Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2012

C. Daboo
Apple

October 24, 2011

CalDAV Managed Attachments
draft-daboo-caldav-attachments-00

Abstract

This document defines how CalDAV servers can provide server managed collections 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 April 26, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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 provides a solution to this problem whereby the client sends 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 privileges . The server can also provide additional information to the client about each attachment, such as the size and an identifier, so clients can manage their list of attachments directly from the calendar data.

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. Since clients need to reload the calendar object resource after a POST operation, should we provide a way for the client to request the server to return the calendar data in the successful POST response?

2. Do we need a way to allow for multiple attachments to be added, updated or removed in a single POST (batching)?

3. Do we really need POST to remove given that a client PUT is pretty easy to do? Only benefit for POST is the request has no body.

4. SIZE uses iCalendar integer - but that is limited to 2^31. However attachments could exceed that size.

5. Do we need a REPORT to allow clients to detect the largest attachments affecting a user's quota? Should there be a request to purge all attachments older than some cut-off?

6. Do we need a way for the server to indicate limits for managed attachments (individual size, total size, total number etc). Also a way for a server to indicate it does not allow inline attachments?

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 for which the attachment is required. One exception to this is the delete operation, which additionally allows for attachment removal by the client simply updating the calendar object resource and removing any "ATTACH" properties. The retrieve operation is accomplished by simply issuing an HTTP GET request targeting the attachment URI.

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 updating 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 query parameters on the POST request-URI.

4.1. Discovering Support for Managed Attachments

Servers advertise support for this extension by including the token "calendar-managed-attachments" in the DAV response header to an OPTIONS request on a calendar home collection. Clients MUST check for the presence of that token before using any of the features defined by this specification.

4.2. 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.

4.2.1. action= Query Parameter

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

attachment-add Indicates an operation that is adding an attachment to a calendar object resource. See Section 4.3 for more details.

attachment-update Indicates an operation that is updating an existing attachment on a calendar object resource. See Section 4.4 for more details.

attachment-remove Indicates an operation that is removing an attachment on a calendar object resource. See Section 4.5 for more details.

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

4.2.2. rid= Query Parameter

The "rid" query parameter is used to identify which recurrence instance is being targeted by the client for the attachment operation. The 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 only be present once.

Specific instance A specific iCalendar instance is targeted by using its "RECURRENCE-ID" value as the query parameter value. 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.

Example:
http://calendar.example.com/events/1.ics?
action=attachment-add&rid=M,20111022T160000Z

4.2.3. id= Query Parameter

The "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.

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

4.3. 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 4.2.1).

   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 4.2.2).

   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" of "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" property parameter whose value is a unique identifier (within the context of the server as a whole). The "ATTACH" property SHOULD contain an "FMTTYPE" property parameter whose value matches the Content-Type header value from the request. The "ATTACH" property SHOULD include a "SIZE" property parameter whose value represents the size in octets of the attachment. 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.

   D. Upon successful creation of the attachment resource and modification of the targeted calendar object resource, the server MUST return a 200 HTTP status response and include a Location HTTP header containing the URI of the stored attachment. The client can use the Location 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.

4.4. Updating Attachments

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 4.2.1).

B. If all recurrence instances are having an attachment updated, 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 4.2.2).

C. The request-URI will include an "id" query parameter with the value matching that of the "MANAGED-ID" property parameter for the "ATTACH" property being updated (see Section 4.2.3).

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

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

F. The client SHOULD include a Content-Disposition header [RFC6266] with a "type" of "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. Validates that the "id" query parameter is valid for the calendar object resource and specific instances being targeted.

C. Stores the supplied attachment data into a resource and generates an appropriate URI for clients to access the resource. The URI for the updated attachment data MUST be different from the one for the attachment being updated. This allows clients to use the URI value as a "change token".

D. For each affected recurrence instance in the calendar object resource targeted by the request, the server updates the "ATTACH" property, whose value is the URI of the updated attachment. The "MANAGED-ID" property parameter value MUST remain the same as it was for the attachment being updated. The "ATTACH" property SHOULD contain an "FMTTYPE" property 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 include a "SIZE" property parameter whose value represents the size in octets of the

updated attachment. 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.

E. Upon successful update of the attachment resource and modification of the targeted calendar object resource, the server MUST return a 200 HTTP status response and include a Location header containing the URI of the updated attachment. It is expected that the client will immediately reload the calendar object resource to refresh any local cache.

4.5. Removing Attachments via POST

To remove an attachment on 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-remove" (see Section 4.2.1).

   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 4.2.2).

   C. The request-URI will include an "id" query parameter with the value matching that of the "MANAGED-ID" property parameter for the "ATTACH" property being removed (see Section 4.2.3).

   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 "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.

D. Upon successful removal of the attachment resource and modification of the targeted calendar object resource, the server MUST return a 200 HTTP status response. It is expected that the client will immediately reload the calendar object resource to refresh any local cache.

4.6. 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 4.10.2).

4.7. 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).

4.8. Removing Attachments via PUT

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

4.9. Retrieving Attachments

Clients retrieve attachments by simply issuing an HTTP GET request using the value of the corresponding "ATTACH" property as the request-URI.

4.10. Additional Considerations

4.10.1. Error Handling

TODO: error codes for each request.

4.10.2. 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.

4.10.3. Server 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 define in Section 8.2.3 of [RFC2616]. That ensures that, if a redirect does occur, the client does not needless 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 as that of the original URI, so that the client can re-use credentials without having to prompt the user.

4.10.4. Server Clean-up

TODO: describe reference counting of managed attachments - server can remove actual attachment data once there are no more references to the attachment URI. Also, servers can purge old attachment data - return a 404 or 410.

4.10.5. 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 [I-D.desruisseaux-caldav-sched].

4.10.6. 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?

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" "=" integer
; positive integers or -1
```

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, if the attachment size exceeds that, the value -1 MUST instead be used.

Example:

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

5.2. 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 MAY be specified on "ATTACH" properties. Its value is generated by the server and uniquely identifies a managed attachment.

Example:

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

6. Security Considerations

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

7. IANA Considerations

7.1. Paramater Registrations

This documents defines the following new iCalendar properties to be added to the registry defined in Section 8.2.3 of [RFC5545]:

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

8. Acknowledgments

This specification came about via discussions at the Calendaring and Scheduling Consortium.

9. References

9.1. Normative References

[I-D.desruisseaux-caldav-sched]
Daboo, C. and B. Desruisseaux, "CalDAV Scheduling Extensions to WebDAV", draft-desruisseaux-caldav-sched-10 (work in progress), September 2011.

[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.

[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.

9.2. Informative References

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

Author's Address

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

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