Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 16, 2014

C. Daboo
Apple
A. Quillaud
Oracle

September 12, 2013

CalDAV Managed Attachments
draft-daboo-caldav-attachments-02

Abstract

This specification defines an extension to the calendar access protocol (CalDAV) to allow attachments associated with iCalendar data, to be stored and managed on the server.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 16, 2014.

Table of Contents

1. Introduction

The iCalendar [RFC5545] data format is used to represent calendar data and is used with iTIP [RFC5546] to handle scheduling operations between calendar users.

[RFC4791] defines the CalDAV Calendar Access protocol, based on HTTP [RFC2616], for accessing calendar data stored on a server.

Calendar users often want to include attachments with their calendar data events or tasks (for example a copy of a presentation, or the meeting agenda). iCalendar provides an "ATTACH" property whose value is either the inline Base64 encoded attachment data, or a URL specifying the location of the attachment data.

Use of inline attachment data is not ideal with CalDAV because the data would need to be uploaded to the server each time a change to the calendar data is done - even minor changes such as a change to the summary. Whilst a client could choose to use a URL value instead, the problem then becomes where and how the client discovers an appropriate URL to use and how it ensures that only those attendees listed in the event or task are able to access it.

This specification solves this problem by having the client send the attachment to the server, separately from the iCalendar data, and the server takes care of adding appropriate "ATTACH" properties in the iCalendar data as well as managing access privileges . The server can also provide additional information to the client about each attachment in the iCalendar data, such as the size and an identifier.

2. Conventions Used in This Document

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

The notation used in this memo is the ABNF notation of [RFC5234] as used by iCalendar [RFC5545]. Any syntax elements shown below that are not explicitly defined in this specification come from iCalendar [RFC5545].

3. Open Issues (to be removed prior to publication as a RFC)

1. Should we say anything about servers automatically stripping inline attachments and making them managed?

4. Overview

There are four main operations a client needs to do with attachments for calendar data: add, update, remove, and retrieve. The first three operations are carried out by the client issuing an HTTP POST request on the calendar object resource to which the attachment is associated and specifying the appropriate "action" query parameter. In the case of the remove operation, the client can alternatively directly update the calendar object resource and remove the relevant "ATTACH" properties. The retrieve operation is accomplished by simply issuing an HTTP GET request targeting the attachment URI specified by the calendar resource's "ATTACH" property.

iCalendar data stored in a CalDAV calendar object resource can contain multiple components when recurrences are involved. In such a situation, the client needs to be able to target a specific recurrence instance or multiple instances when adding or deleting attachments. As a result, the POST request needs to provide a way for the client to specify which recurrence instances should be targeted for the attachment operation. This is accomplished through use of additional query parameters on the POST request-URI.

4.1. Requirements

A server that supports the features described in this specification is REQUIRED to support the CalDAV "calendar-access" [RFC4791] features.

In addition, such a server MUST support the "return-representation" Prefer header value [I-D.snell-http-prefer] on successful HTTP PUT and POST requests targeting existing calendar object resources, by returning the new representation of that calendar resource (including its new Etag header value) in the response.

4.2. Discovering Support for Managed Attachments

A server supporting the features described in this specification MUST include "calendar-managed-attachments" as a field in the DAV response header from an OPTIONS request on a calendar home collection.

4.3. POST Request for Managing Attachments

An HTTP POST request is used to add, update, or remove attachments. The request-URI will contain various query parameters to specify the

behavior.

4.3.1. action= Query Parameter

The "action" query parameter is used to identify which attachment operation the client is requesting. This parameter MUST be present once on each POST request used to manage attachments. One of these three values MUST be used:

attachment-add Indicates an operation that is adding an attachment to a calendar object resource. See Section 4.4 for more details.

attachment-update Indicates an operation that is updating an existing attachment on a calendar object resource. See Section 4.5 for more details.

attachment-remove Indicates an operation that is removing an attachment on a calendar object resource. See Section 4.6 for more details.

Example:
http://calendar.example.com/events/1.ics?action=attachment-add

4.3.2. rid= Query Parameter

The "rid" query parameter is used to identify which recurrence instances are being targeted by the client for the attachment operation. This query parameter MUST contain one or more items, separated by commas (0x2C). The item values can be in one of two forms:

Master instance The value "M" (case-insensitive) refers to the "master" recurrence instance, i.e., the component that does not include a "RECURRENCE-ID" property. This item MUST be present only once.

Specific instance A specific iCalendar instance is targeted by using its "RECURRENCE-ID" value as the item value. That value MUST correspond to the RECURRENCE-ID value as stored in the calendar object resource (i.e. without any conversion to UTC). If multiple items of this form are used, they MUST be unique values.

If the "rid" query parameter is not present, all recurrence instances in the calendar object resource are targeted.

The "rid" query parameter MUST NOT be present in the case of an update operation.

Example:
http://calendar.example.com/events/1.ics?
action=attachment-add&rid=M,20111022T160000

4.3.3. managed-id= Query Parameter

The "managed-id" query parameter is used to identify which "ATTACH" property is being updated or removed. The value of this query parameter MUST match the "MANAGED-ID" property parameter value on the "ATTACH" property in the calendar object resource instance(s) targeted by the request.

The "managed-id" query parameter MUST NOT be present in the case of an add operation.

Example:
http://calendar.example.com/events/1.ics?
action=attachment-update&managed-id=aUNhbGVuZGFy

4.4. Adding attachments

To add an attachment to an existing calendar object resource, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.

   A. The request-URI will include an "action" query parameter with the value "attachment-add" (see Section 4.3.1).

   B. If all recurrence instances are having an attachment added, the "rid" query parameter is not present in the request-URI. If one or more specific recurrence instances are targeted, then the request-URI will include a "rid" query parameter containing the list of instances (see Section 4.3.2).

   C. The body of the request contains the data for the attachment.

   D. The client MUST include a valid Content-Type HTTP header describing the media type of the attachment (as required by HTTP).

   E. The client SHOULD include a Content-Disposition HTTP header [RFC6266] with a "type" parameter set to "attachment", and a "filename" parameter that indicates the name of the attachment.

2. When the server receives the POST request it does the following:

   A. Validates that any recurrence instances referred to via the "rid" query parameter are valid for the calendar object resource being targeted.

   B. Stores the supplied attachment data into a resource and generates an appropriate URI for clients to access the resource.

   C. For each affected recurrence instance in the calendar object resource targeted by the request, the server adds an "ATTACH" property, whose value is the URI of the stored attachment. The "ATTACH" property MUST contain a "MANAGED-ID" parameter whose value is a unique identifier (within the context of the server as a whole). The "ATTACH" property SHOULD contain an "FMTTYPE" parameter whose value matches the Content-Type header value from the request. The "ATTACH" property SHOULD contain an "FILENAME" parameter whose value matches the Content-Disposition header "filename" parameter value from the request, taking into account the restrictions expressed in Section 5.2. The "ATTACH" property SHOULD include a "SIZE" parameter whose value represents the size in octets of the attachment. If a specified recurrence instance does not have a matching component in the calendar object resource, then the server MUST modify the calendar object resource to include the overridden component with the appropriate "RECURRENCE-ID" property included.

   D. Upon successful creation of the attachment resource, and modification of the targeted calendar object resource, the server MUST return an appropriate HTTP success status response and include a "Cal-Managed-ID" HTTP header containing the "MANAGED-ID" parameter value of the newly created "ATTACH" property. The client can use the "Cal-Managed-ID" header value to correlate the attachment with "ATTACH" properties added to the calendar object resource. It is expected that the client will immediately reload the calendar object resource to refresh any local cache, or use the Prefer header "return-representation" option [I-D.snell-http-prefer] to have the server return the modified calendar object resource data in the HTTP response.

In the following example, the client adds a new attachment to a non recurring event and asks the server (via the Prefer [I-D.snell-http-prefer] HTTP header) to return the updated version of that event in the response.

```
>> Request <<


POST /events/64.ics?action=attachment-add HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition:attachment;filename=agenda.html
Content-Length: xxxx
Prefer: return-representation

<html>
  <body>
    <h1>Agenda</h1>
  </body>
</html>



>> Response <<


HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: http://cal.example.com/events/64.ics
ETag: "123456789-000-111"
Cal-Managed-ID: 97S

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VEVENT
UID:20010712T182145Z-123401@example.com
DTSTAMP:20120201T203412Z
DTSTART:20120714T170000Z
DTEND:20120715T040000Z
SUMMARY:One-off meeting
ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;
 FILENAME=agenda.html:https://cal.example.com/attach/64/34X22R
END:VEVENT
END:VCALENDAR
```

4.5. Updating Attachments

When an attachment is updated the server MUST change the associated "MANGED-ID" parameter and MAY change the "ATTACH" property value. With this approach, clients are able to determine when an attachment

has been updated by some other client by looking for a change to either the "ATTACH" property value, or the "MANAGED-ID" parameter value.

To change the data of an existing managed attachment in a calendar object resource, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.

   A. The request-URI will include an "action" query parameter with the value "attachment-update" (see Section 4.3.1).

   B. The request-URI will include a "managed-id" query parameter with the value matching that of the "MANAGED-ID" parameter for the "ATTACH" property being updated (see Section 4.3.3).

   C. The body of the request contains the updated data for the attachment.

   D. The client MUST include a valid Content-Type header describing the media type of the attachment (as required by HTTP).

   E. The client SHOULD include a Content-Disposition header [RFC6266] with a "type" parameter set to "attachment", and a "filename" parameter that indicates the name of the attachment.

2. When the server receives the POST request it does the following:

   A. Validates that the "managed-id" query parameter is valid for the calendar object resource.

   B. Updates the content of the attachment resource corresponding to that managed-id with the supplied attachment data.

   C. For each affected recurrence instance in the calendar object resource targeted by the request, the server updates the "ATTACH" property whose "MANAGED-ID" property parameter value matches the "managed-id" query parameter. The "MANAGED_ID" parameter value is changed to allow other clients to detect the update, and the property value (attachment URI) might also be changed. The "ATTACH" property SHOULD contain a "FMTTYPE" parameter whose value matches the Content-Type header value from the request - this could differ from the original value if the media type of the updated attachment is different. The "ATTACH" property SHOULD contain a "FILENAME"

parameter whose value matches the Content-Disposition header "filename" parameter value from the request, taking into account the restrictions expressed in Section 5.2. The "ATTACH" property SHOULD include a "SIZE" parameter whose value represents the size in octets of the updated attachment.

D. Upon successful update of the attachment resource, and modification of the targeted calendar object resource, the server MUST return an appropriate HTTP success status response, and include a "Cal-Managed-ID" HTTP header containing the new value of the "MANAGED-ID" parameter. It is expected that the client will immediately reload the calendar object resource to refresh any local cache, or use the Prefer header "return-representation" option [I-D.snell-http-prefer] to have the server return the modified calendar object resource data in the HTTP response.

The update operation does not take a "rid" parameter and does not add, or remove, any "ATTACH" property in the targetted calendar object resource. To link an existing attachment to a new instance, the client simply does a PUT on the calendar object resource, adding an "ATTACH" property which duplicates the existing one (see Section 4.7).

In the following example, the client updates an existing attachment and asks the server (via the Prefer [I-D.snell-http-prefer] HTTP header) to return the updated version of that event in the response.

>> Request <<

```
POST /events/64.ics?action=attachment-update&managed-id=97S HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition:attachment;filename=agenda.html
Content-Length: xxxx
Prefer: return-representation

<html>
  <body>
    <h1>Agenda</h1>
    <p>Discuss attachment draft</p>
  </body>
</html>
```

```
>> Response <<


HTTP/1.1 200 OK
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyz
Content-Location: http://cal.example.com/events/64.ics
Cal-Managed-ID: 98S
ETag: "123456789-000-222"

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VEVENT
UID:20010712T182145Z-123401@example.com
DTSTAMP:20120201T203412Z
DTSTART:20120714T170000Z
DTEND:20120715T040000Z
SUMMARY:One-off meeting
ATTACH;MANAGED-ID=98S;FMTTYPE=text/html;SIZE=xxxy;
 FILENAME=agenda.html:https://cal.example.com/attach/64/34X22R
END:VEVENT
END:VCALENDAR
```

4.6. Removing Attachments via POST

To remove an existing attachment from a calendar object, the following occurs:

1. The client issues a POST request targeted at the calendar object resource.

    A. The request-URI will include an "action" query parameter with the value "attachment-remove" (see Section 4.3.1).

    B. If all recurrence instances are having an attachment removed, the "rid" query parameter is not present in the request-URI. If one or more specific recurrence instances are targeted, then the request-URI will include a "rid" query parameter containing the list of instances (see Section 4.3.2).

    C. The request-URI will include a "managed-id" query parameter with the value matching that of the "MANAGED-ID" property parameter for the "ATTACH" property being removed (see Section 4.3.3).

D. The body of the request will be empty.

2. When the server receives the POST request it does the following:

A. Validates that any recurrence instances referred to via the "rid" query parameter are valid for the calendar object resource being targeted.

B. Validates that the "managed-id" query parameter is valid for the calendar object resource and specific instances being targeted.

C. For each affected recurrence instance in the calendar object resource targeted by the request, the server removes the matching "ATTACH" property. Note that if a specified recurrence instance does not have a matching component in the calendar object resource, then the server MUST modify the calendar object resource to include the overridden component with the appropriate "RECURRENCE-ID" property included, and the matching "ATTACH" property removed. This later case is actually valid only if the master component does include the referenced "ATTACH" property.

D. If the attachment resource is no longer referenced by any instance of the calendar object resource, the server can delete the attachment resource to free up storage space.

E. Upon successful removal of the attachment resource and modification of the targeted calendar object resource, the server MUST return an appropriate HTTP success status response. It is expected that the client will immediately reload the calendar object resource to refresh any local cache, or use the Prefer header "return-representation" option [I-D.snell-http-prefer] to have the server return the modified calendar object resource data in the HTTP response.

In the following example, the client deletes an existing attachment by passing its managed-id in the request. The Prefer [I-D.snell-http-prefer] HTTP header is not set in the request so the calendar object resource data is not returned in the response.

>> Request <<

```
POST /events/64.ics?action=attachment-remove&managed-id=98S HTTP/1.1
Host: cal.example.com
Content-Length: 0
```

```
>> Response <<


HTTP/1.1 200 No Content
Content-Length: 0
```

4.7. Adding Existing Managed Attachments via PUT

Clients can make use of existing managed attachments by adding the corresponding "ATTACH" property to calendar object resources (subject to the restrictions described in Section 4.11.3). When this occurs, servers SHOULD NOT change either the "MANAGED-ID" parameter value or the "ATTACH" property value for any managed attachments - this ensures that clients do not have to download the attachment data again if they already have it cached, because it is used in another calendar resource.

4.8. Updating Attachments via PUT

Servers MUST NOT allow clients to update attachment data directly via a PUT on the attachment URI (or via any other HTTP method that modifies content). Instead, attachments can only be manipulated via use of POST requests on the calendar data.

4.9. Removing Attachments via PUT

Clients can remove attachments by simply re-writing the calendar object resource data to remove the appropriate "ATTACH" properties.

4.10. Retrieving Attachments

Clients retrieve attachments by issuing an HTTP GET request using the value of the corresponding "ATTACH" property as the request-URI, taking into account the substitution mechanism associated with the "CALDAV:managed-attachments-server-URL" property (see Section 7.1).

4.11. Additional Considerations

4.11.1. Error Handling

This specification creates additional preconditions for the POST method.

The new preconditions are:

(CALDAV:max-attachment-size): The attachment submitted in the POST request MUST have an octet size less than or equal to the value of the CALDAV:max-attachment-size property value (Section 7.2) on the calendar collection of the target calendar resource;

(CALDAV:max-attachments-per-resource): The addition of the attachment submitted in the POST request MUST result in the target calendar resource having a number of managed attachments less than or equal to the value of the CALDAV:max-attachments-per-resource property value (Section 7.3) on the calendar collection of the target calendar resource;

A POST request to add, modify, or delete a managed attachment results in an implicit modification of the targeted calendar resource (equivalent of a PUT). As a consequence, clients should also be prepared to handle preconditions associated with this implicit PUT. This includes (but is not limited to):

(CALDAV:max-resource-size) (from Section 5.3.2.1 of [RFC4791])

(DAV:quota-not-exceeded) (from Section 6 of [RFC4331])

(DAV:sufficient-disk-space) (from Section 6 of [RFC4331])

4.11.2. Quotas

The WebDAV Quotas [RFC4331] specification defines two live WebDAV properties (DAV:quota-available-bytes and DAV:quota-used-bytes) to communicate storage quota information to clients. Server implementations MAY choose to include managed attachments sizes when calculating the amount of storage used by a particular resource.

4.11.3. Access Control

TODO: talk about authentication for attachment URIs, access limited to attendees only, or token based, only organizer able to re-use "ATTACH" properties.

4.11.4. Redirects

For POST requests that add or update attachment data, the server MAY issue an HTTP redirect to require the client to re-issue the POST request using a different request-URI. As a result, it is always best for clients to use the "100 Continue" behavior defined in Section 8.2.3 of [RFC2616]. Using this mechanism ensures that, if a redirect does occur, the client does not needlessly send the attachment data.

When servers do redirect clients to a different request-URI, they SHOULD ensure that the authentication/authorization for the redirected URI falls within the same domain realm as that of the original URI, so that the client can re-use credentials without having to prompt the user.

4.11.5. Automatic Clean-up by servers

Servers MAY automatically remove attachment data, for example to regain the storage taken by unused attachments, or as the result of a virus scanning. When doing so they SHOULD NOT modify calendar data referencing those attachments. Instead they SHOULD return a 410 HTTP status response to any request on the removed attachment URI.

4.11.6. Sending Scheduling Messages with Attachments

When a managed attachment is added, updated or removed from a calendar object resource, the server MUST ensure that a scheduling message is sent to update any Attendees with the changes, as per [RFC6638].

4.11.7. Attendees managing Attachments

TODO: how do we do this? Organizer adds a property to calendar data to say attendees can add attachments, but how does scheduling happen?

4.11.8. Other Client Considerations

Clients can expect servers to take a while to respond to POST requests that include large attachment bodies. Servers SHOULD use the "100 Continue" behavior defined in Section 8.2.3 of [RFC2616] to keep the client connection alive if the response will take some time.

When exporting calendar data from a CalDAV server supporting managed attachments, clients SHOULD remove all "MANAGED-ID" property parameters from "ATTACH" properties in the calendar data. Similarly when importing calendar data from another source, clients SHOULD remove any "MANAGED-ID" property parameters on "ATTACH" properties (failure to do so will likely result in the server removing those properties automatically).

5. Modifications to iCalendar Syntax

5.1. SIZE Property Parameter

Parameter Name: SIZE

Purpose: To specify the size of an attachment.

Format Definition: This property parameter is defined by the following notation:

```
sizeparam = "SIZE" "=" paramtext
; positive integers
```

Description: This property parameter MAY be specified on "ATTACH" properties. It indicates the size in octets of the corresponding attachment data. Since iCalendar integer values are restricted to a maximum value of 2147483647, the current parameter is defined as text to allow an extended range to be used.

Example:

```
ATTACH;SIZE=1234:http://attachments.example.com/abcd.txt
```

5.2. FILENAME Property Parameter

Parameter Name: FILENAME

Purpose: To specify the file name of a managed attachment.

Format Definition: This property parameter is defined by the following notation:

```
filenameparam = "FILENAME" "=" paramtext
```

Description: This property parameter MAY be specified on "ATTACH" properties corresponding to managed attachments. Its value provides information on how to construct a filename for storing the attachment data. This parameter is very similar in nature to the Content-Disposition HTTP header "filename" parameter and exposes the same security risks. As a consequence, clients MUST follow the guidelines expressed in Section 4.3 of [RFC6266] when consuming this parameter value. Similarly, servers MUST follow those same guidelines before storing a value.

Example:

```
ATTACH;FILENAME=agenda.html:http://attachments.example.c
 om/rt452S
```

5.3. MANAGED-ID Property Parameter

Parameter Name: MANAGED-ID

Purpose: To uniquely identify a managed attachment.

Format Definition: This property parameter is defined by the following notation:

managedidparam = "MANAGED-ID" "=" paramtext

Description: This property parameter MUST be specified on "ATTACH" properties corresponding to managed attachments. Its value is generated by the server and uniquely identifies a managed attachment. This property parameter MUST NOT be present in the case of non managed attachments.

Example:

```
ATTACH;MANAGED-ID=aUNhbGVuZGFy:http://attachments.example.c
 om/abcd.txt
```

6. Additional Message Header Fields

6.1. Cal-Managed-ID Response Header

The Cal-Managed-ID response header provides the value of the MANAGED-ID parameter corresponding to a newly added ATTACH property. It MUST be sent only in response to a successful POST request with an action set to attachment-add.

```
Cal-Managed-ID = "Cal-Managed-ID" ":" paramtext
; "paramtext" is defined in Section 3.1 of [RFC5545]
```

Example:

```
Cal-Managed-ID:aUNhbGVuZGFy
```

7. Additional WebDAV properties

7.1. CALDAV:managed-attachments-server-URL property

Name: managed-attachments-server-URL

Namespace: urn:ietf:params:xml:ns:caldav

Purpose: Specifies the server base URI to use when retrieving managed attachments.

Protected: This property MUST be protected as only the server can update the value.

COPY/MOVE behavior: This property is only defined on a calendar home collection which cannot be moved or copied.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: This property MAY be defined on a calendar home collection. If present, it contains zero or one DAV:href XML elements.

When one DAV:href element is present, its value MUST be an absolute HTTP URI containing only the scheme (i.e. "https") and authority (i.e. host and port) parts . Whenever a managed attachment is to be retrieved via an HTTP GET, the client MUST construct the actual URL of the attachment by substituting the scheme and authority parts of the attachment URI (as stored in the iCalendar "ATTACH" property) with the present WebDAV property value.

When no DAV:href element is present, the client MUST substitute the scheme and authority parts of the attachment URI with the scheme and authority part of the calendar home collection absolute URI.

In the absence of this property, the client can consider the attachment URI as its actual URL.

Definition:

```
<!ELEMENT managed-attachments-server-URL (DAV:href?)>
```

Example:

```
<C:managed-attachments-server-URL xmlns:D="DAV:"
    xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:href>https://attachstore.example.com</D:href>
</C:managed-attachments-server-URL>
```

7.2. CALDAV:max-attachment-size property

Name: max-attachment-size

Namespace: urn:ietf:params:xml:ns:caldav

Purpose: Provides a numeric value indicating the maximum attachment size, in octets, that the server is willing to accept when a managed attachment is stored on the server.

Protected: MUST be protected as it indicates limits provided by the server.

COPY/MOVE behavior: This property value MUST be preserved in COPY and MOVE operations.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: The CALDAV:max-attachment-size property is used to specify a numeric value that represents the maximum attachment size, in octets, that the server is willing to accept when a managed attachment is stored on the server. The property is defined on the parent collection of the calendar object resource to which the attachment is associated. Any attempt to store a managed attachment exceeding this size MUST result in an error, with the CALDAV:max-attachment-size precondition (Section 4.11.1) being violated. In the absence of this property, the client can assume that the server will allow storing an attachment of any reasonable size.

Definition:

```
<!ELEMENT max-attachment-size (#PCDATA)>
<!-- PCDATA value: a numeric value (positive decimal integer) -->
```

Example:

```
<C:max-attachment-size xmlns:C="urn:ietf:params:xml:ns:caldav"
     >102400000</C:max-attachment-size>
```

7.3. CALDAV:max-attachments-per-resource property

Name: max-attachments-per-resource

Namespace: urn:ietf:params:xml:ns:caldav

Purpose: Provides a numeric value indicating the maximum number of managed attachments across all instances of a calendar object resource stored in a calendar collection.

Protected: MUST be protected as it indicates limits provided by the server.

COPY/MOVE behavior: This property value MUST be preserved in COPY and MOVE operations.

allprop behavior: SHOULD NOT be returned by a PROPFIND DAV:allprop request.

Description: The CALDAV:max-attachments-per-resource property is used to specify a numeric value that represents the maximum number of managed attachments across all instances of a calendar object resource stored in a calendar collection. Non managed attachments are not counted toward that limit. The property is defined on the parent collection of the calendar object resource to which the attachment is associated. Any attempt to add a managed attachment that would cause the calendar resource to exceed this limit MUST result in an error, with the CALDAV:max-attachments-per-resource precondition (Section 4.11.1) being violated. In the absence of this property, the client can assume that the server can handle any number of managed attachments per calendar resource.

Definition:

```
<!ELEMENT max-attachments-per-resource (#PCDATA)>
<!-- PCDATA value: a numeric value (positive decimal integer) -->
```

Example:

```
<C:max-attachments-per-resource
    xmlns:C="urn:ietf:params:xml:ns:caldav"
     >12</C:max-attachments-per-resource>
```

8. Security Considerations

TODO:servers need to enforce quotas, attachments could contain malicious content (virus checking could be done)

9. IANA Considerations

9.1. Parameter Registrations

This specification defines the following new iCalendar property parameters to be added to the registry defined in Section 8.2.3 of [RFC5545]:

Property Parameter	Status	Reference
FILENAME	Current	RFCXXXX, Section 5.2
SIZE	Current	RFCXXXX, Section 5.1
MANAGED-ID	Current	RFCXXXX, Section 5.3

9.2. Message Header Field Registrations

The message header fields below should be added to the Permanent Message Header Field Registry (see [RFC3864]).

9.2.1. Cal-Managed-ID

Header field name: Cal-Managed-ID

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): this specification (Section 6.1)

Related information: none

10. Acknowledgments

This specification came about via discussions at the Calendaring and Scheduling Consortium. Thanks in particular to Mike Douglass, Ken Murchison, and Eric York.

11. References

11.1. Normative References

[I-D.snell-http-prefer]
Snell, J., "Prefer Header for HTTP", draft-snell-http-prefer-18 (work in progress), January 2013.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC4331] Korver, B. and L. Dusseault, "Quota and Size Properties fo r Distributed Authoring and Versioning (DAV) Collections", RFC 4331, February 2006.

[RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, March 2007.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

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

[RFC6266] Reschke, J., "Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)", RFC 6266, June 2011.

[RFC6638] Daboo, C. and B. Desruisseaux, "Scheduling Extensions to CalDAV", RFC 6638, June 2012.

11.2. Informative References

[RFC5546] Daboo, C., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, December 2009.

Appendix A. Change History (To be removed by RFC Editor before publication)

Changes in -02:

1. MANAGED-ID changes on PUT.

2. MTAG has been removed.

3. Error pre-conditions added.

4. Interaction with WebDAV QUOTA discussed.

5. max-attachment-* limits added.

6. Updated references.

7. Removed MUST for specific 2xx codes in favor of generic success code.

Changes in -01:

1. Tweaked OPTIONS capability wording.

2. Added section on clients expecting 100-Continue for delayed response.

3. Added text for clean-up and use of HTTP 410 on orphans.

4. Added text on removing "MANAGED-ID" when exporting/importing calendar data.

5. Added protocol examples.

6. Added MTAG property parameter on ATTACH property

7. Added FILENAME property parameter on ATTACH property

8. "id" query parameter is now "managed-id".

9. Use of Cal-Managed-ID header instead of Location header in responses.

10. rid query param MUST contain RECURRENCE-ID without any conversion to UTC (case of floating events).

11. Introduced CALDAV:managed-attachments-server-URL property

12. Made support for Prefer header a MUST for servers.

Appendix B. Example Involving Recurring Events

In the following example, the organizer of a recurring meeting adds an agenda (HTML attachment) to the corresponding calendar resource. Attendees of the meeting are granted read access to the newly created attachment resource. Their own copy of the meeting is updated to include the new ATTACH property pointing to the attachment resource and they are notified of the change via their scheduling inbox.

>> Request <<

```
POST /events/65.ics?action=attachment-add HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition:attachment;filename=agenda.html
Content-Length: xxxx
Prefer: return-representation

<html>
  <body>
    <h1>Agenda</h1>
    <p>As usual</p>
  </body>
</html>
```

>> Response <<

```
   HTTP/1.1 201 Created
   Content-Type: text/calendar; charset="utf-8"
   Content-Length: yyyy
   Content-Location: http://cal.example.com/events/65.ics
   ETag: "123456789-000-111"
   Cal-Managed-ID: 97S

   BEGIN:VCALENDAR
   VERSION:2.0
   PRODID:-//Example Corp.//CalDAV Server//EN
   BEGIN:VTIMEZONE
   LAST-MODIFIED:20040110T032845Z
   TZID:America/Montreal
   BEGIN:DAYLIGHT
   DTSTART:20000404T020000
   RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
   TZNAME:EDT
   TZOFFSETFROM:-0500
   TZOFFSETTO:-0400
   END:DAYLIGHT
   BEGIN:STANDARD
   DTSTART:20001026T020000
   RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
   TZNAME:EST
   TZOFFSETFROM:-0400
   TZOFFSETTO:-0500
   END:STANDARD
   END:VTIMEZONE
   BEGIN:VEVENT
   UID:20010712T182145Z-123401@example.com
   DTSTAMP:20120201T203412Z
   DTSTART;TZID=America/Montreal:20120206T100000
   DURATION:PT1H
   RRULE:FREQ=WEEKLY
   SUMMARY:Planning Meeting
   ORGANIZER:mailto:cyrus@example.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@exampl
    e.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exam
    ple.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@exa
    mple.com
   ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;
    FILENAME=agenda.html:https://cal.example.com/attach/65/34X22R
   END:VEVENT
   END:VCALENDAR
```

The organizer has a more specific agenda for the 20th of February meeting. It is added to that particular instance of the meeting by specifying the rid parameter.

>> Request <<

```
POST /events/65.ics?action=attachment-add&rid=20120220T100000 HTTP/1.1
Host: cal.example.com
Content-Type: text/html; charset="utf-8"
Content-Disposition:attachment;filename=agenda0220.html
Content-Length: xxxx
Prefer: return-representation

<html>
  <body>
    <h1>Agenda</h1>
    <p>Something different, for a change</p>
  </body>
</html>
```

>> Response <<

```
HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: http://cal.example.com/events/65.ics
ETag: "123456789-000-222"
Cal-Managed-ID: 33225

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VTIMEZONE
LAST-MODIFIED:20040110T032845Z
TZID:America/Montreal
BEGIN:DAYLIGHT
DTSTART:20000404T020000
RRULE:FREQ=YEARLY;BYDAY=1SU;BYMONTH=4
TZNAME:EDT
TZOFFSETFROM:-0500
TZOFFSETTO:-0400
END:DAYLIGHT
BEGIN:STANDARD
DTSTART:20001026T020000
```

```
   RRULE:FREQ=YEARLY;BYDAY=-1SU;BYMONTH=10
   TZNAME:EST
   TZOFFSETFROM:-0400
   TZOFFSETTO:-0500
   END:STANDARD
   END:VTIMEZONE
   BEGIN:VEVENT
   UID:20010712T182145Z-123401@example.com
   DTSTAMP:20120201T203412Z
   DTSTART;TZID=America/Montreal:20120206T100000
   DURATION:PT1H
   RRULE:FREQ=WEEKLY
   SUMMARY:Planning Meeting
   ORGANIZER:mailto:cyrus@example.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@exampl
    e.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exam
    ple.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@exa
    mple.com
   ATTACH;MANAGED-ID=97S;FMTTYPE=text/html;SIZE=xxxx;
    FILENAME=agenda.html:https://cal.example.com/attach/65/34X22R
   END:VEVENT
   BEGIN:VEVENT
   UID:20010712T182145Z-123401@example.com
   RECURRENCE-ID;TZID=America/Montreal:20120220T100000
   DTSTAMP:20120201T203412Z
   DTSTART;TZID=America/Montreal:20120220T100000
   DURATION:PT1H
   SUMMARY:Planning Meeting
   ORGANIZER:mailto:cyrus@example.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:cyrus@example.
    com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:mailto:arnaudq@exampl
    e.com
   ATTENDEE;CUTYPE=INDIVIDUAL;PARTSTAT=NEEDS-ACTION:mailto:mike@examp
    le.com
   ATTACH;MANAGED-ID=33225;FMTTYPE=text/html;SIZE=xxxx;
    FILENAME=agenda0220.html:https://cal.example.com/attach/65/FGZ225
   END:VEVENT
   END:VCALENDAR
```

Authors' Addresses

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

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

Arnaud Quillaud
Oracle Corporation
180, Avenue de l'Europe
Saint Ismier cedex, 38334
France

Email: arnaud.quillaud@oracle.com
URI: http://www.oracle.com/