Network Working Group
Internet-Draft
Expires: January 18, 2002

S. Mansour
Netscape, iPlanet
D. Royer

G. Babics
Steltor
P. Hill
Massachusetts Institute of
Technology

July 20, 2001

Calendar Access Protocol (CAP)
draft-ietf-calsch-cap-05

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-calendarand 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.

This protocol is based on request/response form of data units, sent from a client CUA to a calendar server. The protocol data units leverage the standard iCalendar format [iCAL] for conveying CS 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. Calendar access methods defined by this memo, as well as scheduling methods defined by [iTIP], are referred to with capitalized, quoted-strings of text. For example, "CREATE" refers to the method for creating a calendar component on a calendar, "READ" refers to the method 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 lower case, quoted-strings of text, followed by the word "parameter". For example, "value" parameter refers to the iCalendar property parameter used to override the default data type for a property value. Enumerated values defined by this 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].

[IMPL] (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 or definitions from these other memos. Where possible, references are made to the memo that provides for the specification of these concepts or definitions.

1.3 Definitions

Authentication ID (AuthID)

A tuple of username, realm, and authentication method, used by the Calendar Service internally to identify a successfully authenticated CAP session.

Booked

A 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 or calendar user agent. A booked appointment is an entry that has had its state changed from one of the [iTIP] scheduling methods to a CAP method of CREATE by a calendar user or calendar user agent which resulted in decision to keep the entry in the calendar store.

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 a calendar residing on a Calendar Store.

Calendar Access Rights (CAR)

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

Calendar Collection

A collection of Calendars, Resource Calendars, and/or other Calendar Collections. These collections are expanded by the CS and may reside either locally or in an external database or directory. The calendars in the collection may be fixed or dynamic over time.

Calendar Component

An item within a calendar. Some types of calendar components include 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 Properties

An attribute of a calendar. 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 components can be accessed using CAP.

Calendar Store Properties

Properties maintained in a Calendar Store calendar store-wide information. Calendar store properties can be accessed using CAP.

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 or UG utilizes to access and manipulate a calendar.

Calendaring and Scheduling System

The computer sub-system that provides the services for accessing, manipulating calendars and scheduling calendar components.

CAP Session

An open communication channel between a CAP CUA and a CAP CS.

Connected Mode

A mobile computing mode where the CUA is directly connected to the CS.

Delegate

Is 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

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

Disconnected Mode

A mobile computing mode where a CUA can be disconnected from a CS. When the CUA is disconnected, it is in the disconnected mode.

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. This may include the calendar associated with TARGET calendar property.

Hierarchical Calendars

A CS feature where a calendar have a hierarchical relationship with another calendar in the CS. The top-most calendar in the

hierarchical relationship has the CS as its 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" topographical relationship. In addition, sub-calendars may have a relationship with another calendar that has a "child" topographical relationship. In addition, a calendar may have a relationship such that one or more calendars have a "sibling" topographical relationship with the calendar. 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.

High Bandwidth Connection

A communications connection supporting high transfer rates; transfer rates commonly found within a LAN.

Local Store

A CS which is on the same platform as the CUA.

Low Bandwidth Connection

A communications connection supporting slow transfer rates; transfer rates commonly found in remote access technology.

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 and in the "OWNER" calendar store 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. (eg: 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].

Remote Store

A CS which is not on the same platform as the CUA.

Resource Calendar (RC)

A non-human Calendar, associated with an organizational resource. There is no syntactic difference between an RC and a Calendar.

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.

User Name

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

User Principal Name (UPN)

An identifier that denotes a unique CU. A UPN is an 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 to email address mapping MUST never be deliverable to a different person as there is not requirement that they are the same. 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.

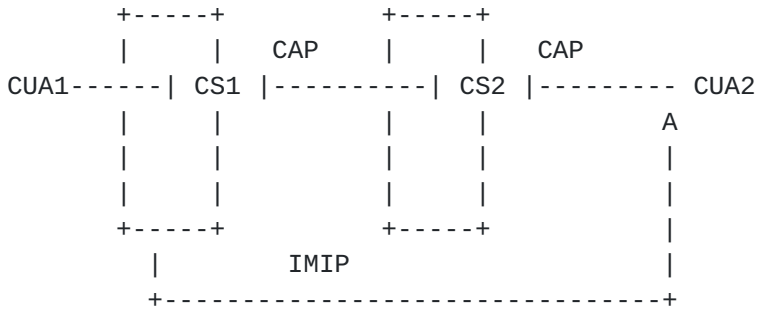
CAP is used by a "Calendar User Agent" (CUA) to send commands to and receive responses from a "Calendar Service" (CS). The CUA prepares a [MIME] encapsulated iCalendar object containing a command, sends it to the CS, and receives a [MIME] encapsulated iCalendar object as a response. There are two distinct protocols in operation to accomplish this exchange. The Transfer Protocol is used to move these encapsulations between a CUA and a CS. The Application Protocol defines the content and semantics of the iCalendar objects sent between the CUA and the CS. This document defines both the Transfer Protocol and the Application Protocol.

In the diagram below, a user uses CUA1 to communicate with CS1 using CAP. The CUA must authenticate the Calendar User (CU) so that access to calendars on CS1 can be controlled. The CUA can then view, create, edit, and delete calendars, calendar properties, and calendar components subject to the access rights.

CAP servers support fanout. Fanout allows a CUA to communicate with a single CS to perform scheduling operations with calendars on multiple CSs. That is, a CU can book or schedule events (or to-dos) on or read events (or to-dos) from calendars on other calendar stores. To accomplish this, a CAP server takes on these roles:

* CS1 MAY play the role of a CUA and use CAP to access CS2;

* CS1 MUST be able play the role of a CUA and use [iMIP] to interoperate with other CUAs.

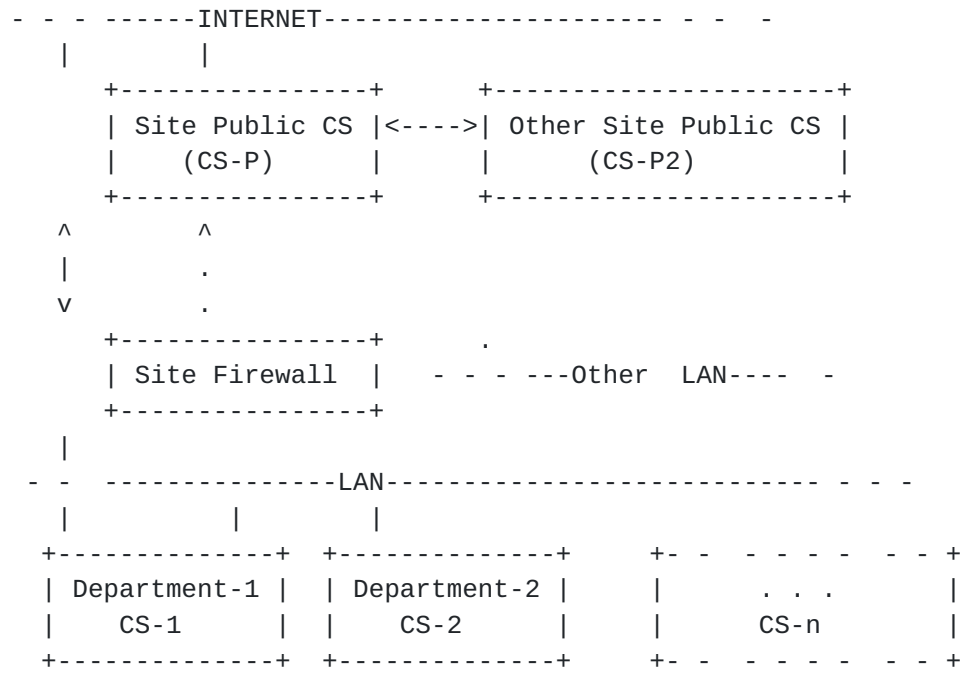


Note that the fanout feature in CAP is a convenience to the CUA. It

is perfectly valid for the CUA to assume the responsibility for fanout if it wishes. That is, [iMIP] messages could also be sent from CUA1 to CUA2.

CAP does not specify any trust model between CS1 and CS2. Implementations may require a trust model to exist in order to accomplish scheduling between CS1 and CS2.

2.1.1 Firewalls and Fanout



One example of a site site policy could be that all entries in department calendars that were marked PUBLIC, were visible on the internet to any users that was granted access to CS-P.

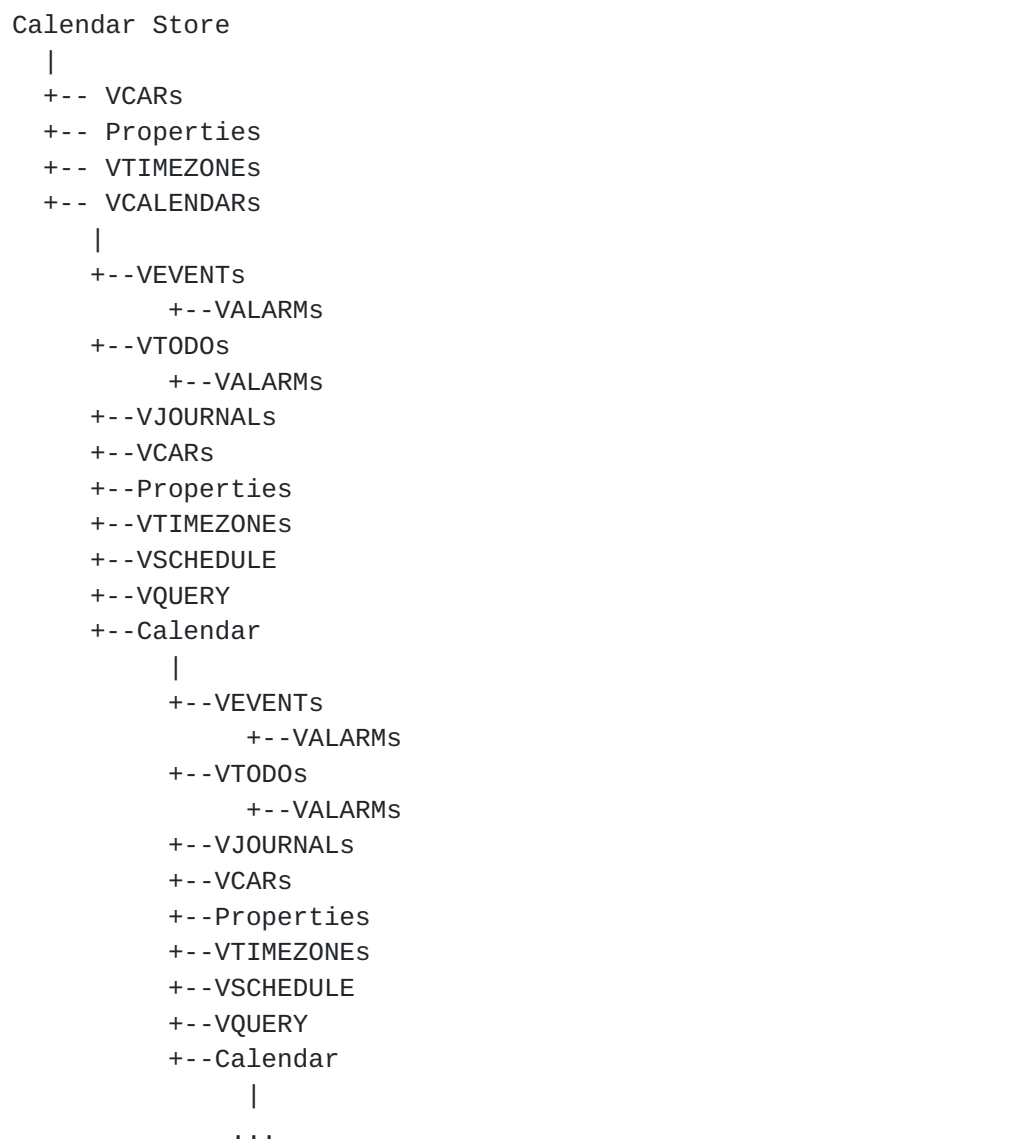
With fanout in place, CS-P would authenticate the internet user and decide in an implementation specific way, which user CS (CS-1, CS-2, CS-n) the TARGET calendar physically existed. Then using fanout, CS- P sends the request subject to VCARs or immutable VCARS from CS-P to the users CS and back to the CUA through CS-P. CS-P could be a smart proxy.

And there is no reason that the target calendar need to exist under the firewall (CS-P2), only that the CSs can authenticate with each other. In this way all users in a domain could appear to be at calendar.site.com, hiding all CS-1 through CS-n from the internet. And also hiding that fact that CS-P2 may be a separate domain controlled by the same site administrators as CS-P (which in turn could do fanout to its LAN).

2.2 Calendar Store Object Model

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

Calendars contain VEVENTs, VTODOs, VJOURNALs, VALARMs, VCARs, and calendar properties. Calendars may also contain other calendars.



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

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

2.3 Protocol Model

A generic transfer-layer, Calendar Server Transfer Protocol (CSTP), is used to move data objects between a CUA and the CS. CSTP commands are listed below and their usage and semantics are defined in section 7.1 of this document.

CSTP Commands

Command	Description
ABORT	Stop a command. Perhaps because its latency time has been exceeded.
AUTHENTICATE	Authenticate a UPN.
CONTINUE	Continue the execution of a command whose Latency time has been exceeded.
IDENTIFY	Set a new identity for calendar access.
DISCONNECT	Terminate a connection with the server.
SENDDATA	Send a data object [MIME] encapsulated iCalendar.
STARTTLS	Negotiate transport level security using [TLS].
NOOP	Do nothing operation.

Application-level commands are used to manipulate data on the calendar store. They are listed below and discussed in detail in section 7.2.

CAP Calendaring Commands

Command	Description
CREATE	Create a new calendar or component.
DELETE	Delete a calendar or component.
GENERATEUID	Generate one or more unique ids.
MODIFY	Change a calendar or component.
MOVE	Move a calendar.
READ	Read a calendar properties or components.

CAP Scheduling Commands
(As defined 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.

2.4 Security Model

Authentication to the CS will be performed using [SASL].

As noted in the CAP system model section, a variety of connectivity scenarios are possible. This complicates the security model considerably, and a thorough familiarity with the concepts is required to ensure interoperability.

Basic threats to a Calendaring and Scheduling System include:

(1) Unauthorized access to data via data-fetching operations,

(2) Unauthorized access to reusable client authentication information by monitoring others' access,

(3) Unauthorized access to data by monitoring others' access,

(4) Unauthorized modification of data,

(5) Unauthorized or excessive use of resources (denial of service), and

(6) Spoofing of CS: Tricking a client into believing that information came from the CS when in fact it did not, either by modifying data in transit or misdirecting the client's connection.

Threats (1), (4), (5) and (6) are due to hostile clients. Threats (2), and (3) are due to hostile agents on the path between client and server, or posing as a server.

CAP can be protected with the following security mechanisms:

(1) Client authentication by means of the SASL mechanism set, possibly backed by the TLS credentials exchange mechanism,

(2) Client authorization by means of access control based on the requestor's authenticated identity,

(3) Data integrity protection by means of the TLS protocol or data integrity SASL mechanisms,

(4) Protection against snooping by means of the TLS protocol or data encrypting SASL mechanisms,

(5) Resource limitation by means of administrative limits on service controls, and

(6) Server authentication by means of the TLS protocol or SASL mechanism.

Imposition of access controls (authorizations) is done by means of VCARs, an overview is provided in the section 2.4.4 <fwd ref> and a detailed syntax is provided in section 6 <fwd ref> Authentication is explained in detail in section 2.4.1 <fwd ref>

2.4.1 Authentication, Credentials, and Identity

Generically, authentication credentials are the evidence supplied by one party to another, asserting the identity of the supplying party (e.g. a user) who is attempting to establish an association with the other party (typically a server). Authentication is the process of generating, transmitting, and verifying these credentials and thus the identity they assert. An authentication identity is the name presented in a credential.

There are many forms of authentication credentials. The form used depends upon the particular authentication mechanism negotiated by the parties. For example: X.509 certificates, Kerberos tickets, simple identity and password pairs. Note that an authentication mechanism may constrain the form of authentication identities used with it.

SASL only provides a protocol to negotiate a mutually acceptable authentication mechanism. SASL itself does not constrain or dictate the form of the authentication identities used to perform the authentication.

2.4.2 Calendar User and UPNs

A Calendar User(CU) is an entity that can be authenticated. It is represented in CAP as a UPN. A UPN is the owner of a calendar and 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 control policy, requiring a server-specific mapping to be done. The method by which a server composes and validates an authorization identity from the authentication credentials supplied by a client is implementation-specific.

For Calendaring and Scheduling Systems that are integrated with a directory system, the CS MUST support the ability to configure which schema attribute stores the UPN. The CS MAY allow one or more attributes to be searched for the UPN.

2.4.2.1 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: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 couldn't be munged to something misleading like "ATTENDEE;CN=Joe Rictus User:juser@example.com" and have it pass validation. This validation will also defeat other attempts at confusion.

2.4.2.2 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.

CAP implementations MUST support anonymous authentication, as defined in section <fwd ref 7.1.3>

CAP implementations MAY support anonymous authentication with TLS, as defined in section <fwd ref 7.1.3.2>

2.4.2.3 Required Security Mechanisms

The following implementation conformance requirements are in place:

(1) For a read-only, public CS, anonymous authentication, described in section <fwd ref 7.1.3.1> can be used.

(2) Implementations providing password-based authenticated access MUST support authentication using Digest, as described in section <fwd ref&gt [ed note: this section has not yet been defined. Paul can you please look into it?]; This provides client authentication with protection against passive eavesdropping attacks, but does not provide protection against active intermediary attacks.

(3) For a CS needing session protection and authentication, the Start TLS extended operation, and either the simple authentication choice or the SASL EXTERNAL mechanism, are to be used together. Implementations SHOULD support authentication with a password as described in section <fwd ref> and SHOULD support authentication with a certificate as described in section <fwd ref> Together, these can provide integrity and disclosure protection of transmitted data, and authentication of client and server, including protection against active intermediary attacks.

2.4.2.4 TLS Ciphersuites

The following ciphersuites defined in [RFC 2246] MUST NOT be used for confidentiality protection of passwords or data:

TLS_NULL_WITH_NULL_NULL

TLS_RSA_WITH_NULL_MD5

TLS_RSA_WITH_NULL_SHA

The following ciphersuites defined in [RFC 2246] can be cracked easily (less than a week of CPU time on a standard CPU in 1997). The client and server SHOULD carefully consider the value of the password or data being protected before using these ciphersuites:

TLS_RSA_EXPORT_WITH_RC4_40_MD5

TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5

TLS_RSA_EXPORT_WITH_DES40_CBC_SHA

TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA

TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA

TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA

TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA

TLS_DH_anon_EXPORT_WITH_RC4_40_MD5

TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA

The following ciphersuites are vulnerable to man-in-the-middle attacks, and SHOULD NOT be used to protect passwords or sensitive data, unless the network configuration is such that the danger of a man-in-the-middle attack is tolerable:

TLS_DH_anon_EXPORT_WITH_RC4_40_MD5

TLS_DH_anon_WITH_RC4_128_MD5

TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA

TLS_DH_anon_WITH_DES_CBC_SHA

TLS_DH_anon_WITH_3DES_EDE_CBC_SHA

A client or server that supports TLS MUST support at least:

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

2.4.3 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 MUST accept a CUA request for UG expansion, although the CS may be configured to restrict some responses. The CS MAY expand a UG (including nested UGs) to obtain a list of unique CUs. Duplicate UPNs are filtered during expansion. Incomplete expansion should be treated as a failure.

Groups may be in a directory service with its own ACL model and CAP should use the directory service to expand a UPN subject to the

directory service access control model for the authenticated entity.

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. However, during fan out to other CSs, the originating CS SHOULD expand all UGs so that the target CS receives only a list of unique CUs. This is to prevent confusion when two CSs do not share the same UG database or directory.

CAP does not define commands or methods for managing UGs.

Small UG:

The Three Stooges. There is only and always three at any one time.

Large UG:

The MIT graduating class of 1999. This is a static list.

Dynamic UG:

The IETF membership which is a large and changing list of members.

Nested UG:

The National Football League, made up of the AFC and NFC, which are made up of 3 divisions each...

2.4.4 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 Vcalendar 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.

2.4.4.1 VCalendar 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>

2.4.4.2 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 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 out side 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.

2.4.5 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.

2.4.6 CAP Session Identity

A CAP 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.

Examples:

Small UG:

The Three Stooges. There will always be three, it will not change.

Large UG:

The MIT graduating class of 1999. This is a static list.

Dynamic UG:

The IETF membership, which is a large and changing list of members.

Nested UG:

The National Football League, made up of the AFC and NFC, which are made up of 3 divisions each.

2.5 Roles

CAP defines methods for managing [iCAL] objects on 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 or UGs as attendees takes on the role of "Organizer". The CUs or UGs asked to participate in the group 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. Its default value is 5229. The port must be present in the URL if the CAP server does not listen on this

default port of 5229.

<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 CAP command set, query feature, and access rights onto the iCalendar format, several extended iCalendar methods and components are defined by this memo.

The search function is specified with the new iCalendar QUERY method. The QUERY method makes use of a new component, called VQUERY, that contains the search filter. The component consists of a set of new properties: EXPAND, MAXSIZE, MAXRESULTS, QUERY and QUERYNAME, that define the search filter. Note that QUERY simply is the filter that is used by the CS to select the data used in METHOD:READ, METHOD:CREATE, METHOD:MODIFY, METHOD:DELETE, any other METHOD that is defined to use a QUERY filter.

Access control is specified the the new iCalendar VCAR component.

The iCalendar METHOD property format has been updated with new values.

A new iCalendar component, VCOMMAND, has been added. VCOMMANDs are needed to specify CAP commands.

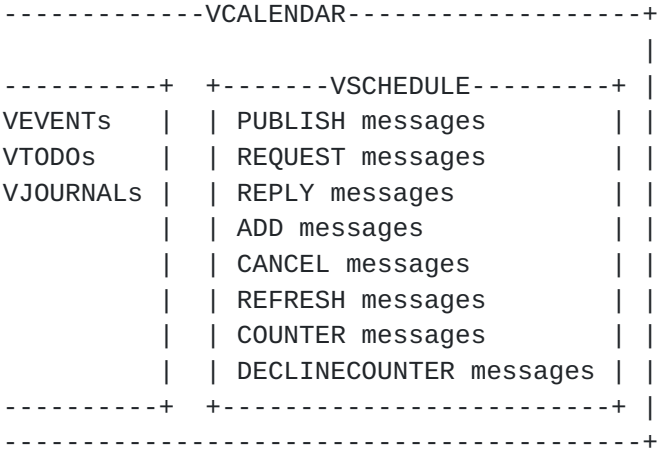
VCAP is a new container for data that is transmitted as part of a VCOMMAND

TARGET is a new property within the VCOMMAND component. It indicates the calendars to which the command applies

CMDID is a Command ID that is used in a VCOMMAND to uniquely identify a command. Responses to a VCOMMAND will contain the same CMDID.

2.8 Relationship of RFC 2446 (ITIP) to CAP

[iTIP] describes scheduling methods which result in indirect manipulation of calendar components. CAP methods 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 queue. Note that this is a conceptual model, the actual storage details are left to implementations. The model is shown pictorially as follows:



The METHOD is saved along with components. Scheduled components become booked components when the METHOD changes from an ITIP method to the CAP storage method CREATE. For example, a component whose METHOD is "REQUEST" is scheduled. The component becomes booked when the METHOD is changed to "CREATED".

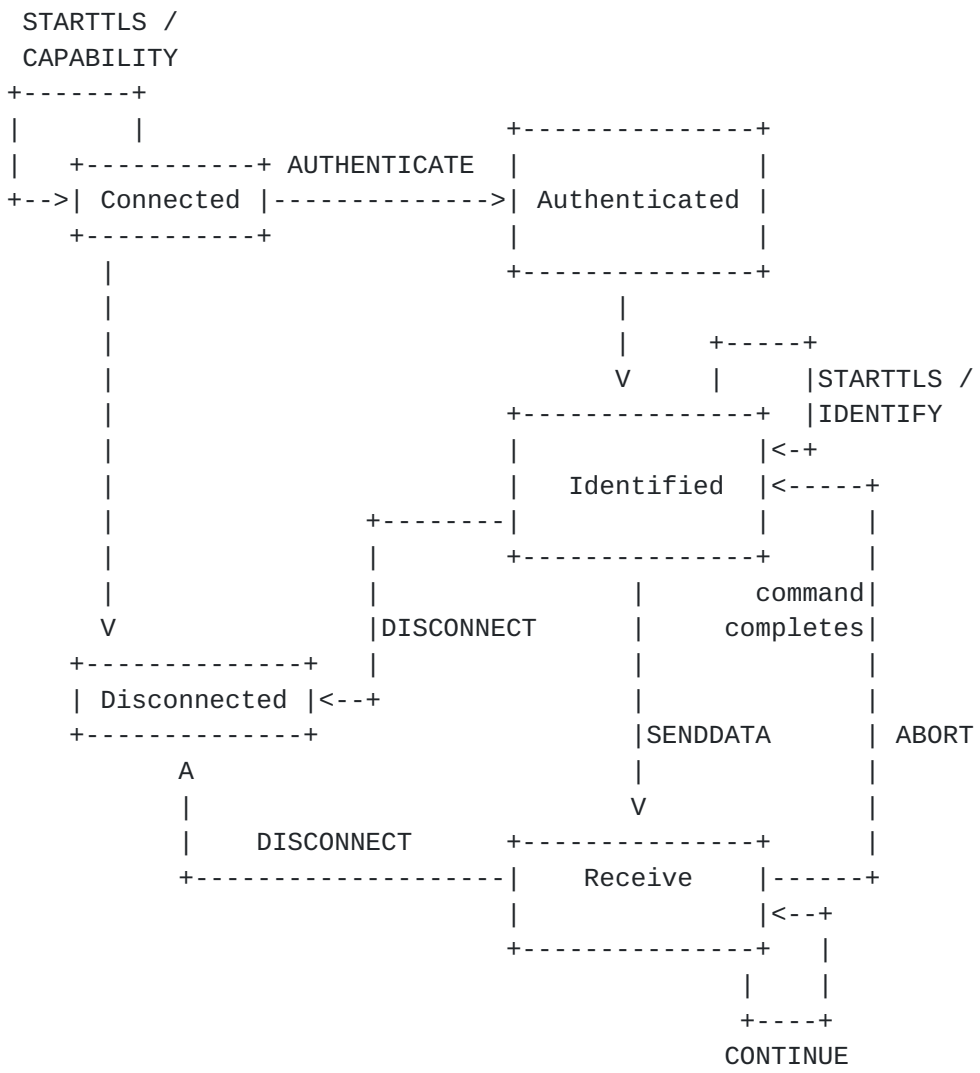
Several scheduled entries can be in the CS for the same UID. They are consolidated when booked. Or they are removed from the CS.

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 READ 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. Followed by a DELETE of all of these now old scheduling requests in the CS. See [iTIP] for details on resolving multiple [iTIP] scheduling entries.

3. State Diagram

This section describes the states of the transport connection between a CUA and a CS. The state diagram is shown below. State names are shown with first letter capitalized. Commands used to switch between states are shown next to an arrow connecting the states. The commands are listed in all capital letters. A condition that causes a state to change is shown in lower case letters.



The connection begins the Connected state when a CUA connects to a CAP server. The capabilities of the CS are reported in the response from the CS. From this state, the CUA can issue the DISCONNECT command to terminate the connection, the CAPABILITY, STARTTLS, or AUTHENTICATE commands. One use of the CAPABILITY command at this stage is to determine the supported authentication mechanisms supported by the server. Once the STARTTLS command has been successfully executed from either the Connected or Authenticated

state, it must not be executed again.

If an AUTHENTICATE command is successful, the connection enters the Authenticated state and then immediately goes to the IDENTIFIED state. While in the Identified state, the CALIDEXPAND and UPNEXPAND commands can be executed. From here the CUA can issue the CAPABILITY command. The capabilities offered by the server in the Authenticated state may be different than those in the Connected state to allow for the user's realm to be used to grant or deny features. The CUA can also use the IDENTIFY command to change the identity of the user subject to access control. The connection remains in the Identified state after the CAPABILITY command completes. The CUA can issue the DISCONNECT command to terminate the connection. The SENDDATA command can be used to send a request to read, write, modify, or delete data on the server.

After the SENDDATA command has been issued the connection enters the Receive state while the CUA awaits and reads a server reply. Normally, the server handles the command, sends a reply which is read by the CUA and the connection returns to the Authenticated state. The CUA may have issued the SENDATA command with a maximum latency time. This informs the server that the CUA expects a response within the maximum latency time, even if the command was not completed. When the server is unable to complete the command in the maximum latency time, it issues an appropriate reply code and waits for the CUA to tell it how to proceed. If the CUA issues a CONTINUE command the server continues processing the command and the connection remains in the Receive state. If the CUA issues the ABORT command the server need not process the command any further and the connection returns to the Identified state. The DISCONNECT command can also be issued from the Receive state.

4. Protocol Framework

4.1 CAP Application Layer

The CAP application layer is used for the manipulation of the calendar store. Commands and responses are transmitted between the CUA and CS inside "VCAP" wrappers. Commands are specified as the value of a "METHOD" property, and responses are specified as the value of a "REQUEST-STATUS" property. An optional "CMDID" may be used to uniquely identify commands.

4.2 CAP Transfer Protocol

The CAP transfer protocol defines the format of data transmitted between a CUA and the CS.

CAP transfer protocol commands are transmitted using the underlying transport. The transport used is a TCP/IP socket connection between the CUA and the CS. The default port that the CS listens for connections on is port 5229.

Messages sent between the CUA and CS are formatted as a command followed by any associated data:

<command> [<command data>]

Server responses consist of a response code and any parameters:

```
<transfer-level response-code> [response-args] [; debug-text ]
<CRLF>.<CRLF>
[<application-data>]
<CRLF>.<CRLF>
```

The response-args are defined in Section 8. The debug-text is human-readable information for protocol debugging and is optional and is only used to aid developers writing CSs and CUAs. The debug-text MUST not be depended on by CUAs in normal interactions with a CS.

The optional application-data begins on the next line.

The response is terminated with the second <CRLF> "." <CRLF> sequence. Any <CRLF> "." sequences which appear in the transmitted data must be quoted by placing an additional "." between the <CRLF> and the ".". For example, the following sequences of characters in the application data:

```
.
..2
```

...3

are quoted as follows:

..
...2
....3

4.3 Pipelining Commands

If the CS returns a PIPELINE number greater then one (1) as a result of a CAPABILITY command then the CUA can send up to PIPELINE VCAP (each with a unique CMDID/VCOMMAND) without waiting for the CS to reply.

It is unspecified what happens of the CUA exceedes the CS limit.

4.4 Auto-logout Timer

If a server has an inactivity auto-logout timer, that timer MUST be of at least 15 minutes duration if there is no value of AUTOLOGOUTTIME returned in the CS CAPABILITY reply. The receipt of ANY command from the client during that interval MAY reset the auto-logout timer, subject to CS administrators policy. A CUA may be discouraged from attempts to do useless things only to keep the connection alive.

A CS MAY have different AUTOLOGOUTTIME values depending on the UPN or the realm of the UPN.

When a timeout occurs, the server drops the connection to the CUA.

4.5 Bounded Latency

CAP is designed so that the CUA can either obtain an immediate response from a request or discover within a specified amount of time that the request could not be completed in the requested amount of time. When the CUA initiates a command that the server cannot complete within the specified latency time, the server returns an appropriate response code. The CUA then issues either a CONTINUE or ABORT command. The ABORT command immediately terminates the command in progress identified by CMDID and the connection returns to the Authenticated state. The CONTINUE command instructs the server to continue processing the command identified by the CMDID.

4.6 Data Elements

The data elements for CAP are [MIME] encapsulated iCalendar objects.

5. Formal Command Syntax

5.1 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].

5.1.1 Grammar For Search Mechanism

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

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

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

maxresults = "MAXRESULTS" ":" integer

maxsize    = "MAXSIZE" ":" integer

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 capfrom-min capwhere-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 cmpmin-oper colvalue
                   / colname cmpmin-oper colvalue
                     capmin-logical capmin-cmp )

    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.
```

5.1.2 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 GMT ascending.

(6) SQL-MIN MUST be implemented.

5.1.3 SQL-92 notes

(1) Although this memo specifies that any [SQL] query can be used, some of the [SQL] grammar is optional and therefore is considered optional in CSs advertising an SQL-92 implementation with CAPABILITY.

(2) A CS implementation MAY implement SQL-92. If a CS does not implement SQL-92 then it MUST advertise SQL-MIN in the capability reply.

5.1.4 Example, Query by UID

This example would select the entire contents of uid123 and not expand any multiple instances of the component. If the CUA does not know if 'uid123' was a VEVENT, VTODO, VJOURNAL, or 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
```

This example would select the entire contents of 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
```

5.1.5 Query by Date-Time range

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

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

5.1.6 Query for all Non-Booked Entries

This example selects the entire contents of all scheduling (non-booked) 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 != 'CREATE'
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 != 'CREATE'
END:VQUERY
```

5.1.7 Query with Subset of Properties by Date/Time

This MUST implement is similar to the one above, except only the named properties will be selected and all booked and non- booked components will be selected that have a DTSTART from Feb 1st to Feb 10th.

```
BEGIN:VQUERY
QUERY:SELECT UID,DTSTART,DESCRIPTION,SUMMARY FROM VEVENT
  WHERE DTSTART >= '20000201T000000Z'
  AND DTSTART <= '20000210T235959Z'
```

```
END:VQUERY
```

5.1.8 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
```

6. Access Rights

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

Individual calendar access rights MUST be specifically granted to an authenticated calendar user (i.e., UPN); all rights are denied unless specifically granted.

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

6.1 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. By implication, 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.

6.2 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 While access control may allow a UPN to store an event on a particular calendar. , the CONFLICTS Calendar or component setting may prevent it, returning an error code "6.3"

7. Commands and Responses

CAP commands and responses are described in this section.

Command arguments, identified by "Arguments:" in the command descriptions below, are described by function, not by syntax. The precise syntax of command arguments is described in the Formal Syntax section.

Some commands cause specific server data to be returned; these are identified by "Data:" in the command descriptions below. See the response descriptions in the Responses section for information on these responses, and the Formal Syntax section for the precise syntax of these responses.

The "Result:" in the command description refers to the possible status responses to a command, and any special interpretation of these status responses.

Commands have the general form:

```
<command> [arguments...]
```

where <command> is a command listed in the table above. A command MAY have arguments. Arguments are defined in the detailed command definitions below.

Responses to commands have the following general form:

```
responseCode [sep transportDescr sep
[applicationDescr]]
CRLF "." CRLF
```

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

7.1 Transfer Protocol Commands

7.1.1 Initial Connection

Arguments: none

Data: none

Result: 2.0 - success. 8.1 - server too busy

Upon session startup, the server sends a response of 2.0 to indicate that it is ready to receive commands. A response of 8.1 indicates that the server is too busy to accept the connection. In addition, the general capabilities of the CS are reported in the response from the CS. These capabilities may be different than those reported in the authenticated state.

7.1.2 ABORT Command

Arguments: [CMDID]

Data: none

Result: 2.0 - success 2.2 - no command is in progress

The ABORT command is issued by the CUA to stop a command. A common usage could be to ABORT a command whose latency time has been exceeded. When the latency time is specified on the SENDATA command, the CS must issue a reply to the CUA within the specified time. It may be a reply code indicating that the CS has not yet processed the request. The CUA must then tell the server whether to continue or abort.

The CUA can issue the ABORT command at any time after the SENDATA command has been completed but before receiving a reply.

If CMDID is supplied then the command identified by CMDID is aborted.

If CMDID is not supplied then the fist command that is still pending is aborted.

7.1.3 AUTHENTICATE Command

Arguments: <SASL mechanism name> [<initial data>]

Data: continuation data may be requested

Result:

2.0 - Authenticate completed, now in authenticated state.

6.0 - Failed authentication. 6.1 - Authorization identity refused.

6.2 - Sender aborted authentication, authentication exchange cancelled.

6.3 - Unsupported Authentication Mechanism.

6.4 - Temporary failure (back end authentication server down).

6.5 - Authentication exchange cancelled.

6.6 - Authentication mechanism too weak.

6.7 - Encryption required.

6.8 - Pass phrase expired. The pass phrase was correct but expired. The user will have to contact a pass phrase change server prior to authenticating.

6.9 - The user is valid but the server does not have an entry in the authentication database for the requested mechanism (e.g., DIGEST- MD5). If the user successfully authenticates using a plain text password, then the back end back end entry will be updated. Note that if the client chooses to fall back to plain text password authentication it should enable an encryption layer or get user-con- firmation that a one-time transition is acceptable.

6.10 - User account disabled. The user will have to contact the system administrator to get the account re-enabled.

9.1 - Unexpected command.

The capabilities of the CS in the authenticated state are reported in the response from the CS. These may be different than the capabilities in the Connected, but unauthenticated state.

The AUTHENTICATE command is used by the CUA to identify the user to the CS. CAP supports a number of authentication methods, the SASL specification for authentication is the preferred method.

If STARTTLS is negotiated prior to the AUTHENTICATE command, the client MUST discard all information about the CS capabilities fetched prior to the TLS negotiation. In particular, the value of supported SASL Mechanisms MAY be different after TLS has been negotiated.

If a SASL security layer is negotiated, the client MUST discard all information about the CS capabilities fetched prior to SASL. In particular, if the client is configured to support multiple SASL mechanisms, it SHOULD fetch supported SASL Mechanisms both before and after the SASL security layer is negotiated and verify that the value has not changed after the SASL security layer was negotiated. This detects active attacks which remove supported SASL mechanisms from the supported SASL Mechanisms list.

<initial data> is an optional parameter which can be used for mechanisms which require an initial response from the CUA.

The AUTHENTICATE command is followed by an authentication protocol exchange, in the form of a series of CS challenges and CUA responses, per the relevant RFC that describes the specific SASL mechanism being used.

Cancelling the authentication process during its negotiation is implementation specific and not within scope of the CAP specification.

If a security layer was negotiated it comes into effect for the CS starting with the first octet transmitted after the CRLF which follows the 2.0 reply code, and for the CUA starting with the first octet after the CRLF of its last response in the authentication exchange. Encrypted data is transmitted as described in SASL.

The service name specified by this protocol's profile of SASL is "cap". [EDITORS NOTE: this must be registered with IANA, has anyone done this yet?]

The result of the AUTHENTICATE command includes data indicating the identity which has been assigned to the session, derived from the supplied authentication credentials, and/or authorization identifier. A CAP session does not have an identity until the CUA has issued the "AUTHENTICATE" command.

The CUA may not issue the "AUTHENTICATE" command multiple times, even if the first attempt was aborted. If a CUA attempts to do this the CS must terminate the session.

Here is an example of a successful authentication:

```
C: AUTHENTICATE KERBEROS_V4
S: AmFYig==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
S: or//EoAADZI=
C: DiAF5A4gA+oOIALuBkAAmw==
S: 2.0
S: .
S: Content-Type:text/calendar; method=REQUEST;
S:  charset=US-ASCII
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: PRODID:-//ACME/CAPserver//EN
S: VERSION:2.1
```

```
S: IDENTITY=bill@example.com
S: CAPVERSION=1.0
S: ITIPVERSION=1.0
S: AUTH=KERBEROS_V4
S: AUTH=DIGEST_MD5
S: CAR=CAR-FULL-1
S: MINDATE=19700101T000000Z
S: MAXDATE=20370201T000000Z
S: END:VCAP
S: .
```

This example shows a failed authentication:

```
C: AUTHENTICATE KERBEROS_V4

S: AmFYig==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
S: .
S: 6.0
S: .
S: .
```

7.1.3.1 AUTHENTICATE ANONYMOUS

RFC-2245 defines the Anonymous SASL mechanism. This RFC states that "the mechanism consists of a single message from the client to the server. The client sends optional trace information in the form of a human readable string. The trace information should take one of three forms: an Internet email address, an opaque string which does not contain the '@' character and can be interpreted by the system administrator of the client's domain, or nothing. For privacy reasons, an Internet email address should only be used with permission from the user."

RFC-2245 goes on to state, "A client which uses the user's correct email address as trace information without explicit permission may violate that user's privacy." Information about who accesses an anonymous CS on a sensitive subject (e.g., AA meeting times and locations) has strong privacy needs. "Clients should not send the email address without explicit permission of the user and should offer the option of supplying no trace token -- thus only exposing the source IP address and time."

Example of CUA using the Capability command followed by an anonymous authentication:

```
C: CAPABILITY
```

```
S: 2.0
S:CAPVERSION=1.0
S:AUTH=KERBEROS_V4
S:AUTH=DIGEST_MD5
S:AUTH=ANONYMOUS
S:.
C:AUTHENTICATE ANONYMOUS
S:+
C:c21yaGM=
S:2.0
```

Subsequent to the initial Anonymous Authentication with a CS, a CUA will have to provide a UPN for some CAP operations. For anonymous access the UPN that MUST be used by the CUA is "@", a UPN with a null username and null realm. The user's normal UPN MUST not be used for the subsequent CAP operations.

Note that the CS implementation may have internal audit logs that use the user's asserted UPN as trace information. However, this UPN will not appear on the wire after the initial SASL anonymous authentication.

Use of the "@" UPN and wild-carding of UPNs within VCARs are covered in

Section <forward ref>.

7.1.4 CALIDEXPAND Command

Arguments: CalID

Data: no specific data for this command

Result: 2.0 Successful, and requested data follows

2.1 Permission Denied

2.2 Requested CSID not found

2.3 Result exceeds MAXEXPANDLIST

2.4 Misc. EXPAND error

Return the members of the argument CalID. Successful result yields one or more Calendars and/or Resource Calendars. More than one C or RC returned implies that the CalID was a CC.

If the number of items exceeds MAXCALIDEXPAND, then up to

MAXCALIDEXPAND items are returned along with result 2.3.

If the result is 2.4, then as many of the items as possible will be returned. It is possible that no items will be returned. The 2.4 response may have optional text describing the problem. Any such text MUST be in the language and charset that is defined by the calendar store properties LANGUAGE and CHARSET. If calendar store CHARSET is not set, but the LANGUAGE property is set, then the error message will be in LANGUAGE in the UTF-8 charset. If calendar store LANGUAGE property is not set, then the CS MUST NOT return any text with the results.

Examples:

Example #1: Request lookup of CSID which is a Calendar Collection

```
C: CALIDEXPAND cap://cal.example.com/calid14
S: 2.0
S: cap://cal.example.com/calid14
S: cap://cal.example.com/calid2
S: cap://cal.example.com/calid5
S: cap://cal.example.com/calid66
S: .
```

Example #2: Request lookup a CSID which is a Resource Calendar

```
C: CALIDEXPAND cap://cal.example.com/conf5
S: 2.0
S: cap://cal.example.com/conf5
S: cap://cal.example.com/conf5
S: .
```

Example #3: Request CSID which exceeds MAXCALIDEXPANDLIST

```
C: CALIDEXPAND cap://cal.example.com/calid76
S: 2.3 Expansion resulted in too much data
S: cap://cal.example.com/calid76
S: cap://cal.example.com/calid3
S: cap://cal.example.com/calid12
S: cap://cal.example.com/calid21
S: cap://cal.example.com/calid33
S: .
```

Example #4: CS loses contact with directory server during CALIDEXPAND

```
C: CALIDEXPAND cap://cal.example.com/calid76
S: 2.4 Lost contact with directory server
S: cap://cal.example.com/calid76
```

```
S: cap://cal.example.com/calid3
S: cap://cal.example.com/calid5
S: .
```

7.1.5 CAPABILITY Command

Arguments: none

Data: none

Result: capabilities as described below

The CAPABILITY command returns information about the CAP server given the current state of the connection with the client. The values returned may differ depending on whether the connection is in the Connected or the Authenticated state. The return values may also be different for a secure versus a non-secure connection.

Client implementations SHOULD NOT require any capability name beyond those defined in this specification, and MAY ignore any non-standard, experimental capability names. Non-standard experimental capability names MUST be prefixed with the text "X-". The prefix SHOULD also include a short character vendor identifier For example, "X-FOO-BARCAPABILITY", for the non- standard "BARCAPABILITY" capability of the implementor "FOO". This command may return different results in the Connected state versus the Authenticated state. It may also return different results depending on the UPN.

Capability	Occurs	Description
AUTH	1+	The authentication mechanisms supported. Implementations MUST implement at least
AUTOLOGOUTTIME	0 or 1	An integer value that specifies the default idle time in seconds the CS will wait before disconnecting an idle CUA.If the CS is busy the CS may adjust down the auto-logout timer. If not specified, the value is 15 minutes (900 seconds). A value of zero (0) indicates unlimited connect time is allowed.
CAPVERSION	1	Revision of CAP, MUST include at

		least "1.0". Comma separated values.
ITIPVERSION	0 or 1	Revision of ITIP, MAY be present. If present, it MUST include at least "1.0"
CAR	0 or 1	Indicates level of CAR support. CAR-MIN or CAR-FULL-1. If not specified the default is CAR-FULL-1. Implementations MUST implement CAR-MIN. Implementations MAY implement CAR-FULL-1.
QUERYLEVEL	0 or 1	Indicates level of SQL support. SQL-MIN or SQL-92. If not specified the default is SQL-92. Implementations MUST implement SQL-MIN. Implementations MAY implement SQL-92.
MAXICALOBJECTSIZE	0 or 1	An integer value that specifies The largest ICAL object the server will accept in bytes. Objects larger than this will be rejected. A value of zero (0) indicates unlimited. The default is zero (0) if not specified.
MAXDATE	0 or 1	The datetime value in GMT beyond which the server cannot accept. If not specified the default is 99991231T235959Z.
MAXCALIDEXPANDLIST	0 or 1	An integer value that specifies the maximum number of CalIDs that can be returned by the CALIDEXPAND Command. A value of zero (0) indicates unlimited which is the default if not specified.
MAXUPNEXPANDLIST	0 or 1	An integer value that specifies the maximum number of UPNs that can be returned by the UPNEXPAND Command. A value of zero (0)

		indicates unlimited which is the default if not specified.
MINDATE	0 or 1	The datetime value prior to which the server cannot accept. If not specified the default is 00000101T000000Z.
PIPELINE	0 or 1	An integer value that specifies the maximum number of commands a CUA may send without the CUA waiting for a reply from the CS. If not specified, the default value is one (1). A value of zero (0)indicates unlimited.

Example:

```
C: CAPABILTIY
S: 2.0
S: .
S: CAPVERSION=1.0
S: ITIPVERSION=1.0
S: AUTH=KERBEROS_V4
S: AUTH=DIGEST_MD5
S: .
```

7.1.6 CONTINUE Command

Arguments: latency time in seconds (optional)

Data: none

Result: results from the command in progress 2.0.2 - reply pending.

The CONTINUE command is issued by the client in response to a SENDATA timeout. When a timeout value is specified on the SENDDATA command, the server must issue a reply to the client within the specified time. If the latency time has elapsed prior to the server completing the command it returns a timeout response code. If the client wants the server to continue processing the command it responds with the CONTINUE command.

If latencyTime is present, it must be a positive integer that specifies the maximum number of seconds the client will wait for the next response. If it is omitted, the receiver waits an indefinite

period of time for the response.

In this example, the client requests a response from the server every 10 seconds.

```
C: SENDDATA:10
C: Content-Type:text/calendar; method=READ; component=VEVENT
C:
C: BEGIN:VCAP
# etc
C: END:VCAP
C: .
# after 10 seconds...
S: .
S: 2.0.2
S: .
S: .
C: CONTINUE:10
S: 2.0
S: .
S: Content-type:text/calendar;
S:   Method=RESPONSE;Component=VCAP;
S: Optinfo=VERSION:2.1
S:
S: BEGIN:VCAP

S: VERSION:2.1
S: CALID:cap://cal.example.com/relcal2
# etc.
S: END:VCAP
S: .
```

7.1.7 DISCONNECT Command

Arguments: none

Data:

Result: 2.0

The DISCONNECT command is used by a client to terminate a connection. It always succeeds.

The CUA MUST wait for the CS 2.0 reply to ensure that TCP/IP (which knows nothing about CAP) can be sure the connection should go away. This avoids the CS connection being stuck in TCP-WAIT state.

Example:

```
C: DISCONNECT
# [EDITORS NOTE: Note: should the client now wait for a response
from
the
server
#  before disconnecting? ]
S: 2.0
S: .
S: .
S: <drops connection>
C: <drops connection>
```

7.1.8 IDENTIFY Command

Arguments: Identity to assume

Data: None

Result: 2.0 .4 Identity not permitted

The "IDENTIFY" command allows the CUA to select a new identity to be used for calendar access. This command may only be called in the Identified State.

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

7.1.9 SENDDATA Command

Arguments: [latencyTime]

Data: a [MIME] encapsulated iCalendar object

Result: 2.0.1 - Server will now accept input until <CRLF>.<CRLF> is encountered.

The SENDDATA command is used to send calendar requests and commands to the server. After a response code of 2.0.1 is issued the CUA sends a [MIME] encapsulated iCalendar object to the server. The end of this [MIME] data is signaled by the special sequence <CRLF>.<CRLF>
.

7.1.10 STARTTLS Command

Arguments: None

Data: None

Result: 2.0 5 TLS not supported

The "STARTTLS" command is issued by the CUA to indicate to the CS that it wishes to negotiate transport level security using [TLS]. If the CS does not support TLS it returns status code 6.5. If the CS supports TLS it issues an initial response of 2.0.12 indicating that the CUA should proceed with TLS negotiation. Once the TLS negotiation is complete the server returns the response code 2.0.

After issuing the "STARTTLS" command the CUA issues the "AUTHENTICATE" command. The SASL external mechanism may be used if the CUA wishes to use the authentication id which was used in the TLS negotiation.

The CUA MUST NOT issue a "STARTTLS" if it has already issued an "AUTHENTICATE" or "STARTTLS" command in this session. If a CUA does this the CS must terminate the session.

The following examples illustrate the use of the "STARTTLS" command:

Unsupported TLS:

```
C: STARTTLS
S: 6.5
```

Supported TLS:

```
C: STARTTLS
S: 2.0.12
 <tls negotiation>
S: 2.0

S: .
S: .
```

7.1.11 UPNEXPAND Method

Arguments: UPN

Data: no specific data for this command

Result: 2.0 - success 2.1 Permission Denied 2.2 Requested UPN not found 2.3 Result exceeds MAXUPNEXPANDLIST 2.4 Misc. UPNEXPAND error

Return the members of the argument UPN. Successful result yields one or more CalIDs. More than one CalID returned implies that the UPN was a UG.

If the number of items exceeds MAXUPNEXPANDLIST, then up to MAXUPNEXPANDLIST items are returned along with result 2.3.

If the result is 2.4, then as many of the items as possible will be returned. It is possible that no items will be returned. The 2.4 response may have optional text describing the problem. Any such text MUST be in the language and charset that is defined by the calendar store properties LANGUAGE and CHARSET.

If calendar store CHARSET is not set, but the LANGUAGE property is set, then the error message will be in LANGUAGE in the UTF-8 charset. If calendar store LANGUAGE property is not set, then the CS MUST NOT return any text with the results.

Example #1: Request lookup of a UPN which is a CU

```
C: UPNEXPAND upn@example.com
S: 2.0
S: upn@example.com
S: cap://cal.example.com/calid3
S: .
```

Example #2: Request lookup of UPN which is a UG

```
C: UPNEXPAND group@example.com
S: 2.0
S: group@example.com
S: cap://cal.example.com/calid3
S: cap://cal.example.com/calid6
S: cap://cal.example.com/calid1342
S: .
```

Example #3: Request lookup exceeds MAXUPNEXPANDLIST

```
C: UPNEXPAND group@example.com
S: 2.3 Lookup resulted in too much data
S: group@example.com
S: cap://cal.example.com/calid3
S: cap://cal.example.com/calid12
S: cap://cal.example.com/calid21
S: cap://cal.example.com/calid33
```

```
S: .
```

Example #4: CS loses contact with directory server during UPNEXPAND

```
C: UPNEXPAND group@example.com
S: group@example.com
S: 2.4 Lost contact with directory server
S: cap://cal.example.com/calid3
S: cap://cal.example.com/calid5
S: .
```

7.1.12 NOOP Command

Arguments: none

Data: none

Result: 2.0 - success

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

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

[EDITORS NOTE: need to add NOOP in the state diagram?]

Example:

```
C: NOOP

S: 2.0
S: .
```

7.2 Application Protocol Commands

7.2.1 Calendaring Commands

The following methods provide a set of calendaring commands in CAP. Calendaring commands (or methods) allow a CU to directly manipulate a calendar.

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

7.2.1.1 Restriction Tables

Commands are sent to CAP servers encapsulated in iCalendar objects. The reply to these commands are also encapsulated in iCalendar objects. 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.

There will be a REQUEST-STATUS for each VCALENDAR and component created, modified, deleted, or requested. The number of REQUEST-STATUS values returned MUST be equal to the number of TARGETS times the number of objects in the command. The responses MUST be ordered first by TARGET then by the order of the objects as supplied in the command.

7.2.1.2 CREATE Method

Arguments: none

Data: no specific data for this command

Result:

2.0 - successfully created the component or calendar

6.0 - Permission denied

6.1 - Container(s) not found

6.2 - Calendar or component already exists

6.3 - Bad args

The CREATE method is used to create a new iCalendar object of type objtype. The property TARGET specifies the container(s) for the create. The type of object wrapped inside of the VCALENDAR/CREATE object is the object type(s) being created. When creating a new calendar at the top level, the CSID is specified. Otherwise the container will be a CalID.

Restriction table

Component/Property	Presence	Comment
VCAP	1+	The CUA can send up PIPELINE commands without processing a response
. VERSION	1	MUST be 2.0
. [IANA-PROP]	0+	any IANA registered property
. VCOMMAND	1	MUST at least one container (VCALENDAR, VCAR, VQUERY, VEVENT, VTODO, VJOURNAL)
. . CMDID	0 or 1	If CMDID is not supplied, then there must not be pending replies to previous command.
. . [IANA-PROP]	0+	any IANA registered property
. . METHOD	1	MUST be CREATE.
. . TARGET	1+	MUST be the CSID or CALID
. . VCALENDAR	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	
. . . MAXRESULTS	0 or 1	
. . . MAXSIZE	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	
. . . METHOD	1	<<placeholder. it may move to meta-info>>
. . . 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	0 or 1	MUST be 1 if multiple VALARMs are present in same component.
. . . . 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
. . . METHOD           1       <<placeholder. it may move
                               to meta-info>>
. . . 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        0 or 1  MUST be 1 if multiple
                               VALARMs are present in
                               same component.
. . . . 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
          . . . METHOD           1        <<placeholder. it may move
                                          to meta-info>>
          . . . 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+
          . . . 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

Server Restriction Table for the CREATE command

Component/Property	Presence	Comment
VCAP	1+	
. VCALENDAR	1+	
. . TARGET	1	
. . VERSION	1	MUST be 2.0

```
. . CMDID              0 or 1   MUST be returned if the
                                request contained
                                a CMDID
. . REQUEST-STATUS     0        if not creating a calendar
                       1+       if creating a calendar

. . . VCAR             0+
. . . . REQUEST-STATUS   1+
. . . . *              0        No other VCAR properties
                                are present
.
. . . VCOMMAND         0

. . . VEVENT           0+
. . . . REQUEST-STATUS    1+
. . . . *              0        No other VEVENT properties
                                are present
. . . . VALARM         0        if VEVENT was successfully
                                saved
                       1+       if there were errors saving
                                alarms
. . . . . REQUEST-STATUS  1+

. . . VFREEBUSY        0+
. . . . REQUEST-STATUS    1+
. . . . *              0        No other VFREEBUSY
                                properties are present

. . . VJOURNAL         0+
. . . . REQUEST-STATUS    1+
. . . . *              0        No other VJOURNAL
                                properties are present

. . . VQUERY           0+
. . . . REQUEST-STATUS    1+
. . . . *              0        No other VQUERY properties
                                are present

. . . VTODO            0+
. . . . REQUEST-STATUS    1+
. . . . *              0        No other VTODO properties
                                are present

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

7.2.1.2.1 Creating New Calendars

Example to create two new calendars different containers. In the following example, the client is in the Authenticated state with CS cal.example.com.

```
C: SENDDATA
C: CONTENT-TYPE: text/calendar; method=CREATE;
C:   component=VCOMMAND
C: Content-Transfer-Encoding:7bit
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:CREATE
C: TARGET:cap://cal.example.com/
C: TARGET:relcal4
C: BEGIN:VCALENDAR
C: RELCALID:relcalz1
C: NAME:CHARSET=us-ascii;LANGUAGE=EN-us:Bill's Soccer Team
C: OWNER:bill
C: CALMASTER:mailto:bill@example.com
C: TZID:US_PST
C: BEGIN:VCAR
C: CARID:12345
C: GRANT:UPN=bill;ACTION=*;OBJECT=*
C: END:VCAR
C: END:VCALENDAR
C: BEGIN:VCALENDAR
C: RELCALID:relcalz2
C: NAME:CHARSET=us-ascii;LANGUAGE=EN-us:Mary's personal
C:   calendar
C: OWNER:mary
C: CALMASTER:mailto:mary@example.com
C: TZID:US_PST
C: BEGIN:VCAR
C: CARID:12346
C: GRANT:UPN=mary;ACTION=*;OBJECT=*
C: END:VCAR
C: END:VCALENDAR
C: END:VCOMMAND
C: END:VCAP
C: .
S: 2.0
S: Content-Type:text/calendar; method=RESPONSE;
S:   OPTINFO="CMDID:abcde"
#  This 2.0 is the transport reply status and ends with a
#  CRLF.CRLF (below)
```

```
S: BEGIN:VCAP
S: METHOD:RESPONSE
S: TARGET:cap://cal.example.com/
S: REQUEST-STATUS:2.0
S: END:VCAP
S: BEGIN:VCAP
S: METHOD:RESPONSE
S: REQUEST-STATUS:2.0
S: END:VCAP
S: .
```

Example to create a new component.

```
C: SENDDATA
C: Content-Type:text/calendar; method=CREATE;
C:   charset=US-ASCII
C: Content-Transfer-Encoding:7bit
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: CMDID:abcde
C: BEGIN:VCOMMAND
C: METHOD:CREATE
C: TARGET:cap://cal.example.com/relcal1
C: TARGET:relcal2
C: BEGIN:VEVENT

C: DTSTART:99990307T180000Z
C: UID:abcd12345
C: DTEND:99990307T190000Z
C: SUMMARY:Important Meeting
C: END:VEVENT
C: END:VCOMMAND
C: END:VCAP
C: .
S: 2.0
S: Content-Type:text/calendar; method=RESPONSE;
S:   OPTINFO="CMDID:abcde"
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: CMDID:abcde
S: METHOD:RESPONSE
S: BEGIN:VEVENT
S: TARGET::cap://cal.example.com/relcal1
S: REQUEST-STATUS:2.9
S: END:VEVENT
S: BEGIN:VEVENT
```

```
S: REQUEST-STATUS:2.9
S: REQUEST-STATUS:2.10
S: END:VEVENT
S: END:VCAP
S: .
```

The response contains the calendar (CALID) and UID of the component so that the CUA can match up the TARGET from multiple objects created on multiple calendards (TARGETs).

7.2.1.2.2 Creating a new VQUERY

This example creates a stored VQUERY that selects all unprocessed scheduling entries. QUERYNAME must not exist in the TARGET calendar.

```
C: SENDDATA
C: Content-Type:text/calendar; method=CREATE; charset=US-ASCII
C: Content-Transfer-Encoding:7bit
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: CMDID:abcde-2
C: METHOD:CREATE
C: TARGET:cap://cal.example.com/relcal1
C: BEGIN:VQUERY
C: BEGIN:VQUERY
C: QUERYNAME:GetAlliTIPinQueue
C: QUERY:SELECT UID FROM VEVENT,VTODO WHERE METHOD != 'CREATE'
C: END:VQUERY
C: END:VEVENT
C: END:VCAP
C: .
S: 2.0
S: Content-Type:text/calendar; method=RESPONSE; OPTINFO="CMDID:abcde"
S:
S: BEGIN:VCAP
S: CMDID:abcde-2
S: METHOD:RESPONSE
S: TARGET::cap://cal.example.com/relcal1
S: BEGIN:VQUERY
S: REQUEST-STATUS:2.0;
S: END:VQUERY
S: END:VCAP
S: .
```

[EDITORS NOTE - can return QUERYNAME already exists error]

7.2.1.3 DELETE Method

Arguments: none

Data: no specific data for this command

Result:

2.0 - successfully deleted the component or calendar

Permission

Calendar or component not found

Bad args

Container(s) not found

The DELETE method is used to delete a calendar or component. The TARGET properties specify the container(s) for the delete. When deleting a calendar at the top level, the CSID is specified. Otherwise the container will be a CalID.

Restriction Table

Component/Property	Presence	Comment
VCAP	1+	The CUA can send up PIPELINE commands without processing a response
VERSION	1	MUST be 2.0
VCOMMAND	1	
CMDID	0 or 1	If CMDID is not supplied, then there must not be pending replies to previous command.
METHOD	1	MUST be DELETE.
TARGET	1+	MUST be a the CSID or CALID
VQUERY	0+	
EXPAND	0 ?	
MAXRESULTS	0 or 1	Limit the solution set to no more than this many entries.

MAXSIZE	0 ?	
QUERYNAME	1	Name by which this query is referenced
QUERY	1	The query

Server Restriction Table for the DELETE command

Component/Property	Presence	Comment
VCAP	1+	
TARGET	1	
VERSION	1	MUST be 2.0
CMDID	0 or 1	MUST be returned if the request contained a CMDID
VCALENDAR		Only if VCALENDARs were deleted
REQUEST-STATUS	1	
*	1+	No other VALARM properties are present
VALARM	0+	Only if VALARM components were deleted
REQUEST-STATUS	1	
*	0	No other VALARM properties are present
VCAR	0+	Only if VCAR components were deleted
REQUEST-STATUS	1	
*	0	No other VCAR properties are present
VEVENT	0+	Only if VEVENT components were deleted
REQUEST-STATUS	1+	
*	0	No other VEVENT properties are present
VFREEBUSY	0	
REQUEST-STATUS	1+	
*	0	No other VFREEBUSY properties are present

VJOURNAL	0+	Only if VJOURNAL components were deleted
REQUEST-STATUS	1+	
*	0	No other VJOURNAL properties are present
VQUERY	0+	Only if VQUERY components were deleted
REQUEST-STATUS	1+	
*	0	No other VQUERY properties are present
VTIMEZONE	0+	Only if VTIMEZONE components were deleted
REQUEST-STATUS	1+	
*	0	No other VQUERY properties are present
VTODO	0+	Only if VTODO components were deleted
REQUEST-STATUS	1+	
*	0	No other VTODO properties are present

[EDITORS NOTE: Issues:

- Currently CAP requires that the server return a response status. However, if there are multiple components deleted, should the UID also be returned?

- VQUERY EXPAND and MAXSIZE parameters do not seem to apply to deletion?

- 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?

- In the example in CAP where a calendar is deleted all the server returns is 2.0, nothing else?

- We should not be able to delete any VFREEBUSY components?

- Can we delete multiple calendars?

- Currently one can not delete a VCALENDAR and some other component in the same DELETE command. This seems OK. Anyone see any need to be able to do this?

- Should the MAXRESULTS property of VQUERY apply to deletion? We can use it to delete only the first n components that match.]

Example to delete a VEVENT with UID 'abcd12345':

```
  C: SENDDATA
  C: Content-Type:text/calendar; method=DELETE;
component=VCOMMAND
  C: Content-Transfer-Encoding:7bit
  C:

C: BEGIN:VCAP
  C: BEGIN:VCOMMAND
  C: METHOD:DELETE
  C: TARGET:cap://cal.foo.com/bill
  C: BEGIN:VQUERY
  C: QUERY:SELECT UID FROM VEVENT WHERE UID = 'abcd12345'
  C: END:VQUERY
  C: END:VCOMMAND
  C: END:VCAP
  C: .
  S: 2.0
  S: Content-Type:text/calendar; method=DELETE;
  S:  component=VCOMMAND
  S:
  S: BEGIN:VCAP
  S: METHOD:RESPONSE
  S: BEGIN:VEVENT
  S: REQUEST-STATUS:2.0
  S: END:VEVENT
  S: END:VCAP
  S: .
  C: SENDDATA
  C: Content-Type:text/calendar; method=DELETE;
  C:  component=VCOMMAND
  C: Content-Transfer-Encoding:7bit
  C:
  C: BEGIN:VCAP
  C: BEGIN:VCOMMAND
  C: METHOD:DELETE
  C: TARGET:cap://cal.foo.com/MrBill
  C: END:VCOMMAND
```

```
C: END:VCAP
C: .
S: 2.0
S: .
```

7.2.1.4 GENERATEUID Method

Arguments: Number of UIDs to generate.

Data: new uids

Result: 2.0

GENERATEUID returns one or more new unique identifier which MUST be unique on the server's calendar store. It is recommended that the return value be a globally unique id.

Example:

```
C: GENERATEUID 2
S: 2.0
S: abcde1234567-asdf-lkhh
S: abcde1234567-asdf-3455
S: .
```

[EDITORS NOTE: this example needs work. It's not packaged right]

7.2.1.5 MODIFY Method

Arguments: none

Data: no specific data for this command

Result:

2.0 - successfully modified the component or calendar

Permission

Calendar or component not found

Bad args

Container(s) not found

The MODIFY method is used to change an existing calendar or component. TARGET specify the container(s) of the modification.

When modifying a calendar at the top level, the CSID is specified. Otherwise the container will be a CalID.

The format of the request is two or three containers inside of a VCOMMAND container:

```
BEGIN:VCOMMAND
[VQUERY]
OLD-VALUES
NEW-VALUES
END:VCOMMAND
```

If a VQUERY is present, then only objects matching the query results are modified.

Restriction Table

Component/Property	Presence	Comment
VCAP	1+	The CUA can send up PIPELINE commands without processing a response
. VERSION	1	MUST be 2.0
. [IANA-PROP]	0+	any IANA registered property
. VCOMMAND	1	MUST have at least one container (VCALENDAR, VCAR, VQUERY, VEVENT, VTODO, VJOURNAL)
. . CMDID	0 or 1	If CMDID is not supplied, then there must not be pending replies to previous command.
. . [IANA-PROP]	0+	any IANA registered property
. . METHOD	1	MUST be MODIFY
. . TARGET	1+	MUST be the CALID
. . VCALENDAR	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	
. . . MAXRESULTS	0 or 1	
. . . MAXSIZE	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	
. . . METHOD	1	<<placeholder. it may move to meta-info>>
. . . 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	0 or 1	MUST be 1 if multiple VALARMs are present in same component.
. . . . 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	
. . . METHOD	1	<<placeholder. it may move to meta-info>>
. . . 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	0 or 1	MUST be 1 if multiple VALARMs are present in

		same component.
. . . . 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	
. . . METHOD	1	<<placeholder. it may move to meta-info>>
. . . 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+	
. . . 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
```

Server Restriction Table for the MODIFY command

Component/Property	Presence	Comment
VCAP	1+	
. VCALENDAR	1+	
. . TARGET	1	
. . VERSION	1	MUST be 2.0
. . CMDID	0 or 1	MUST be returned if the request contained a CMDID
. . REQUEST-STATUS	0	if not creating a calendar
	1+	if creating a calendar
. . . VCAR	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VCAR properties are present
.		
. . . VCOMMAND	0	
. . . VEVENT	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VEVENT properties are present
. . . . VALARM	0	if VEVENT was successfully saved
	1+	if there were errors saving alarms
. . . . . REQUEST-STATUS	1+	
. . . VFREEBUSY	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VFREEBUSY properties are present
. . . VJOURNAL	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VJOURNAL properties are present
. . . VQUERY	0+	
. . . . REQUEST-STATUS	1+	

```
            . . . . *             0        No other VQUERY properties
                                           are present

            . . . VTODO           0+
            . . . . REQUEST-STATUS    1+
            . . . . *             0        No other VTODO properties
                                           are present
            . . . . VALARM        0        if VTODO was successfully
                                           saved
                                  1+       if there were errors saving
                                           alarms
            . . . . . REQUEST-STATUS   1+
```

[EDITORS NOTES: Issues: freebusy - a cap server should dynamically calculate freebusy information we recommend that you cannot create, modify, or delete freebusy composers]

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

```
C: SENDDATA
C: Content-type:text/calendar; Method=MODIFY;
C:  Component=VCOMMAND
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:MODIFY
C: TARGET:relcal2
C: BEGIN:VCOMMAND
C: BEGIN:VQUERY
C: QUERY:SELECT UID FROM VEVENT WHERE UID = 'abcd12345'
C: END:VQUERY
C: BEGIN:VEVENT
C: DTSTART:19990421T160000Z
C: DTEND:19990421T163000Z
C: LOCATION:Joe's Diner
C: END:VCOMMAND
C: END:VEVENT

C: BEGIN:VEVENT
C: DTSTART:19990421T160000Z
C: DTEND:19990421T163000Z
C: END:VEVENT
C: END:VCOMMAND
C: END:VCAP
C: .
```

```
S: 2.0 cap://cal.example.com/relcal2
```

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

```
C: SENDDATA
C: Content-type:text/calendar; Method=MODIFY;
C:  Component=VCOMMAND
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:MODIFY
C: TARGET:relcal2
C: BEGIN:VCOMMAND
C: BEGIN:VEVENT
C: LOCATION:Building 6
C: END:VEVENT
C: BEGIN:VEVENT
C: LOCATION:New office lobby
C: END:VEVENT
C: END:VCOMMAND
C: END:VCOMMAND
C: END:VCAP
C: .
S: 2.0 cap://cal.example.com/relcal2
```

7.2.1.6 MOVE Method

Arguments: ContainerId

Data: data as described below

Result:

2.0 - success

2.2 - will attempt operation on the remote CAP server

Permission

Calendar already exists

Bad args

Parent Calendar(s) not found

The format of the command is:

```
BEGIN:VCALENDAR
METHOD:MOVE
...
TARGET:target-name
BEGIN:VCOMMAND
BEGIN:VCALENDAR
PARENT:<old-parent>
END:VCALENDAR
BEGIN:VCALENDAR
PARENT:<new-parent>
END:VCALENDAR
END:VCOMMAND
END:VCALENDAR
```

This looks similar to a MODIFY command. Except the CS MUST ensure that VCARs are still valid after the move. And the CS MUST update the CHILDREN list in the new and old parent containers.

This method is used to move a calendar within the CS's hierarchy of calendars.

Restriction Table

Component/Property	Presence	Comment
VCAP	1+	The CUA can send up PIPELINE commands without processing a response
. VERSION	1	MUST be 2.0
. VCOMMAND	1	MUST have at least one VCALENDAR
. . CMDID	0 or 1	If CMDID is not supplied, then there must not be pending replies to previous command.
. . [IANA-PROP]	0+	any IANA registered property
. . METHOD	1	MUST be MOVE.
. . TARGET	1	MUST be a the CSID or CALID

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

Server Restriction Table for the MOVE command

Component/Property	Presence	Comment
VCAP	1+	
. VCALENDAR	1+	
. . TARGET	1	
. . VERSION	1	MUST be 2.0
. . CMDID	0 or 1	MUST be returned if the request contained a CMDID
. . 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?)

2) Should we also be able to move components from one calendar to another? What if the calendars are owned by different users? (With components one can do a copy, then delete the original.)

3) There was very little information about MOVE in CAP. Why was it added? Was there some major need for it?

4) Can one move multiple calendars into one calendar?]

An example of moving a calendar from Nellis to Area-51

```
C: SENDDATA
C: Content-type:text/calendar; Method=MODIFY; Component=VCOMMAND
```

```
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: METHOD:MOVE
C: TARGET:Nellis
C: BEGIN:VCOMMAND
C: BEGIN:VCALENDAR
C: PARENT:Department-33
C: END:VCALENDAR
C: BEGIN:CALENDAR
C: PARENT:Area-51
C: END:VCALENDAR
C: END:VCOMMAND
C: END:VCAP
C: .
S: 2.0
S: Content-Type:text/calendar; method=RESPONSE
S:
S: BEGIN:VCAP
S: METHOD:RESPONSE
S: TARGET:relcal2
S: BEGIN:VCALENDAR
S: REQUEST-STATUS:2.0;
S: END:VCALENDAR
S: END:VCAP
S: .
```

7.2.1.7 READ Method

Arguments: ContainerId

Data: data as described below

Result:

2.0 - - successful and the requested data follows

2.2 - will attempt read on the remote CAP server

Permission

Bad args

Restriction Table

Component/Property	Presence	Comment

VCAP	1+	The CUA can send PIPELINE commands without processing a response
. VERSION	1	MUST be 2.0
. [IANA-PROP]	0+	any IANA registered property
. VCOMMAND	1	MUST have at least one container (VCALENDAR, VCAR, VQUERY, VEVENT, VTODO, VJOURNAL)
. . CMDID	0 or 1	If CMDID is not supplied, then there must not be pending replies to previous command.
. . [IANA-PROP]	0+	any IANA registered property
. . METHOD	1	MUST be READ
. . TARGET	1+	MUST be the CALID
. . VCALENDAR	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	
. . VQUERY	1	
. . . EXPAND	0 or 1	
. . . MAXRESULTS	0 or 1	
. . . MAXSIZE	0 or 1	
. . . QUERYNAME	1	
. . . QUERY	1	
. . . [IANA-PROP]	0+	any IANA registered property
. . VEVENT	0	
. . VTODO	0	
. . VJOURNAL	0	
. . 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
```

Server Restriction Table for the READ command

```
          Component/Property    Presence Comment
          --------------------- -------- --------------------------

          VCAP                   1+
          . VCALENDAR            1+
          . . TARGET             1
```

. . VERSION	1	MUST be 2.0
. . CMDID	0 or 1	MUST be returned if the request contained a CMDID
. . REQUEST-STATUS	0	if not creating a calendar
	1+	if creating a calendar
. . . VCAR	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VCAR properties are present
.		
. . . VCOMMAND	0	
. . . VEVENT	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VEVENT properties are present
. . . . VALARM	0	if VEVENT was successfully saved
	1+	if there were errors saving alarms
. . . . . REQUEST-STATUS	1+	
. . . VFREEBUSY	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VFREEBUSY properties are present
. . . VJOURNAL	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VJOURNAL properties are present
. . . VQUERY	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VQUERY properties are present
. . . VTODO	0+	
. . . . REQUEST-STATUS	1+	
. . . . *	0	No other VTODO properties are present
. . . . VALARM	0	if VTODO was successfully saved

```
                                  1+     if there were errors saving
                                         alarms
          . . . . . REQUEST-STATUS 1+
```

Read 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 returns two VEVENTS that match in that TARGET, the second result returns only one VEVENT for the second TARGET.

```
C: SENDDATA
C: Content-type:text/calendar; Method=READ; Component=VQUERY
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:READ
C: CMDID:xyz12345
C: TARGET:relcal2
C: TARGET:cap://bobo.ex.com/relcal3
C: BEGIN:VQUERY
C: QUERY:SELECT DTSTART,DTEND,SUMMARY,UID
    FROM VEVENT
    WHERE DTEND >= '19990310T080000Z'
    AND DTSTART <= '19990310T190000Z'
    AND METHOD = 'CREATE'
C: END:VQUERY
C: END:VCOMMAND
C: END:VCAP
C: .
S: 2.0 cap://cal.example.com/relcal2
S: Content-type:text/calendar; Method=RESPONSE;
    Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: METHOD:RESPONSE
S: BEGIN:VEVENT

S: DTSTART:19990310T090000Z
S: DTEND:19990310T100000Z
S: UID:abcxyz12345
S: SUMMARY:Meet with  Sir Elton
S: END:VEVENT
```

```
S: BEGIN:VEVENT
S: DTSTART:19990310T130000Z
S: DTEND:19990310T133000Z
S: UID:abcxyz8999
S: SUMMARY:Meet with  brave brave Sir Robin
S: END:VEVENT
S: END:VCAP
S: .
S: 2.0 cap://bobo.ex.com/relcal3
S: Content-type:text/calendar; Method=RESPONSE;
S:    Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: METHOD:RESPONSE
S: BEGIN:VEVENT
S: DTSTART:19990310T140000Z
S: DTEND:19990310T150000Z
S: UID:123456asdf
S: SUMMARY:Summer Budget
S: END:VEVENT
S: END:VCAP
S: .
```

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 events, 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: 2.0 cap://bobo.ex.com/sally
S: Content-type:text/calendar;
Method=RESPONSE;
S:    Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: BEGIN:VEVENT
S: REQUEST-STATUS:3.8
S: END:VEVENT
S: END:VCAP
```

```
S: .
```

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 requester has been denied access.

```
S: 2.0 cap://bobo.ex.com/sally
S: Content-type:text/calendar; Method=RESPONSE;
S: Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: END:VCAP
S: .
```

Find alarms within a range of time for booked (METHOD = CREATE) VEVENTs.

```
C: SENDDATA
C: Content-type:text/calendar; Method=READ; Component=VQUERY
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:READ
C: CMDID:xyz12345
C: TARGET:relcal2
C: TARGET:cap://bobo.ex.com/relcal3
C: BEGIN:VQUERY
C: QUERY:SELECT DTSTART,DTEND,SUMMARY,UID, VALARM.*
    FROM VEVENT,VTODO
    WHERE VALARM.TRIGGER >= '19990310T080000Z'
    AND VALARM.TRIGGER <= '19990310T190000Z'
    AND METHOD = 'CREATE'
C: END:VQUERY
C: END:VCOMMAND
C: END:VCAP
C: .
S: 2.0 cap://bobo.ex.com/relcal3
S: Content-type:text/calendar; Method=RESPONSE;
S: Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: METHOD:RESPONSE
S: CMDID:xyz12345
```

```
S: TARGET:cap://bobo.ex.com/relcal3
S: BEGIN:VEVENT
S: DTSTART:19990310T130000Z
S: DTEND:19990310T133000Z

S: UID:abcxyz8999
S: SUMMARY:Meet with brave brave Sir Robin
S: BEGIN:VALARM
S: TRIGGER:19990310T132500Z
S: SUMMARY:Almost time..
S: ACTION:DISPLAY
S: END:VALARM
S: END:VEVENT
S: END:VCAP
S: .
S: 2.0 cap://bobo.ex.com/relcal2
S: Content-type:text/calendar; Method=RESPONSE;
S: Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: METHOD:RESPONSE
S: CMDID:xyz12345
S: TARGET:cap://bobo.ex.com/relcal2
S: BEGIN:VEVENT
S: REQUEST-STATUS:2.0
S: END:VEVENT
S: END:VCAP
S: .
```

7.2.1.7.1 Query With A Stored Query

This will run a stored VQUERY and return the results of the VQUERY.

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

This will fetch all calendar store properties. This MUST NOT return any VCALENDARs.

```
BEGIN:VCAP
VERSION:2.1
METHOD:READ
CMDID:-2ir-
```

```
TARGET:cap://bobo.ex.com
BEGIN:VQUERY
QUERY:SELECT  * FROM CALSTORE
END:VQUERY
END:VCAP
```

This will fetch all calendar properties. This MUST NOT return any components.

```
BEGIN:VCAP
VERSION:2.1
METHOD:READ
CMDID:-2ir-
TARGET:cap://bobo.ex.com/relcal4
BEGIN:VQUERY
QUERY:SELECT  * FROM VCALENDAR
END:VQUERY
END:VCAP
```

This will fetch all stored VQUERYs.

```
BEGIN:VCAP
VERSION:2.1
METHOD:READ
CMDID:didmc
TARGET:cap://bobo.ex.com/relcal4
BEGIN:VQUERY
QUERY:SELECT  * FROM VQUERY
END:VQUERY
END:VCAP
```

7.2.2 Scheduling Commands

The following provide a set of scheduling commands (or methods) in CAP. 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].]

7.2.2.1 Reading 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 automaticly 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:VCAP
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:VCAP
```

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: SENDDATA
C: Content-type:text/calendar; Method=READ; Component=VQUERY
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:READ
C: CMDID:xyz12345
C: TARGET:relcal2
C: BEGIN:VQUERY
C: QUERY:SELECT * WHERE FROM VEVENT METHOD != 'CREATE'
C: END:VQUERY
C: END:VCOMMAND
C: END:VCAP
```

C: .

The the CUA and CU could determine which scheduling items were to remain on the calendar. Each scheduling component could be deleted or updated with METHOD:MODIFY to change the METHOD from PUBLISH, REQUEST, REPLY, ADD, CANCEL, REFRESH, COUNTER, or DECLINE-COUNTER to what the CU and CUA had decided.

In some cases the CUA could automaticly do the work and inform the CU. An example of this is CANCEL. If configured to process METHOD:CANCEL it could METHOD:DELETE the component an 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 METHOD:DELETE 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.

7.2.2.2 PUBLISH

Arguments:

Data: data as described below

Result:

2.0 - success

2.2 - will attempt operation on the remote CAP server

Permission

Calendar already exists

Bad args

Parent Calendar(s) not found

If the CU wishes to keep the published entry. A METHOD:MODIFY changing the entries METHOD from PUBLISH to CREATE would be done, booking the entry. Or METHOD:DELETE if the CU did not wish this

scheduled item to exist in their calendar.

7.2.2.3 REQUEST

Arguments:

Data: data as described below

Result:

2.0 - success

2.2 - will attempt operation on the remote CAP server

Permission

Calendar already exists

Bad args

Parent Calendar(s) not found

This as described in [iTIP] and would be modified just like PUBLISH above.

7.2.2.4 REPLY

Arguments:

Data: data as described below

Result:

2.0 - success

2.2 - will attempt operation on the remote CAP server

Permission

Calendar already exists

Bad args

Parent Calendar(s) not found

7.2.2.5 ADD

Arguments:

Data: data as described below

Result:

2.0 - success

2.2 - will attempt operation on the remote CAP server

Permission

Calendar already exists

Bad args

Parent Calendar(s) not found

7.2.2.6 CANCEL

Arguments:

Data: data as described below

Result:

7.2.2.7 REFRESH

Arguments:

Data: data as described below

Result:

This as described in [iTIP] and would be modified just like PUBLISH above. The CS MAY automaticly send out REFRESH replies via iMIP or CAP if able, then METHOD:DELETE the REFRESH. But only if there are no other pending scheduled entries for this calendar that may effect what REFRESH would send back. If the CS is not able to reply to the REFRESH request then it is left in the scheduling queue until the CUA and CU processes the queue. At the point where there are no outstanding scheduled command that would effect the reply results, the CS may then automaticly send the reply to the REFRESH request.

7.2.2.8 COUNTER

Arguments:

Data: data as described below

Result:

7.2.2.9 DECLINECOUNTER

Arguments:

Data: data as described below

Result:

7.2.3 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.

7.2.3.1 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.foo.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.foo.com".

```
BEGIN:VCAP
VERSION:2.1

CMDID:xhj-dd
BEGIN:VCOMMAND
METHOD:REQUEST
```

```
  TARGET:cap://cal.foo.com/relcal2
  TARGET:relcal3
  BEGIN:VEVENT
  UID:abcd12345
  DTSTART:19990307T180000Z
  DTEND:19990307T190000Z
  ORGANIZER:cap://cal.foo.com/relcal1
  ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-
ACTION:cap://cal.foo.com/relcal2
  ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-
ACTION:cap://cal.foo.com/relcal3
  SUMMARY:Important Meeting
  END:VEVENT
  END:VCOMMAND
  END:VCAP
```

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.foo.com/relcal1
  ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-ACTION:cap://cal.foo.c
   om/relcal2
  ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-ACTION:cap://cal.foo.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:

```
  BEGIN:VCAP
  VERSION:2.1
  BEGIN:VCOMMAND
  METHOD:READ
  CMDID:xhr-de
  TARGET:relcal2
  # or TARGET:relcal3
  BEGIN:VCOMMAND
  BEGIN:VQUERY
  QUERY:SELECT * FROM VEVENT WHERE METHOD = 'REQUEST';
  END:VQUERY
  END:VCOMMAND
  END:VCOMMAND
```

```
END:VCAP
```

In response to this search they get the above event. B-c and C-c must then crack 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:

```
BEGIN:VCAP
VERSION:2.1
BEGIN:VCOMMAND
METHOD:READ
CMDID:xhr-df
TARGET:relcal2

BEGIN:VCOMMAND
BEGIN:VQUERY
QUERY:SELECT * FROM VEVENT WHERE UID = 'abcd12345' AND
 METHOD = 'CREATE'
END:VQUERY
END:VCOMMAND
END:VCOMMAND
END:VCAP
```

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:

```
BEGIN:VCAP
VERSION:2.1
CMDID:xhj-dd
METHOD:REPLY
TARGET:cap://cal.foo.com/relcal1
BEGIN:VEVENT
UID:abcd12345
DTSTART:19990307T180000Z
DTEND:19990307T190000Z
ORGANIZER:cap://cal.foo.com/relcal1
ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.foo.com/relcal2
SUMMARY:Important Meeting
END:VEVENT
END:VCAP
```

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

```
BEGIN:VCAP
VERSION:2.1
CMDID:xhj-dd
METHOD:REPLY
TARGET:cap://cal.foo.com/relcal1
BEGIN:VEVENT
UID:abcd12345
DTSTART:19990307T180000Z
DTEND:19990307T190000Z
ORGANIZER:cap://cal.foo.com/relcal1
ATTENDEE;PARTSTAT=DECLINED:cap://cal.foo.com/relcal3
SUMMARY:Important Meeting
END:VEVENT
END:VCAP
```

It is preferable that C-c store 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:

```
BEGIN:VEVENT
METHOD:REQUEST
UID:abcd12345
DTSTART:19990307T180000Z
DTEND:19990307T190000Z
ORGANIZER:cap://cal.foo.com/relcal1
ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.foo.com/relcal2
ATTENDEE;PARTSTAT=DECLINED:cap://cal.foo.com/relcal3
SUMMARY:Important Meeting
END:VEVENT
```

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

7.2.3.2 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:

```
BEGIN:VCAP
VERSION:2.1
CMDID:xud-pn
```

```
METHOD:REFRESH
TARGET:cap://cal.foo.com/relcal1
BEGIN:VEVENT
UID:abcd12345
ORGANIZER:cap://cal.foo.com/relcal1
ATTENDEE:cap://cal.foo.com/relcal3
DTSTAMP:19990306T202333Z
END:VEVENT
END:VCAP
```

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 up as an iTIP request and sends it to relcal3:

```
BEGIN:VCAP
VERSION:2.1
CMDID: xud-pn
METHOD:REQUEST
TARGET:cap://cal.foo.com/relcal3
BEGIN:VEVENT
UID:abcd12345
DTSTART:19990307T180000Z
DTEND:19990307T190000Z
ORGANIZER:cap://cal.foo.com/relcal1
ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.foo.com/relcal2
ATTENDEE;PARTSTAT=DECLINED:cap://cal.foo.com/relcal3
SUMMARY:Important Meeting
SEQUENCE:0
DTSTAMP:19990306T204333Z
END:VEVENT
END:VCAP
```

7.2.3.3 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:

```
BEGIN:VCAP
VERSION:2.1
CMDID:zzykjjk
METHOD:COUNTER
TARGET:cap://cal.foo.com/relcal1
BEGIN:VEVENT
```

```
UID:abcd12345
DTSTART:19990307T180000Z
DTEND:19990307T190000Z
ORGANIZER:cap://cal.foo.com/relcal1
ATTENDEE;PARTSTAT=DECLINED:cap://cal.foo.com/relcal3
SUMMARY:Important Meeting
LOCATION:La Belle Province
COMMENT:My favorite restaurant, I'll definitely go if it's
there.
END:VEVENT
END:VCAP
```

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:

```
BEGIN:VEVENT
METHOD:CREATE
UID:abcd12345
DTSTART:19990307T180000Z
DTEND:19990307T190000Z
ORGANIZER:cap://cal.foo.com/relcal1
ATTENDEE;PARTSTAT=ACCEPTED:cap://cal.foo.com/relcal2

ATTENDEE;PARTSTAT=DECLINED:cap://cal.foo.com/relcal3
SUMMARY:Important Meeting
LOCATION:La Belle Province
SEQUENCE:1
END:VEVENT
```

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

```
BEGIN:VCAP
VERSION:2.1
CMDID:xup-po
METHOD:REQUEST
```

```
  TARGET:cap://cal.foo.com/relcal2
  TARGET:cap://cal.foo.com/relcal3
  BEGIN:VEVENT
  UID:abcd12345
  DTSTART:19990307T180000Z
  DTEND:19990307T190000Z
  ORGANIZER:cap://cal.foo.com/relcal1
  ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-
ACTION:cap://cal.foo.com/relcal2
  ATTENDEE;RSVP=TRUE;PARTSTAT=NEEDS-
ACTION:cap://cal.foo.com/relcal3
  SUMMARY:Important Meeting
  LOCATION:La Belle Province
  SEQUENCE:1
  DTSTAMP:19990307T054339Z
  END:VEVENT
  END:VCAP
```

7.2.3.4 Declining an iTIP counter

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

```
  BEGIN:VCAP
  VERSION:2.1
  CMDID:xim-ef
  METHOD:COUNTER
  TARGET:cap://cal.foo.com/relcal1
  BEGIN:VEVENT
  UID:abcd12345
  DTSTART:19990307T180000Z
  DTEND:19990307T190000Z
  ORGANIZER:cap://cal.foo.com/relcal1
  ATTENDEE:cap://cal.foo.com/relcal2
  SUMMARY:Important Meeting
  LOCATION:Au Coin Dor=E9
  END:VEVENT
  END:VCAP
```

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

```
  BEGIN:VCAP
  VERSION:2.1
  CMDID:xim-ef
  METHOD:DECLINE-COUNTER
  TARGET:cap://cal.foo.com/relcal2
```

```
BEGIN:VEVENT
DTSTAMP:19990307T093245Z
UID:abcd12345
ORGANIZER:cap://cal.foo.com/relcal1
SEQUENCE:1
END:VEVENT
END:VCAP
```

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.

8. 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	Params	Description
2.0	[CMDID]	varies Success, The parameters vary with the operation and are specified.
2.0.1	[CMDID]	Success, send data, terminate with >CRLF>.>CRLF>
2.0.2	[CMDID]	A reply is pending. It could not be completed in the specified amount of time. The server awaits a CONTINUE or ABORT command. If can optionally be followed by a CMDID. If the SENDDATA object contained a CMDID, then the CS MUST append the CMDID to the 2.0.2 reply for that object.
2.0.3	[CMDID]	In response to the client issuing an ABORT command, this reply code indicates that any command currently underway was successfully aborted. If can optionally be followed by a CMDID. If the SENDDATA object contained a CMDID, then the CS MUST append the CMDID to the 2.0.2 reply for that object.
2.0.6	[CMDID]	An operation is being attempted on a re-mote server. This response indicates that the server has not yet been contacted but an attempt will be made to contact it after the command has been

		sent.
3.1.4	[CMDID]	Capability not supported.
4.1	[CMDID]	Calendar store access denied.
6. 1	[CMDID]	Authenticate failure: unsupported authentication mechanism, credentials rejected.
6.2	[CMDID]	Sender aborted authentication, authentication exchange cancelled.
6.3	[CMDID]	Attempt to create or modify an event such that it would overlap another event in either of the following two circumstances: (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	[CMDID]	[EDITORS NOTE: More are in this memo - add here TODO]
7.0	[CMDID]	A timeout has occurred. The server was unable to complete the operation in the requested time.
8.0	[CMDID]	A failure has occurred in the Receiver that prevents the operation from succeeding.
8.1	[CMDID]	Sent when a session cannot be established because the CAP Server is too busy.
8.2	[CMDID]	Used to signal that an ICAL object has exceeded the server's size limit
8.3	[CMDID]	A DATETIME value was too large to be represented on this Calendar.
8.4	[CMDID]	A DATETIME value was too far in the past to be represented on this Calendar.

8.5	[CMDID]	An attempt was made to create a new object but the unique id specified is already in use.
9.0	[CMDID]	An unrecognized command was received.
10.1	<old-address> <new- address> [CMDID]	Accompanied by an alternate address. The RECIPIENT specified should be contacted at the given alternate address. The referral address MUST follow the reply code.
10.2		The server is shutting down.
10.4		The operation has not be performed because it would cause the resources (memory, disk,CPU, etc) to exceed the allocated quota.
10.5	[CMDID]	The ITIP message has been queued too too long. Delivery has been aborted.

8.1 Examples

8.1.1 Authentication Examples

8.1.1.1 Login Using Kerberos V4

8.1.1.2 Error Scenarios

8.2 Read Examples

8.2.1 Read From A Single Calendar

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

```
C: SENDDATA
C: Content-type:text/calendar; Method=READ; Component=VQUERY
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:READ
C: CMDID:xyz12345
```

```
C: TARGET:cap://cal.example.com/opaqueid99
C: BEGIN:VQUERY
C: QUERY:SELECT DTSTART,DTEND,SUMMARY, UID FROM VEVENT
  WHERE DTEND >= '19990714T080000Z'
  AND DTSTART <= '19990715T080000Z'
C: END:VQUERY
C: END:VCOMMAND
C: END:VCAP
C: .
# this response code means that the transport successfully
# delivered the data.
S: 2.0 ; got the request OK ; really
S: .
S: Content-type:text/calendar; Method=RESPONSE;
S:  Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: METHOD:RESPONSE
S: TARGET:cap://cal.example.com/opaqueid99
S: CMDID:xyz12345
S: REQUEST-STATUS:2.0
S: BEGIN:VEVENT
S: DTSTART:19990714T200000Z
S: DTEND:19990714T210000Z
S: UID:000444888929922
S: SUMMARY:Blah bla
S: END:VEVENT
S: BEGIN:VEVENT
S: UID:0034848098038888989443
S: SUMMARY:meeting
S: DTEND:19990714T233000Z
S: DTSTART:19990714T223000Z
S: END:VEVENT
S: END:VCAP
S: .
```

8.2.2 Read From Multiple Calendars

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: SENDDATA
C: Content-type:text/calendar; Method=READ; Component=VQUERY
C:
```

```
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:READ
C: CMDID:xyz12346
C: TARGET:cap://cal.example.com/opaqueid101
C: TARGET:opaqueid103
C: BEGIN:VQUERY
C: VSCOPE:VEVENT


C: QUERY:SELECT DTSTART,DTEND,SUMMARY,UID FROM VEVENT
C: WHERE DTEND >= 19990714T080000Z AND
C:   DTSTART <= 19990715T080000Z
C: END:VQUERY
C: END:VCOMMAND
C: END:VCAP
C: .
S: 2.0
S: .
S: Content-Type:multipart/mixed;
S:  boundary="--FEE3790DC7E35189CA67"
S:
S: ----FEE3790DC7E35189CA67
S: Content-type:text/calendar; Method=RESPONSE;
S: Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
S: VERSION:2.1
S: METHOD:RESPONSE
S: TARGET:cap://cal.example.com/opaqueid103
S: CMDID:xyz12346
S: REQUEST-STATUS:2.0
S: BEGIN:VEVENT
S: UID:0034848098038888989443
S: SUMMARY:meeting
S: DTEND:19990714T233000Z
S: DTSTART:19990714T223000Z
S: END:VEVENT
S: END:VCAP
S:
S: ----FEE3790DC7E35189CA67CE2C
S: Content-type:text/calendar; Method=RESPONSE;
S: Optinfo=VERSION:2.1
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCAP
```

```
S: VERSION:2.1
S: METHOD:RESPONSE
S: TARGET:cap://cal.example.com/opaqueid101
S: CMDID:xyz12346
S: REQUEST-STATUS:4.1 ; access denied
S: END:VCAP
S:
S: ----FEE3790DC7E35189CA67CE2C
S: .
```

8.2.3 Timeouts

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

```
C: SENDDATA 3
C: Content-type:text/calendar; Method=READ; Component=VQUERY
C:
C: BEGIN:VCAP
C: VERSION:2.1
C: BEGIN:VCOMMAND
C: METHOD:READ
C: CMDID:xyz12346
C: TARGET:cap://cal.example.com/opaqueid101
C: TARGET:opaqueid103
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:VCOMMAND
C: END:VCALENDAR
C: .
S: 7.0 ; xyz12346
S: .
S: .
```

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

```
C: CONTINUE 10 ; xyz12346
```

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

```
C: ABORT ; xyz12346
```

```
S: 2.0
S: .
S: .
```

8.2.4 Using Parent and Children Properties

8.2.5 Query VEVENT.DTSTART and VALARM.DTSTART

9. 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?]

10. 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].]

10.1 Calendar Store Properties

The following are properties of the calendar store.

Name	Read Only	Value Type	Description
CALMASTER	N	URI	The email 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 or virtual host name.
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.

10.2 Calendar Properties

Name	Read Only	Value Type	Description
ALLOW-CONFLICTS	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_VCAR	N	TEXT	The default VCARs for newly Created top level calendars. This is a CARID. The default value is the value of DEFAULT_VCAR 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.
OWNERS	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 calendars 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 name 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 GMT.

11. Security Considerations

For the mandatory SASL mechanism that CAP specifies, the mechanism support is:

MUST authentication

MUST authorization

MAY impersonation

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], as defined in [VCARD].

[EDITORS NOTE (DR): RFC2426 - is vCard. This needs more explanation. What does vCARD have todo with this?]

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
Netscape, iPlanet
501 E Middlfield Road
Mountain View, CA 94043
US

Phone: +1-408-276-4268
EMail: sman@netscape.com

Doug Royer

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. Acknowledegements

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

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

Appendix B. Bibliography

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

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.

[VCARD] RFC.... TODO

Full Copyright Statement

Acknowledgement

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