z.

       icalendar       1          Container for CAP related elements.
         version       1+         Version(s) of iCalendar that is (are)
supported.
         max-component-size
                       0 or 1     A positive integer value that specifies
                                  the size of the largest iCalendar object that
                                  the server will accept in bytes. Objects
                                  larger than this will be rejected.
                                  The absence of this attribute indicates
                                  no limit.

       itip            1          Container for iTIP related elements.
         version       1+         Version(s) of ITIP, MUST include at least
                                  "1.0".
```

Example:

```
   C: MSG 1 6 . 3225 57
   C: Content-Type: application/beep+xml
   C:
   C: <get-capability/>
   C: END
   S: RPY 1 6 . 3282 423
   S: Content-Type: application/beep+xml
   S:
   S:
   S: <capability>
   S:  <icalendar>
   S:    <version>2.1</version>
   S:    <max-component-size>65536</max-component-size>
   S:  </icalendar>
   S:  <itip>
   S:    <version>1.0</version>
   S:  </itip>
   S:  <cap>
   S:    <version>1.0</version>
   S:    <car>CAR-MIN</car>
   S:    <query-level>SQL-MIN</query-level>
   S:    <date>
   S:         <min>00000101T000000Z</min>
   S:         <max>99991231T235959Z</max>
   S:     </date>
   S:   </cap>
   S: </capability>
   S: END
```

6.1.3 "identify" Command

Attribute:

upn: The UPN of the new identify to assume.

Element:

None

Response:

"result" with one of the following request-status codes:

2.0 Successful.

6.4 Identity not permitted.

The "identify" command allows the CUA to set a new identity to be used for calendar access.

The CS determines through an internal mechanism if the credentials supplied at authentication permit the assumption of the selected identity. If they do, the session assumes the new identity, otherwise a security error is returned.

6.1.4 "noop" Command

Arguments:

None

Element:

None

Response:

"result" with the following request-status code:

      2.0 successful

This command does nothing. It can be sent to the server periodically to request that the CS does not time out the session.

[EDITORS NOTE: should an unauthenticated and unidentified client be able to issue this command?]

[EDITORS NOTE: in view of the integration with BEEP should "noop" be removed?]

Example:

```
C: MSG 1 7 . 3705 47
C: Content-Type: application/beep+xml
C:
C: <noop/>
C: END
S: RPY 1 7 . 3752 91
S: Content-Type: application/beep+xml
S:
S: <result>
S:   <request-status code="2.0"/>
S: </result>
S: END
```

6.2 Calendaring and Scheduling Commands

6.2.1 Restriction Tables

Calendaring data are sent encapsulated in iCalendar objects (see Section 6.2.3.1). The restriction tables listed in the commands below describe the composition of the iCalendar data for these commands and replies.

The presence column uses the following values to assert whether a property is required, is optional and the number of times it may appear in the iCalendar object. A comment may be provided to further clarify the presence criteria.

The table below defines the values for the presence column.

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

While the tables list every component and property, their purpose is not to define the meaning of the component or property.

6.2.2 Common Attributes

6.2.2.1 "id" Attribute

The "id" attribute is an optional identifier for the command. When specified, the CS will include this attribute in all the related messages it returns to the client.

The "id" attribute is mainly useful for the "timeout" message (see Section 3.3). The CAP server imposes no restriction on the value. If uniqueness is required, then it is the responsibility of the CUA to generate unique values.

6.2.3 Common Elements

6.2.3.1 "data" Element

The role of the "data" element is to join an iCalendar document to an XML document forming a CAP command or response. The "data" element is composed of a single attribute ("content") that MUST be set to a "cid" URL that refers to an iCalendar document. See Section 3.2 for more information.

Depending of the context, the content of the referred iCalendar object is subject to restrictions. See Section 6.2.1 for more details.

6.2.3.2 "select" Element

Many calendaring commands can target several components stored on the CS (e.g., "search", "delete", "modify" and "move"). The "select" element is used to identify the targeted components.

The "select" element is composed of the following:

A "data" element that MUST refer to a VQUERY component.

One or more "source" elements that identify the containers to consider.

Restriction Table for the "data" element:

Component/Property	Presence	Comment
VCALENDAR	1	
. VERSION	1	MUST be 2.1
. [IANA-PROP]	0+	any IANA registered property
. VQUERY	1	
. . EXPAND	0 or 1	
. . QUERYNAME	0 or 1	MUST be present if QUERY is absent.
. . QUERY	0 or 1	MUST be present if QUERYNAME is absent.
. . [IANA-PROP]	0+	any IANA registered property
. VTIMEZONE	0+	MUST be present if any date/time refers to a timezone
. . DAYLIGHT	0+	MUST be one or more of either STANDARD or DAYLIGHT
. . . . COMMENT	0 or 1	
. . . . DTSTART	1	
. . . . RDATE	0+	if present RRULE MUST NOT be present
. . . . RRULE	0+	if present RDATE MUST NOT be present
. . . . TZNAME	0 or 1	
. . . . TZOFFSET	1	
. . . . TZOFFSETFROM	1	
. . . . TZOFFSETTO	1	
. . . . X-PROPERTY	0+	
. . . . [IANA-PROP]	0+	any IANA registered property
. . LAST-MODIFIED	0 or 1	
. . STANDARD	0+	
. . . . COMMENT	0 or 1	
. . . . DTSTART	1	
. . . . RDATE	0+	if present RRULE MUST NOT be present
. . . . RRULE	0+	if present RDATE MUST NOT be present
. . . . TZNAME	0 or 1	
. . . . TZOFFSETFROM	1	
. . . . TZOFFSETTO	1	
. . . . X-PROPERTY	0+	

```
          . . . . [IANA-PROP]  0+       any IANA registered
                                        property
          . . TZID             1
          . . TZURL            0 or 1
          . . X-PROPERTY       0+
          . . [IANA-PROP]      0+       any IANA registered
```

6.2.3.3 "source" Element

The "source" element is used to specify container(s) that will be examined during the execution of a CAP command. The "source" element is similar to the "target" element (see Section 6.2.3.4, but can refer to several containers (e.g., a calendar hierarchy or all the calendar owned by a given CU).

Attributes:

csid: when specified MUST point to a CSID. When omitted the CSID of the current server is assumed.

relcalid: when specified MUST point to a RELCALID. The value is relative the value of the "csid" attribute.

depth: specifies the maximal depth of the calendar hierarchy to explore. When omitted the value "0" is assumed. The accepted values are positives integers and "*".

owner: if present MUST be set to a UPN. When specified only the VAGENDA owned by the given UPN are considered.

6.2.3.4 "target" Element

The "target" element is used to specify a container targeted by a CAP command (e.g., the destination of a "create" command). A "target" element MAY refer to a VAGENGA or the top level container of a Calendar Store.

Attributes:

csid: when specified MUST point to a CSID. When omitted the CSID of the current server is assumed.

relcalid: when specified MUST point to a RELCALID. The value is relative the value of the "csid" attribute.

6.2.4 Calendaring Commands

Calendaring commands allow a CUA to directly manipulate a calendar.

Calendar access rights can be granted for the more generalized access provided by the calendar commands.

6.2.4.1 "create" Command

Attributes:

"id" (see Section 6.2.2.1).

Elements:

"max-time": See Section 3.3.

"target": Each "target" element points to a container where the new component will be created.

"data": MUST point to an iCalendar object defining the component(s) to create. See the restriction table given below.

Response:

One "result" message per "target" element MUST be returned (see Section 3.1).

One of the following "request-status" codes MUST be returned:

2.0 - successfully created the component or calendar

6.1 - Container not found

6.3 - Bad args

The "data" element of each "result" message is subject to the result restriction table defined below.

The "create" command is used to create one or more iCalendar objects. The "target" elements specify the containers where the component(s) will be created.

Restriction table for the "data" element of the "create" command:

Component/Property	Presence	Comment
VCALENDAR	1	

```
       . VERSION              1        MUST be 2.1
       . [IANA-PROP]          0+       any IANA registered
                                       property

       . VAGENDA              0+
       . . CALMASTER          0 or 1
       . . NAME               0 or 1
       . . OWNER              1+
       . . RELCALID           1
       . . TZID               0 or 1
       . . [IANA-PROP]        0+       any IANA registered
                                       property

       . VCAR                 0+
       . . CARID              0 or 1
       . . DENY               0+       Note, there must be at
                                       least one GRANT or DENY
                                       within the VCAR.
       . . GRANT              0+       Note, there must be at
                                       least one GRANT or DENY
                                       within the VCAR.
       . . [IANA-PROP]        0+       any IANA registered
                                       property

       . VQUERY               0+
       . . EXPAND             0 or 1
       . . QUERYNAME          1
       . . QUERY              1
       . . [IANA-PROP]        0+     any IANA registered
                                     property

       . VEVENT               0+
       . . ATTENDEE           0+
       . . 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 or 1
       . . CLASS              0 or 1
       . . COMMENT            0 or 1
       . . CONTACT            0+
       . . CREATED            0 or 1
       . . DESCRIPTION        0 or 1   Can be null
       . . DTEND              0 or 1   if present DURATION MUST
                                       NOT be present
```

```
       . . DTSTAMP            1
       . . DTSTART            1
       . . DURATION           0 or 1   if present DTEND MUST NOT
                                       be present
       . . EXDATE             0+
       . . EXRULE             0+
       . . GEO                0 or 1
       . . LAST-MODIFIED      0 or 1
       . . LOCATION           0 or 1
       . . ORGANIZER          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 or 1   This property MAY contain a
                                       list of values
       . . RRULE              0+
       . . STATUS             0 or 1
       . . TRANSP             0 or 1
       . . URL                0 or 1
       . . X-PROPERTY         0+
       . . [IANA-PROP]        0+       any IANA registered
                                       property

       . . VALARM             0+
       . . . ACTION           1
       . . . ALARMID          1
       . . . ATTACH           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
       . . . X-PROPERTY       0+
       . . . [IANA-PROP]      0+      any IANA registered property


       . VTODO                0+
       . . ATTENDEE           0+
       . . 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 or 1	This property may contain a list of values
. . CLASS	0 or 1	
. . COMMENT	0 or 1	
. . CONTACT	0+	
. . CREATED	0 or 1	
. . DESCRIPTION	0 or 1	Can be null
. . DTSTAMP	1	
. . DTSTART	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+	
. . EXRULE	0+	
. . GEO	0 or 1	
. . LAST-MODIFIED	0 or 1	
. . LOCATION	0 or 1	
. . ORGANIZER	1	
. . PRIORITY	1	
. . PERCENT-COMPLETE	0 or 1	
. . RDATE	0+	
. . RECURRENCE-ID	0 or 1	MUST 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 or 1	This property may contain a list of values
. . RRULE	0+	
. . STATUS	0 or 1	MAY be one of COMPLETED, NEEDS-ACTION, IN-PROCESS, CANCELLED
. . URL	0 or 1	
. . X-PROPERTY	0+	
. . [IANA-PROP]	0+	any IANA registered property
. . VALARM	0+	
. . . ACTION	1	
. . . ALARMID	1	

```
      . . . ATTACH           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
      . . . X-PROPERTY       0+
      . . . [IANA-PROP]      0+      any IANA registered property


      . VJOURNAL             0+
      . . ATTENDEE           0
      . . DESCRIPTION        1    Can be null.
      . . DTSTAMP            1
      . . DTSTART            1
      . . ORGANIZER          1
      . . UID                1

      . . ATTACH             0+
      . . CATEGORIES         0 or 1 This property MAY contain a list
                                    of values
      . . CLASS              0 or 1
      . . COMMENT            0 or 1
      . . CONTACT            0+
      . . CREATED            0 or 1
      . . EXDATE             0+
      . . EXRULE             0+
      . . LAST-MODIFIED      0 or 1
      . . RDATE              0+
      . . RECURRENCE-ID      0 or 1   MUST only if referring to
                                      an instance of a recurring
                                      calendar component.
                                      Otherwise it MUST NOT be
                                      present.
      . . RELATED-TO         0+
      . . REQUEST-STATUS     0+
      . . RRULE              0+
      . . SEQUENCE           0 or 1   MUST be present if
                                      non-zero. MAY be
                                      present if zero.
      . . STATUS             0 or 1
      . . SUMMARY            0 or 1   Can be null
      . . URL                0 or 1
      . . X-PROPERTY         0+
      . . [IANA-PROP]        0+       any IANA registered
                                      property
```

. VFREEBUSY	0	
. VTIMEZONE	0+	MUST be present if any date/time refers to a timezone
. . DAYLIGHT	0+	MUST be one or more of either STANDARD or DAYLIGHT
. . . . COMMENT	0 or 1	
. . . . DTSTART	1	
. . . . RDATE	0+	if present RRULE MUST NOT be present
. . . . RRULE	0+	if present RDATE MUST NOT be present
. . . . TZNAME	0 or 1	
. . . . TZOFFSET	1	
. . . . TZOFFSETFROM	1	
. . . . TZOFFSETTO	1	
. . . . X-PROPERTY	0+	
. . . . [IANA-PROP]	0+	any IANA registered property
. . LAST-MODIFIED	0 or 1	
. . STANDARD	0+	
. . . . COMMENT	0 or 1	
. . . . DTSTART	1	
. . . . RDATE	0+	if present RRULE MUST NOT be present
. . . . RRULE	0+	if present RDATE MUST NOT be present
. . . . TZNAME	0 or 1	
. . . . TZOFFSETFROM	1	
. . . . TZOFFSETTO	1	
. . . . X-PROPERTY	0+	
. . . . [IANA-PROP]	0+	any IANA registered property
. . TZID	1	
. . TZURL	0 or 1	
. . X-PROPERTY	0+	
. . [IANA-PROP]	0+	any IANA registered property

Restriction Table for the "data" element of the "result" response:

Component/Property	Presence	Comment
VCALENDAR	1+	
. VERSION	1	MUST be 2.1
. VAGENDA	0+	

```
       . . RELCALID           1
       . . REQUEST-STATUS     1+

       . VCAR                 0+
       . . CARID              1
       . . REQUEST-STATUS     1+

       . VEVENT               0+
       . . UID                1        The UID for which this
                                       REQUEST-STATUS applies
       . . REQUEST-STATUS     1+
       . . RECURRENCE-ID      0 or 1   MUST be specified only if instance of
                                       a recurring component was created.
       . . VALARM             0        if VEVENT was successfully
                                       saved
                              1+       if there were errors saving
                                       alarms
       . . . ALARMID          1
       . . . REQUEST-STATUS   1+

       . VFREEBUSY            0

       . VJOURNAL             0+
       . . UID                1        The UID for which this
                                       REQUEST-STATUS applies
       . . RECURRENCE-ID      0 or 1   MUST be specified only if instance of
                                       a recurring component was created.
       . . REQUEST-STATUS     1+

       . VQUERY               0+
       . . REQUEST-STATUS     1+

       . VTODO                0+
       . . UID                1        The UID for which this
                                       REQUEST-STATUS applies
       . . RECURRENCE-ID      0 or 1   MUST be specified only if instance of
                                       a recurring component was created.
       . . REQUEST-STATUS     1+
       . . VALARM             0        if VTODO was successfully
                                       saved
                              1+       if there were errors saving
                                       alarms
       . . . ALARMID          1
       . . . REQUEST-STATUS   1+
```

Example:

In the following example, two new top level VAGENDAs are created.

Note that the CSID of the server is cal.example.com.

```
C: MSG 1 8 . 3843 778
C: Content-Type: multipart/related; boundary="boundary-foo321";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-foo321
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <create id="creation01">
C:   <target csid="cal.example.com"/>
C:   <data content="cid:2@cal.example.com"/>
C: </create>
C: --boundary-foo321
C: Content-Type: text/calendar
C: Content-ID: 2@cal.example.com
C:
C: BEGIN:VCALENDAR
C: VERSION:2.1
C: BEGIN:VAGENDA
C: RELCALID:relcalz1
C: NAME;LANGUAGE=EN-us:Bill's Soccer Team
C: OWNER:bill
C: CALMASTER:mailto:bill@example.com
C: TZID:US_PST
C: END:VAGENDA
C: BEGIN:VAGENDA
C: RELCALID:relcalz2
C: NAME;LANGUAGE=EN-us:Mary's personal calendar
C: OWNER:mary
C: CALMASTER:mailto:mary@example.com
C: TZID:US_PST
C: END:VAGENDA
C: END:VCALENDAR
C: --boundary-foo321--
C: END
S: RPY 1 8 . 4621 647
S: Content-Type: multipart/related; boundary="boundary-bar321";
S:    start="1@cal.example.com";
S:    type="application/beep+xml"
S:
S: --boundary-bar321
S: Content-Type: application/beep+xml
S: Content-ID: 1@cal.example.com
S:
S: <result id="creation01">
```

```
S:    <target csid="cal-example.com"/>
S:    <request-status code="2.0"/>
S:    <data content="cid:2@cal.example.com"/>
S: </result>
S: --boundary-bar321
S: Content-Type:text/calendar;
S: Content-ID: 2@cal.example.com
S:
S: BEGIN:VCALENDAR
S: VERSION:2.1
S: BEGIN:VAGENDA
S: RELCALID:relcalz1
S: REQUEST-STATUS:2.0
S: END:VAGENDA
S: BEGIN:VAGENDA
S: RELCALID:relcalz2
S: REQUEST-STATUS:2.0
S: END:VAGENDA
S: END:VCALENDAR
S: --boundary-bar321--
S: END
```

Example to create a new component in multiple containers.

```
C: MSG 1 9 . 5268 622
C: Content-Type: multipart/related; boundary="boundary-kshgd";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-kshgd
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <create id="creation02">
C:   <target relcalid="relcalz1"/>
C:   <target relcalid="relcalz2"/>
C:   <data content="cid:2@cal.example.com"/>
C: </create>
C: --boundary-kshgd
C: Content-Type: text/calendar
C: Content-ID: 2@cal.example.com
C:
C: BEGIN:VCALENDAR
C: VERSION:2.1
C: BEGIN:VEVENT
C: DTSTART:99990307T180000Z
C: UID:abcd12345
C: DTEND:99990307T190000Z
```

```
   C: SUMMARY:Important Meeting
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-kshgd--
   C: END
   S: ANS 1 9 . 58901 563 0
   S: Content-Type: multipart/related; boundary="boundary-eqrga";
   S:    start="1@cal.example.com";
   S:    type="application/beep+xml"
   S:
   S: --boundary-eqrga
   S: Content-Type: application/beep+xml
   S: Content-ID: 1@cal.example.com
   S:
   S: <result id="creation02">
   S:    <target relcalid="relcalz1"/>
   S:    <request-status code="2.0"/>
   S:    <data content="2@cal.example.com"/>
   S: </result>
   S: --boundary-eqrga
   S: Content-Type: text/calendar
   S: Content-ID: 2@cal.example.com
   S:
   S: BEGIN:VCALENDAR
   S: VERSION:2.1
   S: BEGIN:VEVENT
   S: UID:abcd12345
   S: REQUEST-STATUS:2.9
   S: END:VEVENT
   S: END:VCALENDAR
   S: --boundary-eqrga--
   S: END
   S: ANS 1 9 . 6453 563 1
   S: Content-Type: multipart/related; boundary="boundary-982hf";
   S:    start="1@cal.example.com";
   S:    type="application/beep+xml"
   S:
   S: --boundary-982hf
   S: Content-Type: application/beep+xml
   S: Content-ID: 1@cal.example.com
   S:
   S: <result id="creation02">
   S:    <target relcalid="relcalz2"/>
   S:    <request-status code="2.0"/>
   S:    <data content="2@cal.example.com"/>
   S: </result>
   S: --boundary-982hf
   S: Content-Type: text/calendar
```

```
S: Content-ID: 2@cal.example.com
S:
S: BEGIN:VCALENDAR
S: VERSION:2.1
S: BEGIN:VEVENT
S: UID:abcd12345
S: REQUEST-STATUS:6.0
S: END:VEVENT
S: END:VCALENDAR
S: --boundary-982hf--
S: END
S: NUL 1 9 . 7016 0
S: END
```

As described in Section 3.1, the CS sends one response per "target" element present in the "create" command.

6.2.4.2 "delete" Command

Attributes:

"id" (see Section 6.2.2.1).

Elements:

"max-time": See Section 3.3.

"select": specifies the components to delete (see Section 6.2.3.2).

Response:

One "result" message per "source" in the "select" element (see Section 3.1).

One of the following "request-status" codes MUST be returned:

2.0 - successfully created the component or calendar

6.1 - Container not found

6.3 - Bad args

The "data" element of each "result" message is subject to the result restriction table define below.

The "delete" command is used to delete a calendar or component. The "select" element specifies the container(s) to delete.

Restriction Table for the "data" element of the "result" response(s).

Component/Property	Presence	Comment
VCALENDAR	1+	
. VERSION	1	MUST be 2.1
. VAGENDA		Only if VAGENDAS were deleted
. RELCALID	1	
. REQUEST-STATUS	1	
. VCAR	0+	Only if VCAR components were deleted
. . CARID	1	
. . REQUEST-STATUS	1	
. VEVENT	0+	Only if VEVENT components were deleted
. . UID	1	
. . REQUEST-STATUS	0 or 1	Omitted if an embedded VALARM was the target of the deletion.
. . VALARM	0+	Only if VALARM components were deleted
. . . ALARMID	1	
. . . REQUEST-STATUS	1	
. VFREEBUSY	0	
. . UID	1	
. . DTSTAMP	1	
. . REQUEST-STATUS	1	
. VJOURNAL	0+	Only if VJOURNAL components were deleted
. . UID	1	
. . REQUEST-STATUS	1	
. VQUERY	0+	Only if VQUERY components were deleted
. UID	1	
. REQUEST-STATUS	1	
. VTIMEZONE	0+	Only if VTIMEZONE components
. . TZID		were deleted
. . REQUEST-STATUS	1	

```
     . VTODO               0+       Only if VTODO components were
                                    deleted
     . . UID               1
     . . REQUEST-STATUS    0 or 1   Omitted if an embedded VALARM was
                                    the target of the deletion.

     . . VALARM            0+       Only if VALARM components
                                    were deleted
     . . . ALARMID         1
     . . . REQUEST-STATUS  1


     ----------------------------------------------------------
```

[EDITORS NOTE: Issues:

- Can one use DELETE to remove all VALARMs and VTIMEZONEs that match a certain search criteria and that belong to all components, event though VALARMs and VTIMEZONEs never exist as independent components? Or should one use MODIFY? If they can be deleted, do we return the REQUEST-STATUS of their deletion in a VEVENT or separately?

Example to delete a VEVENT with UID 'abcd12345' from any of the calendar owned by the CU with the UPN="user@cal.example.com":

```
C: MSG 1 10 . 7016 558
C: Content-Type: multipart/related; boundary="boundary-gsdmx3";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-gsdmx3
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <delete id="delete01">
C:    <select>
C:       <source depth=* owner="user@cal.example.com"/>
C:       <data content="cid:2@cal.example.com"/>
C:    </select>
C: </delete>
C: --boundary-gsdmx3
C: Content-Type: text/calendar
C: Content-ID: 2@cal.example.com
C:
C: BEGIN:VQUERY
C: QUERY:SELECT * FROM VEVENT WHERE UID = 'abcd12345'
C: END:VQUERY
```

```
   C: --boundary-gsdmx3--
   C: END
   S: RPY 1 10 . 7574 587
   S: Content-Type: multipart/related; boundary="boundary-oifc3j";
   S:    start="1@cal.example.com";
   S:    type="application/beep+xml"
   S:
   S: --boundary-oifc3j
   S: Content-Type: application/beep+xml
   S: Content-ID: 1@cal.example.com
   S:
   S: <result id="delete01">
   S:    <source depth=* owner="user@cal.example.com"/>
   S:    <data content="cid:2@cal.example.com"/>
   S:    <request-status code="2.0"/>
   S: </result>
   S: --boundary-oifc3j
   S: Content-Type: text/calendar
   S: Content-ID: 2@cal.example.com
   S:
   S: BEGIN:VCALENDAR
   S: VERSION:2.1
   S: BEGIN:VEVENT
   S: UID:abcd12345
   S: REQUEST-STATUS: 2.0
   S: END:VEVENT
   S: END:VCALENDAR
   S: --boundary-oifc3j--
   S: END
```

6.2.4.3 "modify" Command

Attributes:

"id" (see Section 6.2.2.1).

Elements:

"max-time": See Section 3.3.

"select": identifies the component(s) to modify.

"add": adds properties to the selected component(s).

"remove": removes properties from the selected component(s).

"update": updates the content of the selected component(s).

Response:

One "result" message per "source" in the "select" is returned (see Section 3.1).

One of the following "request-status" codes MUST be returned:

2.0 - successfully created the component or calendar

6.1 - Container not found

6.3 - Bad args

The "data" element of each "result" message is subject to the restriction table defined below.

The "modify" command is used to modify existing components. The "select" element specifies the components to modify. The "add", "remove" and "update" elements define the operations to perform.

The "add" element is used to add properties or nested components to the selected components. The "add" element is composed of a "data" element that contains a component with the properties to add. For example to add an inline attachment to a VEVENT the following iCalendar object could be:

```
BEGIN:VCALENDAR
BEGIN:VEVENT
ATTACH;FMTTYPE=image/basic;ENCODING=BASE64;VALUE=BINARY:
 MIICajCCAdOgAwIBAgICBEUwDQYJKoZIhvcNAQEEBQAwdzELMAkGA1U
 EBhMCVVMxLDAqBgNVBAoTI05ldHNjYXBlIENvbW11bmljYXRpb25zIE
   <...remainder of "BASE64" encoded binary data...>
END:VEVENT
END:VCALENDAR
```

The "remove" element is used to remove properties from the selected components. The "data" element contains an iCalendar with the properties to delete. When the "ignore-value" attribute is set to true, all the properties specified in the "data" element are removed even if the values do not match the current state of the component. This is useful to remove potentially large properties efficiently (e.g., "ATTACH").

The "update" element is used to update or add the properties referred to by the "data" element. If the "remove-missing" attribute is set to true, then all the elements not present in the "data" element document will be removed from the selected components.

When more than one operations is specified, the modifications MUST must respect the following order: "remove" followed by "update" followed by "add". The modifications MUST only be applied if the resulting component respects the restriction table of the "create" command.

Restriction Table for "data" element of the "result" response:

Component/Property	Presence	Comment
VCALENDAR	1+	
. VERSION	1	MUST be 2.1
. VAGENDA	0+	
. . RELCALID	1	
. . REQUEST-STATUS	1+	
. VCAR	0+	
. . CARID	1	
. . REQUEST-STATUS	1+	
. VEVENT	0+	
. . UID	1	
. . RECURRENCE-ID	0 or 1	MUST be specified only if instance of a recurring component was modified.
. . REQUEST-STATUS	1+	
. . VALARM	0	if VEVENT was successfully saved
	1+	if there were errors saving alarms
. . . REQUEST-STATUS	1+	
. VFREEBUSY	0	
. VJOURNAL	0+	
. . UID	1	
. . RECURRENCE-ID	0 or 1	MUST be specified only if instance of a recurring component was modified.
. . REQUEST-STATUS	1+	
. VQUERY	0+	
. . REQUEST-STATUS	1+	
. VTODO	0+	
. . UID	1	
. . RECURRENCE-ID	0 or 1	MUST be specified only if instance of

```
                                     a recurring component was modified.
       . . REQUEST-STATUS     1+
       . . VALARM             0        if VTODO was successfully
                                       saved
                              1+       if there were errors saving
                                       alarms
       . . . REQUEST-STATUS   1+
```

In the example below, the start and end time of the event with UID abcd12345 is changed and the LOCATION property is removed.

```
C: MSG 1 11 . 8161 1144
C: Content-Type: multipart/related; boundary="boundary-324dav";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-324dav
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.cal.example.com
C:
C: <modify id="modify01"/>
C:    <select>
C:       <source owner="user@cal.example.com" depth=*/>
C:       <data content="cid:query@cal.example.com"/>
C:    </select>
C:    <remove ignore-value=true>
C:       <data content="remove@cal.example.com"/>
C:    </remove>
C:    <update remove-missing=false>
C:       <data content="cid:update@cal.example.com"/>
C:    </update>
C: </modify>
C: --boundary-324dav
C: Content-Type: text/calendar
C: Content-ID: query@cal.example.com
C:
C: BEGIN:VCALENDAR
C: BEGIN:VQUERY
C: QUERY: SELECT * FROM VEVENT WHERE UID='abcd12345'
C: END:VQUERY
C: END:VCALENDAR
C: --boundary-324dav
C: Content-Type: text/calendar
C: Content-ID: remove@cal.example.com
C:
C: BEGIN:VCALENDAR
C: BEGIN:VEVENT
C: LOCATION:
```

```
C: END:VEVENT
C: END:VCALENDAR
C: --boundary-324dav
C: Content-Type: text/calendar
C: Content-ID: update@cal.example.com
C:
C: BEGIN:VCALENDAR
C: BEGIN:VEVENT
C: DTSTART:19990421T160000Z
C: DTEND:19990421T163000Z
C: END:VEVENT
C: END:VCALENDAR
C: --boundary-324dav--
C: END
S: RPY 1 11 . 9305 570
S: Content-Type: multipart/related; boundary="boundary-tvx2";
S:    start="command@cal.example.com";
S:    type="application/beep+xml"
S:
S: --boundary-tvx2
S: Content-Type: application/beep+xml
S: Content-ID: command@cal.example.com
S:
S: <result id="modify01">
S:    <source owner="user@cal.example.com"/>
S:    <request-status code="2.0"/>
S:    <data content="cid:2@cal.example.com"/>
S: </result>
S: --boundary-tvx2
S: Content-Type: text/calendar
S: Content-ID: 2@cal.example.com
S:
S: BEGIN:VCALENDAR
S: BEGIN:VEVENT
S: UID:abcd12345
S: REQUEST-STATUS: 2.0
S: END:VEVENT
S: END:VCALENDAR
S: --boundary-tvx2--
S: END
```

In this example, all instances of "Building 6" are replaced by "New office lobby" in VEVENTs:

```
C: MSG 1 12 . 9875 870
C: Content-Type: multipart/related; boundary="boundary-trew2";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
```

```
   C:
   C: --boundary-trew2
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com
   C:
   C: <modify id="modify02"/>
   C:    <select>
   C:       <source owner="user@cal.example.com" depth=*/>
   C:       <data content="cid:query@cal.example.com"/>
   C:    </select>
   C:    <update remove-missing=false>
   C:       <data content="cid:update@cal.example.com"/>
   C:    </update>
   C: </modify>
   C: --boundary-trew2
   C: Content-Type: text/calendar
   C: Content-ID: query@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: BEGIN:VQUERY
   C: QUERY: SELECT * FROM VEVENT WHERE LOCATION='Building 6'
   C: END:VQUERY
   C: END:VCALENDAR
   C: --boundary-trew2
   C: Content-Type: text/calendar
   C: Content-ID: update@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: BEGIN:VEVENT
   C: LOCATION:New office lobby
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-trew2--
   C: END
   S: RPY 1 12 . 10745 578
   S: Content-Type: multipart/related; boundary="boundary-poiu51";
   S:    start="command@cal.example.com";
   S:    type="application/beep+xml"
   S:
   S: --boundary-poiu51
   S: Content-Type: application/beep+xml
   S: Content-ID: command@cal.example.com
   S:
   S: <result id="modify02">
   S:    <source owner="user@cal.example.com"/>
   S:    <request-status code="2.0"/>
   S:    <data content="cid:2@cal.example.com"/>
   S: </result>
```

```
   S: --boundary-poiu51
   S: Content-Type: text/calendar
   S: Content-ID: 2@cal.example.com
   S:
   S: BEGIN:VCALENDAR
   S: BEGIN:VEVENT
   S: UID:abcd12345
   S: REQUEST-STATUS: 2.0
   S: END:VEVENT
   S: END:VCALENDAR
   S: --boundary-poiu51--
   S: END
```

6.2.4.4 "move" Command

Attributes:

"id" (see Section 6.2.2.1).

Elements:

"max-time": See Section 3.3.

"target": The "target" element points to the container where the components are to be relocated.

"select": identifies the component(s) to move.

Response:

One "result" message for each "source" in the "select" element is returned (see Section 3.1).

One of the following "request-status" codes MUST be returned:

2.0 - successfully created the component or calendar

6.1 - Container not found

6.3 - Bad args

The "data" element of each "result" message is subject to the result restriction table defined below.

The "move" command is used to move components within the CS's hierarchy of calendars. When moving VAGENDA, the CS MUST ensure that VCARs are still valid after the move, and the CS MUST update the

PARENT and CHILDREN properties of the new and old parent containers.

Restriction Table for "data" element of the "result" response:

Component/Property	Presence	Comment
VCALENDAR	1+	
. VERSION	1	MUST be 2.1
. VAGENDA	0+	
. . RELCALID	1	
. . REQUEST-STATUS	1+	
. VCAR	0+	
. . CARID	1	
. . REQUEST-STATUS	1+	
. VEVENT	0+	
. . UID	1	
. . REQUEST-STATUS	1+	
. . VALARM	0	if VEVENT was successfully saved
	1+	if there were errors saving alarms
. . . ALARMID	1	
. . . REQUEST-STATUS	1+	
. VFREEBUSY	0	
. VJOURNAL	0+	
. . UID	1	
. . REQUEST-STATUS	1+	
. VQUERY	0+	
. . REQUEST-STATUS	1+	
. VTODO	0+	
. . UID	1	
. . REQUEST-STATUS	1+	
. . VALARM	0	if VTODO was successfully saved
	1+	if there were errors saving alarms
. . . ALARMID	1	
. . . REQUEST-STATUS	1+	

```
          ---------------------------------------------------------
```

[EDITORS NOTE: Issues:

   1) Should one be able to move a calendar owned by person X into a
   calendar owned by person Y. (Can these such rights be specified
   in VCARs?)

Example: moving the VAGENDA Nellis to Area-51

```
C: MSG 1 12 . 11323 613
C: Content-Type: multipart/related; boundary="boundary-kljr";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-kljr
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <move id="move01"/>
C:    <select>
C:       <source csid="cal@example.com" depth=*>
C:       <data content="cid:query@cal.example.com"/>
C:    </select>
C:    <target relcalid="area-51"/>
C: </move>
C: --boundary-kljr
C: Content-Type: text/calendar
C: Content-ID: query@cal.example.com
C:
C: BEGIN:VCALENDAR
C: BEGIN:VQUERY
C: QUERY: SELECT * FROM VAGENDA WHERE RELCALID='Nellis'
C: END:VQUERY
C: END:VCALENDAR
C: --boundary-kljr--
C: END
S: RPY 1 2 . 11936 571
S: Content-Type: multipart/related; boundary="boundary-mnbvd";
S:    start="reply@cal.example.com";
S:    type="application/beep+xml"
S:
S: --boundary-mnbvd
S: Content-Type: application/beep+xml
S: Content-ID: reply@cal.example.com
S:
S: <result id="move01">
S:    <source csid=cal@example.com depth=*>
```

```
   S:    <request-status code="2.0"/>
   S:    <data content="cid:2@cal.example.com"/>
   S: </result>
   S: --boundary-mnbvd
   S: Content-Type: text/calendar
   S: Content-ID: 2@cal.example.com
   S:
   S: BEGIN:VCALENDAR
   S: BEGIN:VAGENDA
   S: RELCALID:Nellis
   S: REQUEST-STATUS: 2.0
   S: END:VAGENDA
   S: END:VCALENDAR
   S: --boundary-mnbvd--
   S: END
```

6.2.4.5 "search" Command

Attributes:

"id" (see Section 6.2.2.1).

Elements:

"max-time": See Section 3.3.

"select": identifies the components to return.

"max-results": maximum number of components to return per source (if omited unlimited).

"max-size": maximum size in bytes, of the iCalendar object to return.

Response:

A "result" message per "source" in the "select" element is returned (see Section 3.1).

One of the following "request-status" codes MUST be returned:

2.0 - successfully created the component or calendar

6.1 - Container not found

6.3 - Bad args

The "data" element of each "result" message points to an iCalendar object composed of all the selected components. Only "REQUEST-STATUS" and the properties mentioned in the "SELECT" clause of the QUERY are included in the components.

Searching for Events

In the example below events on March 10,1999 between 080000Z and 190000Z are read. In this case only 4 properties for each event are returned. Two calendars are specified. Only booked (vs scheduled) entries are to be returned. The first result returns two VEVENTs that match in that "source", the second result returns only one VEVENT for the second "source".

```
C: MSG 1 13 . 12507 704
C: Content-Type: multipart/related; boundary="boundary-5329";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-5329
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <search id="search01"/>
C:    <select>
C:       <source relcalid="relcal2"/>
C:       <source relcalid="relcal3"/>
C:       <data content="cid:query@cal.example.com"/>
C:    </select>
C: </search>
C: --boundary-5329
C: Content-Type: text/calendar
C: Content-ID: query@cal.example.com
C:
C: BEGIN:VCALENDAR
C: BEGIN:VQUERY
C: QUERY:SELECT DTSTART,DTEND,SUMMARY,UID
C:  FROM VEVENT
C:  WHERE DTEND >= '19990310T080000Z'
C:  AND DTSTART <= '19990310T190000Z'
C:  AND METHOD IS NULL
C: END:VQUERY
C: END:VCALENDAR
C: --boundary-5329--
C: END
S: ANS 1 13 . 13211 803 0
S: Content-Type: multipart/related; boundary="boundary-f4fw2";
S:    start="answer@cal.example.com";
```

```
   S:    type="application/beep+xml"
   S:
   S: --boundary-f4fw2
   S: Content-Type: application/beep+xml
   S: Content-ID: answer@cal.example.com
   S:
   S: <result id="search01">
   S:    <source relcalid="relcal2"/>
   S:    <request-status code="2.0"/>
   S:    <data content="cid:2@cal.example.com"/>
   S: </result>
   S: --boundary-f4fw2
   S: Content-Type: text/calendar
   S: Content-ID: 2@cal.example.com
   S:
   S: BEGIN:VCALENDAR
   S: VERSION:2.1
   S: BEGIN:VEVENT
   S: DTSTART:19990310T090000Z
   S: DTEND:19990310T100000Z
   S: UID:abcxyz12345
   S: SUMMARY:Meet with Sir Elton
   S: REQUEST-STATUS:2.0
   S: END:VEVENT
   S: BEGIN:VEVENT
   S: DTSTART:19990310T130000Z
   S: DTEND:19990310T133000Z
   S: UID:abcxyz8999
   S: SUMMARY:Meet with  brave Sir Robin
   S: REQUEST-STATUS:2.0
   S: END:VEVENT
   S: END:VCALENDAR
   S: --boundary-f4fw2--
   S: END
   S: ANS 1 13 . 14014 664 1
   S: Content-Type: multipart/related; boundary="boundary-r432";
   S:    start="answer@cal.example.com";
   S:    type="application/beep+xml"
   S:
   S: --boundary-r432
   S: Content-Type: application/beep+xml
   S: Content-ID: answer@cal.example.com
   S:
   S: <result id="search01">
   S:    <source relcalid="relcal3"/>
   S:    <request-status code="2.0"/>
   S:    <data content="cid:2@cal.example.com"/>
   S: </result>
```

```
S: --boundary-r432
S: Content-Type: text/calendar
S: Content-ID: 2@cal.example.com
S:
S: BEGIN:VCALENDAR
S: VERSION:2.1
S: BEGIN:VEVENT
S: REQUEST-STATUS:2.0
S: DTSTART:19990310T140000Z
S: DTEND:19990310T150000Z
S: UID:123456asdf
S: SUMMARY:Summer Budget
S: REQUEST-STATUS:2.0
S: END:VEVENT
S: END:VCALENDAR
S: --boundary-r432--
S: END
S: NUL 1 13 . 14678 0
S: END
```

The return values are subject to VCAR filtering. That is, if the request contains properties to which the UPN does not have access, those properties will not appear in the return values. If the UPN has access to at least one property of the component, but has been denied access to all properties called out in the request, the response will contain a single REQUEST-STATUS property indicating the error. That is, the VEVENT components will be the following:

[EDITORS NOTE: Should the one(s) that the UPN has access to - be returned?]

```
S: ANS 1 13 . 14014 548 0
S: Content-Type: multipart/related; boundary="boundary-fmei3";
S:    start="command@cal.example.com";
S:    type="application/beep+xml"
S:
S: --boundary-fmei3
S: Content-Type: application/beep+xml
S: Content-ID: command@cal.example.com
S:
S: <result>
S:    <source relcalid="relcalid"/>
S:    <request-status code="2.0"/>
S:    <data content="cid:2@cal.example.com"/>
S: </result>
S: --boundary-fmei3
S: Content-Type: text/calendar
S: Content-ID: 2@cal.example.com
```

```
S:
S: BEGIN:VCALENDAR
S: VERSION:2.1
S: BEGIN:VEVENT
S: REQUEST-STATUS:4.1
S: END:VEVENT
S: END:VCALENDAR
S: --boundary-fmei3--
S: END
```

If the UPN has no access to any events at all, the response will simply be an empty data set. The response looks the same if there are particular events to which the CU has been denied access.

```
S: ANS 1 13 . 14014 502 0
S: Content-Type: multipart/related; boundary="boundary-ewrvc";
S:    start="command@cal.example.com";
S:    type="application/beep+xml"
S:
S: --boundary-ewrvc
S: Content-Type: application/beep+xml
S: Content-ID: command@cal.example.com
S:
S: <result>
S:    <source relcalid="relcalid"/>
S:    <request-status code="2.0"/>
S:    <data content="cid:2@cal.example.com"/>
S: </result>
S: --boundary-ewrvc
S: Content-Type: text/calendar
S: Content-ID: 2@cal.example.com
S:
S: BEGIN:VCALENDAR
S: VERSION:2.1
S: END:VCALENDAR
S: --boundary-ewrvc--
S: END
```

Find alarms within a range of time for booked VEVENTs.

```
C: MSG 1 15 . 14678 747
C: Content-Type: multipart/related; boundary="boundary-weoiu";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-weoiu
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
```

```
   C:
   C: <search id="search02"/>
   C:    <select>
   C:       <source relcalid="relcal2"/>
   C:       <source relcalid="relcal3"/>
   C:       <data content="cid:query@cal.example.com"/>
   C:    </select>
   C: </search>
   C: --boundary-weoiu
   C: Content-Type: text/calendar
   C: Content-ID: query@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: BEGIN:VQUERY
   C: QUERY:SELECT DTSTART,DTEND,SUMMARY,UID,VALARM.*
   C:    FROM VEVENT,VTODO
   C:    WHERE VALARM.TRIGGER >= '19990310T080000Z'
   C:    AND VALARM.TRIGGER <= '19990310T190000Z'
   C:    AND METHOD IS NULL
   C: END:VQUERY
   C: END:VCALENDAR
   C: --boundary-weoiu--
   C: END
   S: ANS 1 15 . 15426 511 0
   S: Content-Type: multipart/related; boundary="boundary-kjhs";
   S:    start="command@cal.example.com";
   S:    type="application/beep+xml"
   S:
   S: --boundary-kjhs
   S: Content-Type: application/beep+xml
   S: Content-ID: command@cal.example.com
   S:
   S: <result id="search02">
   S:    <source relcalid="relcal2"/>
   S:    <request-status code="2.0"/>
   S:    <data content="cid:2@cal.example.com"/>
   S: </result>
   S: --boundary-kjhs
   S: Content-Type: text/calendar
   S: Content-ID: 2@cal.example.com
   S:
   S: BEGIN:VCALENDAR
   S: VERSION:2.1
   S: END:VCALENDAR
   S: --boundary-kjhs--
   S: END
   S: ANS 1 2 . 15937 734 1
   S: Content-Type: multipart/related; boundary="boundary-435fe";
```

```
S:    start="command@cal.example.com";
S:    type="application/beep+xml"
S:
S: --boundary-435fe
S: Content-Type: application/beep+xml
S: Content-ID: command@cal.example.com
S:
S: <result id="search02">
S:    <source relcalid="relcal3"/>
S:    <request-status code="2.0"/>
S:    <data content="cid:2@cal.example.com"/>
S: </result>
S: --boundary-435fe
S: Content-Type: text/calendar
S: Content-ID: 2@cal.example.com
S:
S: BEGIN:VCALENDAR
S: VERSION:2.1
S: BEGIN:VEVENT
S: DTSTART:19990310T130000Z
S: DTEND:19990310T133000Z
S: UID:abcxyz8999
S: SUMMARY:Meet with brave Sir Robin
S: BEGIN:VALARM
S: TRIGGER:19990310T132500Z
S: SUMMARY:Almost time...
S: ACTION:DISPLAY
S: END:VALARM
S: END:VEVENT
S: END:VCALENDAR
S: --boundary-435fe--
S: END
S: NUL 1 15 . 16671 0
S: END
```

In this example bill@example.com reads a day's worth of events from cap://cal.example.com/opaqueid99.

```
C: MSG 1 16 . 16671 668
C: Content-Type: multipart/related; boundary="boundary-vnj43";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-vnj43
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <search id="xyz12345"/>
```

```
   C:    <select>
   C:       <source relcalid="opaqueid99"/>
   C:       <data content="cid:query@cal.example.com"/>
   C:    </select>
   C: </search>
   C: --boundary-vnj43
   C: Content-Type: text/calendar
   C: Content-ID: query@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: BEGIN:VQUERY
   C: QUERY:SELECT DTSTART,DTEND,SUMMARY, UID FROM VEVENT
   C:   WHERE DTEND >= '19990714T080000Z'
   C:   AND DTSTART <= '19990715T080000Z'
   C: END:VQUERY
   C: END:VCALENDAR
   C: --boundary-vnj43--
   C: END
   S: RPY 1 16 . 17359 751
   S: Content-Type: multipart/related; boundary="boundary-rfew";
   S:    start="command@cal.example.com";
   S:    type="application/beep+xml"
   S:
   S: --boundary-rfew
   S: Content-Type: application/beep+xml
   S: Content-ID: command@cal.example.com
   S:
   S: <result id="xyz12345">
   S:    <source relcalid="opaqueid99"/>
   S:    <request-status code="2.0"/>
   S:    <data content="cid:2@cal.example.com"/>
   S: </result>
   S: --boundary-rfew
   S: Content-Type: text/calendar
   S: Content-ID: 2@cal.example.com
   S:
   S: BEGIN:VCALENDAR
   S: VERSION:2.1
   S: BEGIN:VEVENT
   S: DTSTART:19990714T200000Z
   S: DTEND:19990714T210000Z
   S: UID:000444888929922
   S: SUMMARY:Blah blah
   S: END:VEVENT
   S: BEGIN:VEVENT
   S: UID:0034848098038888989443
   S: SUMMARY:meeting
```

```
   S: DTEND:19990714T233000Z
   S: DTSTART:19990714T223000Z
   S: END:VEVENT
   S: END:VCALENDAR
   S: --boundary-rfew--
   S: END
```

In this example bill@example.com reads a day's worth of events from cap://cal.example.com/opaqueid101 and cap://cal.example.com/opaqueid103

```
   C: MSG 1 17 . 18110 694
   C: Content-Type: multipart/related; boundary="boundary=wtu";
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary=wtu
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com
   C:
   C: <search id="xyz12346"/>
   C:    <select>
   C:       <source relcalid="opaqueid101"/>
   C:       <source relcalid="opaqueid103"/>
   C:       <data content="cid:query@cal.example.com"/>
   C:    </select>
   C: </search>
   C: --boundary=wtu
   C: Content-Type: text/calendar
   C: Content-ID: query@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: BEGIN:VQUERY
   C: QUERY:SELECT DTSTART,DTEND,SUMMARY,UID FROM VEVENT
   C: WHERE DTEND >= 19990714T080000Z AND
   C:   DTSTART <= 19990715T080000Z
   C: END:VQUERY
   C: END:VCALENDAR
   C: --boundary=wtu--
   C: END

   S: ANS 1 17 . 18804 717 0
   S: Content-Type: multipart/related; boundary="boundary-09436";
   S:    start="command@cal.example.com";
   S:    type="application/beep+xml"
   S:
   S: --boundary-09436
```

```
   S: Content-Type: application/beep+xml
   S: Content-ID: command@cal.example.com
   S:
   S: <result id="xyz12346">
   S:    <source relcalid="opaqueid103"/>
   S:    <request-status code="2.0"/>
   # this response code means that the transport successfully
   # delivered the data.
   S:    <data content="cid:2@cal.example.com"/>
   S: </result>
   S: --boundary-09436
   S: Content-Type: text/calendar
   S: Content-ID: 2@cal.example.com
   S:
   S: BEGIN:VCALENDAR
   S: VERSION:2.1
   S: BEGIN:VEVENT
   S: UID:0034848098038888989443
   S: SUMMARY:meeting
   S: DTEND:19990714T233000Z
   S: DTSTART:19990714T223000Z
   S: END:VEVENT
   S: END:VCALENDAR
   S: --boundary-09436--
   S: END

   S: ANS 1 18 . 19521 216 1
   S: Content-Type: application/beep+xml
   S:
   S: <result id="xyz12346">
   S:    <source relcalid="opaqueid101"/>
   S:    <request-status code="4.1">Access Denied<request-status/>
   S:    <data content="cid:2@cal.example.com"/>
   S: </result>
   S: END

   S: NUL 1 18 . 19737 0
   S: END
```

Stored VQUERY can be used by specifying the property QUERYNAME instead of QUERY.

Example:

```
           BEGIN:VQUERY
           QUERYNAME:StoredVQuery-1
           END:VQUERY
```

This match all calendar store properties. This MUST NOT return any VAGENDAs.

```
             BEGIN:VCALENDAR
             VERSION:2.1
             BEGIN:VQUERY
             QUERY:SELECT * FROM CALSTORE WHERE CSID='bobo.ex.com'
             END:VQUERY
             END:VCALENDAR
```

This will match all properties of the VAGENDA relcal4. This MUST NOT return any components.

```
             BEGIN:VCALENDAR
             VERSION:2.1
             BEGIN:VQUERY
             QUERY:SELECT * FROM VAGENDA WHERE RELCALID='relcal4'
             END:VQUERY
             END:VCALENDAR
```

This will fetch all stored VQUERYs.

```
             BEGIN:VCALENDAR
             VERSION:2.1
             BEGIN:VQUERY
             QUERY:SELECT * FROM VQUERY WHERE QUERYNAME IS NOT NULL
             END:VQUERY
             END:VCALENDAR
```

6.3 Scheduling Commands

Scheduling commands allow a CU to indirectly manipulate a calendar by asking another CU to perform an operation on their calendar. For example, CU-A can request CU-B to add a meeting to their calendar; in effect inviting CU-B to the meeting.

Calendar access rights can be granted for scheduling commands without granting rights for more generalized access with the calendar commands.

[EDITORS NOTE: This section needs to be completed by adding the restriction tables for each of these iTIP methods. The basis for the text is to be taken from [iTIP].]

6.3.1 "schedule" Command

Attributes:

"id" (see Section 6.2.2.1).

Elements:

"max-time": See Section 3.3.

"target": Each "target" element points to a container where the sceduled element will be created.

"data": MUST point to a valid iTip iCalendar object. Refer to [iTIP] for the definition of the accepted METHOD and restriction tables.

Response:

One "result" message per "target" element MUST be returned (see Section 3.1).

One of the following "request-status" codes MUST be returned:

2.0 - Success

6.1 - Container not found

6.3 - Bad args

Additionnal request-status code MAY be returned as described on [iTIP].

The "data" element of each "result" message is subject to the result restriction table defined below.

The "schedule" command insert a new scheduled component into the VSCHEDULE set of the container(s) referred to by the "target" element(s).

A Calendar Service MUST accept iCalendar object with the following METHODs as described in [iTIP]:

Command	Description
PUBLISH	Publish a calendar entry to one or more calendars.
REQUEST	Schedule a calendar entry with one or more calendars.
REPLY	Response to a scheduling request.
ADD	Add one or more instances to an existing

	calendar entry.
CANCEL	Cancel one or more instances to an existing calendar entry.
REFRESH	A request for the latest version of a calendar entry.
COUNTER	A request for a change (a counter-proposal) to a calendar entry.
DECLINECOUNTER	Decline a counter proposal.

6.3.2 Processing Scheduling Components

A CU might be invited to a meeting. If the CU had been invited by CAP, the entry in the CU calendar will be scheduled, but not booked. So a CUA will need to look for scheduled entries in the calendar and present them to the CU or automatically decide if the invitation is to be accepted or processed.

Example:

The little league coach places the teams entire schedule into your calendar. Lets say that every game and practice is on a Friday night and your calendar now has this iTIP scheduling data:

```
            BEGIN:VCALENDAR
            VERSION:2.0
            METHOD:PUBLISH
            BEGIN:VEVENT
            DTSTAMP;TZID=US/Pacific:20000229T180000
            DTSTART;TZID=US/Pacific:20000303T180000
            ORGANIZER:coach@little.league.com
            SUMMARY:Schedule of games and practice
            UID:1-coach@little.league.com
            SEQUENCE:0
            DESCRIPTION:Please have your child at the field
             and ready to play by 6pm.
            RRULE:FREQ=WEEKLY;COUNT=10
            END:VEVENT
            END:VCALENDAR
```

At this point the above VEVENT is not booked in your calendar; it is simply scheduled. A CUA would fetch this and all other scheduled VEVENTs from your calendar with:

```
C: MSG 1 19 . 19737 582
C: Content-Type: multipart/related; boundary="boundary-rtij41";
C:    start="1@cal.example.com";
```

```
   C:    type="application/beep+xml"
   C:
   C: --boundary-rtij41
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com
   C:
   C: <search id="xyz12345"/>
   C:    <select>
   C:       <source relcalid="relcal2"/>
   C:       <data content="cid:query@cal.example.com"/>
   C:    </select>
   C: </search>
   C: --boundary-rtij41
   C: Content-Type: text/calendar
   C: Content-ID: query@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: BEGIN:VQUERY
   C: QUERY:SELECT * WHERE FROM VEVENT METHOD IS NOT NULL
   C: END:VQUERY
   C: END:VCALENDAR
   C: --boundary-rtij41--
   C: END
```

The CUA and CU could determine which scheduling entries were to remain on the calendar. Each scheduling component could be deleted or updated with the "modify" command to remove the iTip METHOD.

In some cases the CUA could automatically do the work and inform the CU. An example of this is CANCEL. If configured to process METHOD:CANCEL it could execute the "delete" command to delete the component and inform the CU that the booked component had been canceled.

The CUA MUST process the scheduled components in the order sent. Some optimization could be done by the CUA. One example is if a PUBLISH and later a CANCEL for the same component with the same SEQUENCE number were scheduled, but not booked. The CUA might never inform the CU and treat it as if the PUBLISH had never been received by doing a "delete" command on both entries.

It is important to note that scheduled components do not show up in busy time as BUSY. Only when they are booked does the TRANSP:OPAQUE property take effect. A CS implementation MAY mark the time as TENTATIVE. This is an implementation and administrative choice.

The CS MAY automatically process some iTIP request. For example a CS MAY automatically send out REFRESH replies via iMIP or CAP, then

delete the REFRESH entry. But only if there are no other pending scheduled entries for this calendar that may affect what REFRESH would send back. If the CS is not able to reply to the REFRESH request then it is left in the scheduling set until the CUA and CU processes the set. At the point where there are no outstanding scheduled command that would effect the reply results, the CS may then automatically send the reply to the REFRESH request.

6.3.3 iTIP Examples

The following examples describe scenarios for the handling of incoming iTIP data. An appropriate sort-order for the handling of incoming iTIP is by UID, Recurrence-id, sequence, dtstamp. This processing may be optimized, for instance, REFRESHs could be processed last.

As an update to [iTIP], data with the "COUNTER" method should be processed even if the Sequence number is stale.

6.3.3.1 Sending and Receiving an iTIP request

In this example A invites B and C to a meeting, B accepts the meeting and C rejects it. The calendars for A, B and C are relcal1, relcal2 and relcal3 respectively, and are all on the same server, "cal.example.com". A lot of these described actions are performed by the CUAs and not the users themselves, the CUAs are called A-c, B-c and C-c respectively.

A wishes to create a meeting with B and C, so A-c uses CAP to send the following iTIP request to relcal2 and relcal3, while logged in to "cal.example.com".

```
C: MSG 1 20 . 22254 874
C: Content-Type: multipart/related; boundary="boundary-rewf4";
C:    start="1@cal.example.com;
C:    type="application/beep+xml"
C:
C: --boundary-rewf4
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <schedule id="xhj-dd"/>
C:    <target relcalid="relcal3"/>
C:    <target relcalid="relcal2"/>
C:    <select>
C:       <data content="cid:request@cal.example.com"/>
C:    </select>
C: </schedule>
```

```
C: --boundary-rewf4
C: Content-Type: text/calendar
C: Content-ID: request@cal.example.com
C:
C: BEGIN:VCALENDAR
C: VERSION:2.1
C: METHOD:REQUEST
C: BEGIN:VEVENT
C: UID:abcd12345
C: DTSTART:19990307T180000Z
C: DTEND:19990307T190000Z
C: ORGANIZER:cap://cal.example.com/relcal1
C: ATTENDEE;RSVP=TRUE;
C:   PARTSTAT=NEEDS-ACTION:cap://cal.example.com/relcal2
C: ATTENDEE;RSVP=TRUE;
C:   PARTSTAT=NEEDS-ACTION:cap://cal.example.com/relcal3
C: SUMMARY:Important Meeting
C: END:VEVENT
C: END:VCALENDAR
C: --boundary-rewf4--
C: END
```

An incoming event (indicated by the value of the "METHOD" property) then appears in relcal2 and relcal3, with the following data:

```
            BEGIN:VEVENT
            METHOD:REQUEST
            UID:abcd12345
            DTSTART:19990307T180000Z
            DTEND:19990307T190000Z
            ORGANIZER:cap://cal.example.com/relcal1
            ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-ACTION:cap://cal.example.c
             om/relcal2
            ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-ACTION:cap://cal.example.c
             om/relcal3
            SUMMARY:Important Meeting
            END:VEVENT
```

B-c and C-c must search for such incoming events, they do so using the following CAP search:

```
C: MSG 1 21 . 24655 631
C: Content-Type: multipart/related; boundary="boundary-ytem";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-ytem
C: Content-Type: application/beep+xml
```

```
   C: Content-ID: 1@cal.example.com;
   C:
   C: <search id="xhr-de">
   C:   <max-time latency=60 action=ask/>
   C:   <select>
   C:     <source relcalid='relcal2'/>
   # or relcalid='relcal3'
   C:     <data content="cid:2@cal.example.com"/>
   C:   </select>
   C: </search>
   C: --boundary-ytem
   C: Content-Type: text/calendar
   C: Content-ID: 2@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: BEGIN:VQUERY
   C: QUERY:SELECT * FROM VEVENT WHERE METHOD = 'REQUEST';
   C: END:VQUERY
   C: END:VCALENDAR
   C: --boundary-ytem--
   C: END
```

In response to this search they get the above event. B-c and C-c must then open the VEVENT, find the UID and determine if there is already an event on their calendar with that UID. To do this they use the following search:

```
   C: MSG 1 22 . 26087 654
   C: Content-Type: multipart/related; boundary="boundary-rtylk";
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary-rtylk
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com;
   C:
   C: <search id="xhr-df">
   C:   <max-time latency=60 action=ask/>
   C:   <select>
   C:     <source relcalid='relcal2'/>
   # or relcalid='relcal3'
   C:     <data content="cid:2@cal.example.com"/>
   C:   </select>
   C: </search>
   C: --boundary-rtylk
   C: Content-Type: text/calendar
   C: Content-ID: 2@cal.example.com
```

```
C:
C: BEGIN:VCALENDAR
C: VERSION:2.1
C: BEGIN:VQUERY
C: QUERY:SELECT * FROM VEVENT WHERE UID = 'abcd12345' AND
C:  METHOD IS NULL
C: END:VQUERY
C: END:VCALENDAR
C: --boundary-rtylk--
C: END
```

We assume that the event is not already in their relcal2 or relcal3.

B-c prompts B who decides to accept the meeting request, and B-c creates a copy of the event in relcal2, with the "PARTSTAT" parameter set to ACCEPTED. B-c also sends this copy to the Organizer at relcal1 as an iTIP REPLY, preserving the CMDID:

```
C: MSG 1 23 . 26741 697
C: Content-Type: multipart/related; boundary="boundary-1943";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-1943
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com;
C:
C: <schedule id="xhj-dd">
C:   <target relcalid="relcal1">
C:   <data content="cid:2@cal.example.com"/>
C: </schedule>
C: --boundary-1943
C: Content-Type: text/calendar
C: Content-ID: 2@cal.example.com
C:
C: BEGIN:VCALENDAR
C: VERSION:2.1
C: METHOD:REPLY
C: BEGIN:VEVENT
C: UID:abcd12345
C: DTSTART:19990307T180000Z
C: DTEND:19990307T190000Z
C: ORGANIZER:cap://cal.example.com/relcal1
C: ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.example.com/relcal2
C: SUMMARY:Important Meeting
C: END:VEVENT
C: END:VCALENDAR
C: --boundary-1943--
```

```
   C: END
```

C, on the other hand, decides to decline the meeting, and C-c sends a reply to the Organizer to that effect, as follows:

```
   C: MSG 1 24 . 27438 705
   C: Content-Type: multipart/related; boundary="boundary-oiudfc";
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary-oiudfc
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com;
   C:
   C: <schedule id="xhj-de">
   C:   <target relcalid="relcal1">
   C:   <data content="cid:2@cal.example.com"/>
   C: </schedule>
   C: --boundary-oiudfc
   C: Content-Type: text/calendar
   C: Content-ID: 2@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: METHOD:REPLY
   C: BEGIN:VEVENT
   C: UID:abcd12345
   C: DTSTART:19990307T180000Z
   C: DTEND:19990307T190000Z
   C: ORGANIZER:cap://cal.example.com/relcal1
   C: ATTENDEE;PARTSTAT=DECLINED:cap://cal.example.com/relcal3
   C: SUMMARY:Important Meeting
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-oiudfc--
   C: END
```

It is preferable that C-c stores the event in relcal3 even though it has been declined. Storing the event in relcal3 allows subsequent iTIP messages to be interpreted correctly. The "PARTSTAT" parameter indicates that the event was refused.

After receiving the replies from relcal2 and relcal3, A-c updates the version of the event in relcal1 to indicate the new participation status:

```
   C: MSG 1 25 . 29450 934
   C: Content-Type: multipart/related; boundary="boundary-wer3";
```

```
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary-wer3
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.cal.example.com
   C:
   C: <modify id="modify01"/>
   C:    <select>
   C:       <source relcalid="relcal1"/>
   C:       <data content="cid:query@cal.example.com"/>
   C:    </select>
   C:    <update remove-missing=false>
   C:       <data content="cid:update@cal.example.com"/>
   C:    </update>
   C: </modify>
   C: --boundary-wer3
   C: Content-Type: text/calendar
   C: Content-ID: query@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: BEGIN:VQUERY
   C: QUERY: SELECT * FROM VEVENT WHERE UID='abcd12345'
   C: END:VQUERY
   C: END:VCALENDAR
   C: --boundary-wer3
   C: Content-Type: text/calendar
   C: Content-ID: update@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: BEGIN:VEVENT
   C: ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.example.com/relcal2
   C: ATTENDEE;PARTSTAT=DECLINED:cap://cal.example.com/relcal3
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-wer3--
   C: END
```

A-c then sends a new iTIP request to relcal2 only, indicating the updated information.

6.3.3.2 Handling an iTIP refresh

A little bit later, C is thinking about accepting the event in the previous example, but first wants to check the current state of the event. To find the current state C-c uses the iTIP "REFRESH" method, sending the following to relcal1:

```
   C: MSG 1 26 . 31005 649
   C: Content-Type: multipart/related; boundary="boundary-fsdk3";
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary-fsdk3
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com
   C:
   C: <schedule id="xud-pn"/>
   C:    <target relcalid="relcal1"/>
   C:    <data content="cid:refresh@cal.example.com"/>
   C: </schedule>
   C: --boundary-fsdk3
   C: Content-Type: text/calendar
   C: Content-ID: refresh@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: METHOD:REFRESH
   C: BEGIN:VEVENT
   C: UID:abcd12345
   C: ORGANIZER:cap://cal.example.com/relcal1
   C: ATTENDEE:cap://cal.example.com/relcal3
   C: DTSTAMP:19990306T202333Z
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-fsdk3--
   C: END
```

A-c finds the refresh as an incoming iTIP, and searches for the corresponding event. Having found the event (with no changes since the last example) A-c then verifies that relcal3 is in fact an attendee of the event and is thus allowed to request a refresh. (In the case of a published event things are more complicated.) A-c packages the event as an iTIP request and sends it to relcal3:

```
   C: MSG 1 27 . 32541 856
   C: Content-Type: multipart/related; boundary="boundary-trekvg";
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary-trekvg
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com
   C:
   C: <schedule id="xud-pn"/>
   C:    <target relcalid="relcal3"/>
```

```
   C:    <data content="cid:request@cal.example.com"/>
   C: </schedule>
   C: --boundary-trekvg
   C: Content-Type: text/calendar
   C: Content-ID: request@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: METHOD:REQUEST
   C: TARGET:cap://cal.example.com/relcal3
   C: BEGIN:VEVENT
   C: UID:abcd12345
   C: DTSTART:19990307T180000Z
   C: DTEND:19990307T190000Z
   C: ORGANIZER:cap://cal.example.com/relcal1
   C: ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.example.com/relcal2
   C: ATTENDEE;PARTSTAT=DECLINED:cap://cal.example.com/relcal3
   C: SUMMARY:Important Meeting
   C: SEQUENCE:0
   C: DTSTAMP:19990306T204333Z
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-trekvg--
   C: END
```

6.3.3.3 Sending and accepting an iTIP counter

Having received the latest copy of the event C wishes to propose a venue for the event, using an iTIP COUNTER. To do this C-c sends the following to relcal1:

```
   C: MSG 1 28 . 34587 883
   C: Content-Type: multipart/related; boundary="boundary-werf";
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary-werf
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com
   C:
   C: <schedule id="zzykjjk"/>
   C:    <target relcalid="relcal1"/>
   C:    <data content="cid:counter@cal.example.com"/>
   C: </schedule>
   C: --boundary-werf
   C: Content-Type: text/calendar
   C: Content-ID: counter@cal.example.com
```

```
C:
C: BEGIN:VCALENDAR
C: VERSION:2.1
C: METHOD:COUNTER
C: BEGIN:VEVENT
C: UID:abcd12345
C: DTSTART:19990307T180000Z
C: DTEND:19990307T190000Z
C: SEQUENCE:0
C: ORGANIZER:cap://cal.example.com/relcal1
C: ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.example.com/relcal2
C: ATTENDEE;PARTSTAT=DECLINED:cap://cal.example.com/relcal3
C: SUMMARY:Important Meeting
C: LOCATION:La Belle Province
C: COMMENT:My favorite restaurant, I'll definitely go if it's
C:   there.
C: END:VEVENT
C: END:VCALENDAR
C: --boundary-werf--
C: END
```

Having sent the information to relcal1, C-c shouldn't store the new details in relcal3. If C-c updated the version in relcal3 and relcal1 did not reply to the counter, then relcal3 would have incorrect information. Instead C-c preserves the correct information and waits for a response from relcal1. A CUA implementation may wish to preserve this information itself, externally to the CS.

In order to receive an iTIP counter A-c follows the same search as for other iTIP data, first find the incoming message, next find any matching events in the calendar store.

Having found the matching event, A reviews the proposed changes and decides to accept the COUNTER. To do this, A-c modifies the version in relcal1 (bumping the sequence number) to:

```
C: MSG 1 29 . 37650 850
C: Content-Type: multipart/related; boundary="boundary-kmcrf";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-kmcrf
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.cal.example.com
C:
C: <modify id="asdf123"/>
C:    <select>
C:       <source relcalid="relcal1"/>
```

```
C:       <data content="cid:query@cal.example.com"/>
C:    </select>
C:    <update remove-missing=false>
C:       <data content="cid:update@cal.example.com"/>
C:    </update>
C: </modify>
C: --boundary-kmcrf
C: Content-Type: text/calendar
C: Content-ID: query@cal.example.com
C:
C: BEGIN:VCALENDAR
C: BEGIN:VQUERY
C: QUERY: SELECT * FROM VEVENT WHERE UID='abcd12345'
C: END:VQUERY
C: END:VCALENDAR
C: --boundary-kmcrf
C: Content-Type: text/calendar
C: Content-ID: update@cal.example.com
C:
C: BEGIN:VCALENDAR
C: BEGIN:VEVENT
C: LOCATION:La Belle Province
C: SEQUENCE:1
C: END:VCALENDAR
C: --boundary-kmcrf--
C: END
```

A-c then sends the updated version as a request to both relcal2 and relcal3:

```
C: MSG 1 30 . 39450 909
C: Content-Type: multipart/related; boundary="boundary-plmng";
C:    start="1@cal.example.com";
C:    type="application/beep+xml"
C:
C: --boundary-plmng
C: Content-Type: application/beep+xml
C: Content-ID: 1@cal.example.com
C:
C: <schedule id="xup-po"/>
C:    <target relcalid="relcal2"/>
C:    <target relcalid="relcal3"/>
C:    <data content="cid:request@cal.example.com"/>
C: </schedule>
C: --boundary-plmng
C: Content-Type: text/calendar
C: Content-ID: request@cal.example.com
C:
```

```
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: METHOD:REQUEST
   C: BEGIN:VEVENT
   C: UID:abcd12345
   C: DTSTART:19990307T180000Z
   C: DTEND:19990307T190000Z
   C: ORGANIZER:cap://cal.example.com/relcal1
   C: ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-
   C:  ACTION:cap://cal.example.com/relcal2
   C: ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-
   C:  ACTION:cap://cal.example.com/relcal3
   C: SUMMARY:Important Meeting
   C: LOCATION:La Belle Province
   C: SEQUENCE:1
   C: DTSTAMP:19990307T054339Z
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-plmng--
   C: END
```

6.3.3.4 **Declining an iTIP counter**

B does not like the new location and also counters the event, B-c sends the following iTIP:

```
   C: MSG 1 31 . 41620 762
   C: Content-Type: multipart/related; boundary="boundary-cafe3";
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary-cafe3
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com
   C:
   C: <schedule id="xim-ef"/>
   C:    <target relcalid="relcal1"/>
   C:    <data content="cid:counter@cal.example.com"/>
   C: </schedule>
   C: --boundary-cafe3
   C: Content-Type: text/calendar
   C: Content-ID: counter@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: METHOD:COUNTER
   C: BEGIN:VEVENT
```

```
   C: UID:abcd12345
   C: DTSTART:19990307T180000Z
   C: DTEND:19990307T190000Z
   C: ORGANIZER:cap://cal.example.com/relcal1
   C: ATTENDEE:cap://cal.example.com/relcal2
   C: ATTENDEE:cap://cal.example.com/relcal3
   C: SUMMARY:Important Meeting
   C: LOCATION:Guy et Dodo
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-cafe3--
   C: END
```

However, C does not accept the counter, and C-c replies with a decline counter:

```
   C: MSG 1 32 . 42901 631
   C: Content-Type: multipart/related; boundary="boundary-meme34";
   C:    start="1@cal.example.com";
   C:    type="application/beep+xml"
   C:
   C: --boundary-meme34
   C: Content-Type: application/beep+xml
   C: Content-ID: 1@cal.example.com
   C:
   C: <schedule id="xim-ef"/>
   C:    <target relcalid="relcal2"/>
   C:    <data content="cid:decline-counter@cal.example.com"/>
   C: </schedule>
   C: --boundary-meme34
   C: Content-Type: text/calendar
   C: Content-ID: decline-counter@cal.example.com
   C:
   C: BEGIN:VCALENDAR
   C: VERSION:2.1
   C: METHOD:DECLINE-COUNTER
   C: BEGIN:VEVENT
   C: DTSTAMP:19990307T093245Z
   C: UID:abcd12345
   C: ORGANIZER:cap://cal.example.com/relcal1
   C: SEQUENCE:1
   C: END:VEVENT
   C: END:VCALENDAR
   C: --boundary-meme34--
   C: END
```

CUA-b MUST keep the original information when sending the counter, and there is no problem when no information is returned in the

DECLINE-COUNTER.

7. Response Codes

Numeric response codes are returned at both the transfer and application layer. The same set of codes is used in both cases.

[EDITORS NOTE: Do we want to use the same set of codes?]

The format of these codes is described in [iCAL], and extend in [iTIP] and [iMIP]. The following describes new codes added to this set.

At the application layer response codes are returned as the value of a "REQUEST-STATUS" property. The value type of this property is modified from that defined in [iCAL], to make the accompanying text optional.

Code	Description
2.0	Success. The parameters vary with the operation and are specified.
2.0.3	In response to the client issuing an "abort" reply, this reply code indicates that any command currently underway was successfully aborted.
3.1.4	Capability not supported.
4.1	Calendar store access denied.
6.3	Attempt to create or modify an event such that it would overlap another event in either of the following two circum-stances: (a) One of the events has a TRANSP property set to OPAQUE-NOCONFLICT or TRANSPARENT-NOCONFLICT. (b) The calendar's ALLOW-CONFLICT property is set to NO.
6.XXX	[EDITORS NOTE: More are in this memo - add here TODO]
7.0	A timeout has occurred. The server was unable to complete the operation in the requested time.

8.0	A failure has occurred in the Calendar Service that prevents the operation from succeeding.
8.2	Used to signal that an iCalendar object has exceeded the server's size limit
8.3	A DATETIME value was too far in the future represented on this Calendar.
8.4	A DATETIME value was too far in the past to be represented on this Calendar.
8.5	An attempt was made to create a new object but the unique id specified is already in use.
9.0	An unrecognized command was received.
10.4	The operation has not been performed because it would cause the resources (memory, disk, CPU, etc) to exceed the allocated quota.

----

8. BEEP Profile Registration

Profile Identification:
http://iana.org/beep/transient/calsch/cap/1.0

Messages exchanged during Channel Creation: none

Messages starting one-to-one exchanges:

"timeout", "generate-uid", "identify", "get-capability"

Messages in positive replies:

"uid-list", "abort", "continue", "result", "capability"

Messages in negative replies:

"error"

Messages in one-to-many exchanges: "create", "search", "delete", "modify" or "schedule"

Message Syntax: c.f., Section 9

Message Semantics: c.f., Section 6

Contact Information: c.f., the "Author's Address" section of this memo

9. CAP DTD

```
   <!--
     DTD for CAP commands and responses.
   -->

   <!ENTITY % CAP PUBLIC "-//IETF/DTD CAP//EN" "">
   %CAP;

   <!--
     DTD data types:

         entity      syntax/reference        example
         ======      ================        =======
         UPN         c.f. UPN Section 1.3    mary@cal.example.com
         CMDID       string                  read_12321
         SECONDS     1..2147483647           60
         CODE        1*DIGIT *("." 1*DIGIT)  2.0
         COUNT       1..32768                1
         CSID        c.f. CSID Section 1.3   cap://cap.example.com:5229
         DEPTH       '*' | 1..2147483647     1
   -->


   <!ENTITY % UPN       "CDATA">
   <!ENTITY % CMDID     "CDATA">
   <!ENTITY % SECONDS   "CDATA">
   <!ENTITY % CODE      "CDATA">
   <!ENTITY % COUNT     "CDATA">
   <!ENTITY % RELCALID  "CDATA">
   <!ENTITY % CSID      "CDATA">
   <!ENTITY % DEPTH     "CDATA">


   <!ELEMENT create (max-time?,target+,data)>
   <!ATTLIST create id %CMDID; #IMPLIED>

   <!ELEMENT search (max-time?,select,max-results?)>
   <!ATTLIST search id %CMDID; #IMPLIED>

   <!ELEMENT max-size (#PCDATA)>
   <!ELEMENT max-results (#PCDATA)>

   <!ELEMENT delete (max-time?,select)>
   <!ATTLIST delete id %CMDID; #IMPLIED>

   <!ATTLIST modify (max-time?,select,add,delete,update)>
```

```
   <!ATTLIST modify id %CMDID; #IMPLIED>

   <!ELEMENT itip (max-time?,data)>
   <!ATTLIST itip id %CMDID; #IMPLIED>

   <!ELEMENT target EMPTY>
   <!ATTLIST target relcalid %RELCALID; #IMPLIED
                    csid     %RELCALID; #IMPLIED>

   <!ELEMENT max-time EMPTY>
   <!ATTLIST max-time latency %SECONDS; #REQUIRED>
                      action (ask|abort) "abort">

   <!ELEMENT data EMPTY>
   <!ATTLIST data content %URI; #REQUIRED>

   <!ELEMENT select (data,source+)>

   <!ELEMENT source EMPTY>
   <!ATTLIST source relcalid %RELCALID; #IMPLIED
                    csid     %CALID;    #IMPLIED
                    depth    %DEPTH;    "0">

   <!ELEMENT remove (data)>
   <!ATTLIST remove ignore-value (true|false) "false">

   <!ELEMENT update (data)>
   <!ATTLIST update remove-missing (true|false) "false">

   <!ELEMENT generate-uid EMPTY>
   <!ATTLIST generate-uid num %COUNT; '1'>

   <!ELEMENT uid-list (uid*)>
   <!ELEMENT uid (#PCDATA)>

   <!ELEMENT identify EMPTY>
   <!ATTLIST identify upn %UPN; #REQUIRED>

   <!ELEMENT timeout EMPTY>
   <!ATTLIST timeout id %CMDID; #IMPLIED>

   <!ELEMENT abort EMPTY>
   <!ELEMENT continue (timeout?)>

   <!ELEMENT get-capability EMPTY>

   <!ELEMENT capability (cap,itip,icalendar,date?)>
```

```
   <!ELEMENT version (#PCDATA)>

   <!ELEMENT icalendar (version,max-component-size?)>
   <!ELEMENT max-component-size (#PCDATA)>
   <!ELEMENT query-level (#PCDATA)>

   <!ELEMENT itip (version)>

   <!ELEMENT cap (version,query-level,car?)>

   <!ELEMENT car (#PCDATA)>
   <!ELEMENT query-level (#PCDATA)>

   <!ELEMENT date (max?,min?)>
   <!ELEMENT min (#PCDATA)>
   <!ELEMENT max (#PCDATA)>

   <!ELEMENT result (data?,target?,source?,request-status)>
   <!ATTLIST result id   %CMDID; #IMPLIED>

   <!ELEMENT request-status (#PCDATA)>
   <!ATTLIST request-status code %CODE; #REQUIRED>

   <!ELEMENT error (request-status)>
```

10. Implementation Issues

1. What are the minimum component properties set required to create a new VEVENT, VTODO and VJOURNAL?. PROPOSAL: DTSTART, SUMMARY and UID.

[EDITORS NOTE (dr): They MUST be the same as for iTIP]

2. What is the state of all undefined properties? PROPOSAL: Not defined. So a query will not return them, if they are selected.

[EDITORS NOTE (dr): Many have default values, a CS may return the default values?]

11. Properties

[EDITORS NOTE: These extensions/changes to iCalendar need to be reformatted to conform to the IANA registration process defined in section 7 of [iCAL].]

11.1 Calendar Store Properties

The following are properties of the calendar store.

Name	Read Only	Value Type	Description
CALMASTER	N	URI	The e-mail address for a responsible person. MUST be a mailto URL.
CSID	Y	URI	The CSID of this calendar store. If not specified, it is the same as the hostname.
DEFAULT_VCARS	N	TEXT	A multivalued property containing the default VCARs for newly created top level calendars. Each entry is a CARID. It MUST include at a minimum READBUSYTIMEINFO,REQUESTONLY, UPDATEPARTSTATUS, and DEFAULTOWNER.
MAXDATE	Y	DATE-TIME	The date/time in the future beyond which the server cannot represent. If not specified, then 99991231T235959 will be assumed.
MINDATE	Y	DATE-TIME	The date/time in the past prior to which the server cannot represent. If not specified, then 00000101T000000 will be assumed.
CURRENT_DATETIME	Y	DATE-TIME	Current server time. This is returned as a local time and TZID.
RECUR_ACCEPTED	Y	BOOLEAN	Boolean value will be set to

			TRUE if the server will accept recurrence rules. It will be set to FALSE if the server will not accept recurrence rules. If not specified a CUA MUST assume TRUE.
RECUR_EXPAND	Y	BOOLEAN	If set to TRUE, the CS supports the expansion of recurrence rules. If set to FALSE, the CS is incapable of expanding recurrence rules. If not specified a CUA MUST assume TRUE.
RECUR_LIMIT	Y	INTEGER	This numeric value describes how the server handles unbounded recurrences. The value is only valid if RECURRENCE is TRUE. If the value is 0 it means that the server supports unbounded recurrence rules. If it is non-zero, it is a positive integer indicating the number of instances that will be created when the server expands an unbounded recurrence rule when fetched from the CS. A CUA MUST query for date ranges when this value is zero.
VERSION	Y	TEXT	The version of the CS. The default and the only currently Supported version is "2.0" to match the [iCAL] VERSION.

11.2 Calendar Properties

Name	Read Only	Value Type	Description
ALLOW-CONFLICT	N	BOOLEAN	This boolean value indicates Whether or not the calendar supports event conflicts. That is, whether or not any of the events in the calendar can

			overlap. If not specified the default value is TRUE meaning that conflicts are allowed.
CALSCALE	N	TEXT	The CALSCALE for this calendar. If not specified the default is GREGORIAN.
CHARSET	N	TEXT	The default charset for Localized strings in this calendar. If not specified, the default is UTF-8.
CHILDREN	Y	TEXT	The list of sub-calendars Belonging to this calendar. An empty list means no children. The results may be a comma separated list of children. Each entry returned is a CALID. The default is an empty list.
CREATED	Y	DATE-TIME	The timestamp of the calendar's create date.
DEFAULT_VCARS	N	TEXT	The default VCARs for newly Created top level calendars. This is a CARID. The default value is the value of DEFAULT_VCARS in the CALSTORE table.
LANGUAGE	N	TEXT	The default language for localizable strings in this calendar. There is no default. This value MUST NOT be empty.
LAST-MODIFIED	N	DATE-TIME	The timestamp when the Properties of the calendar were last updated. Default is the same as CREATED.
NAME	N	TEXT	The display name for this calendar. It is a localizable string. If not provided, an empty value will be returned.
OWNER	N	URI	A multi-instanced property indicating the calendar owner.

			Each entry returned will be a UPN. There must be at least one owner.
PARENT	N	URI	The CALID of this calendar's Parent maintained by a CAP server. An empty value means the calendar is the top level parent. The default value is no parent.
RELCALID	N	URI	A unique identifier for the calendar. There is no default value and This value MUST NOT be empty.
TOMBSTONE	N	BOOLEAN	TRUE indicator that this Calendar has been marked as deleted. The default value is FALSE.
TZID	N	TEXT	The id of the timezone Associated with this calendar. See TZID in [iCAL]. The default value is UTC.

12. CAP Item Registration

This section provides the process for registration of new or modified CAP entities.

12.1 Registration of New and Modified CAP Entities

New CAP entities are registered by the publication of an IETF Request for Comment (RFC). Changes to a CAP item are registered by the publication of a revision of the RFC defining the method.

12.2 Registration of New Entities

This section defines procedures by which new entities (i.e., components, properties, parameters, enumerated values or restriction tables) for a CAP item can be registered with the IANA.

Non-standard, experimental entities can be used by bilateral agreement, provided the associated properties names follow the "X-" convention. Such non-standard and experimental entities are non-IANA entities and need not be registered using this process.

The procedures defined here are designed to allow public comment and review of new CAP entities, while posing only a small impediment to the definition of new properties.

Registration of a new CAP item is accomplished by the following steps.

12.2.1 Define the Item

A CAP item is defined by completing the following template.

```
            To: ietf-calendar@imc.org
            Subject: Registration of CAP item XXX
            Item name:
            Item purpose:
            Description:
            CAP terminology changes:
            CAP data model changes:
            CAP system model changes:
            Conformance considerations:
            Format definition:
            Examples:
```

The meaning of each field in the template is as follows.

   Item name: The name of the item.

Item purpose: The purpose of the item (e.g., Extends the CAP command set to poll for notifications, etc.). Give a short but clear description.

Description: Any special notes about the item, how it is to be used, etc.

CAP terminology changes: Any change or additions to the existing CAP terminology needs to be specified.

CAP data model changes: Any of the valid property parameters for the property needs to be specified.

CAP system model changes:

Conformance: A clear summary of how and where this CAP item extension MUST, MAY, SHOULD or can be used. Any changes or impact to the existing conformance definition for CAP should be explained. The impact to implementations conforming to the existing CAP specification should be clearly described.

Format definition: The ABNF for each element of the CAP item needs to be specified.

Examples: One or more examples of instances of the CAP item and each of its usage scenarios needs to be specified.

12.2.2 Post the item definition

The item description MUST be posted to the new item discussion list, ietf-calendar@imc.org.

12.2.3 Allow a comment period

Discussion on the new item MUST be allowed to take place on the list for a minimum of two weeks. Consensus MUST be reached on the property before proceeding to the next step.

12.2.4 Submit the proposal for approval

Once the two-week comment period has elapsed, and the proposer is convinced consensus has been reached on the proposal, the registration application should be submitted to the Method Reviewer for approval. The Method Reviewer is appointed by the Application Area Directors and can either accept or reject the proposal registration. An accepted registration should be passed on by the Method Reviewer to the IANA for inclusion in the official IANA method

registry. The registration can be rejected for any of the following reasons. 1) Insufficient comment period; 2) Consensus not reached; 3) Technical deficiencies raised on the list or elsewhere have not been addressed. The Method Reviewers decision to reject a proposal can be appealed by the proposer to the IESG, or the objections raised can be addressed by the proposer and the proposal resubmitted.

[EDITORS NOTE: John Stracke to review any updates]

12.3 Property Change Control

Existing CAP entities can be changed using the same process by which they were registered.

1. Define the change

2. Post the change

3. Allow a comment period

4. Submit the proposal for approval

Note that the original author or any other interested party can propose a change to an existing CAP object, but that such changes should only be proposed when there are serious omissions or errors in the published memo. The Method Reviewer can object to a change if it is not backward compatible, but is not required to do so.

CAP objects definitions can never be deleted from the IANA registry, but objects which are no longer believed to be useful can be declared OBSOLETE by adding this text to their "Item purpose" field.

13. IANA Considerations

This memo defines IANA registered extensions to the attributes defined by iCalendar, as defined in [iCAL], and [iTIP].

IANA registration proposals for iCalendar and iTIP are to be mailed to the registration agent for the "text/calendar" [MIME] content-type, <MAILTO: ietf-calendar@imc.org> using the format defined in section 7 of [iCAL].

Authors' Addresses

Steve Mansour
AOL/Netscape
466 Ellis Road
Mountain View, CA 94043
US

Phone: +1-650-937-3351
EMail: sman@netscape.com

Doug Royer
INET-Consulting LLC
1795 W. Broadway #266
Idaho Falls, ID 83402

Phone: 208-520-4044
EMail: Doug@royer.com

George Babics
Steltor
2000 Peel Street
Montreal, Quebec H3A 2W5
CA

Phone: +1-514-733-8500 x4201
Fax: +1-514-733-8878
EMail: georgeb@steltor.com

Paul Hill
Massachusetts Institute of Technology
W92-172
77 Massachusetts Avenue
Cambridge, MA 02139
US

Phone: +1-617-253-0124
Fax: +1-617-258-8736
EMail: phb@mit.edu

Appendix A. Acknowledgments

The following have individuals were major contributors in the drafting and discussion of this memo:

Harald Alvestrand, Mario Bonin, Andre Courtemanche, Dave Crocker, Bernard Desruisseaux, Pat Egen, Gilles Fortin, Jeff Hodges, Alex Hoppman, Bruce Kahn, Lisa Lippert, David Madeo, Bob Mahoney, Bob Morgan, Patrice Lapierre, Pete O'Leary, Richard Shusterman, Tony Small, John Stracke, Alexander Taler, Mark Wahl.

Appendix B. Bibliography

[BEEP] Rose, "The Block Extensible Exchange Protocol Core", RFC 3080, March 2001

[BEEPTCP] Rose, "Mapping the BEEP Core onto TCP", RFC 3081, March 2001

[MIME] N. Borenstein and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Internet Draft UTF-825 July 1996

[RFC 3087] Campbell, Sparks, "Control of Service Context using SIP Request-URI", RFC 3087, April 2001

[RFC 2392] Levinson, "Content-ID and Message-ID Uniform Resource Locators", RFC 2392, August 1998

Specifying and Describing the Format of Internet Message Bodies", RFC 1521, Bellcore, Innosoft, September 1993.

[TLS] Dierks, Allen, "The TLS Protocol", RFC 2246, January 1999

[RFC2119] TODO...

[SASL] RFC2222 TODO...

[URL] Berners-Lee, Fielding, Masinter, "Uniform Resource Identifiers

(URI): Generic Syntax", RFC 2396, August 1998.

[iCAL] Dawson, Stenerson, "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 2445, November 1998

[iTIP] Silverberg, Mansour, Dawson, Hopson, "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 2446, November 1998

[iMIP] Dawson, Mansour, Silverberg, "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 2445, November 1998

[SQL] "Database Language SQL", ANSI/ISO/IEC 9075: 1992, aka ANSI X3.135-1992, aka FiPS PUB 127-2

[SQLCOM] ANSI/ISO/IEC 9075:1992/TC-1-1995, Technical corrigendum 1 to ISO/IEC 9075: 1992, also adopted as Amendment 1 to ANSI X3.135.1992

[UNICODE] The Unicode Consortium, "The Unicode Standard -

Worldwide Character Encoding -- Version 1.0", Addison-Wesley, Volume 1, 1991, Volume 2, 1992. UTF-8 is described in Unicode Technical Report #4.

[US-ASCII] Coded Character Set--7-bit American Standard Code for Information Interchange, ANSI X3.4-1986.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.