CalConnect TC CALENDAR

CalDAV Managed Attachments

Working Draft Standard

Warning for drafts

This document is not a CalConnect Standard. It is distributed for review and comment, and is subject to change without notice and may not be referred to as a Standard. Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

The Calendaring and Scheduling Consortium, Inc. 2019

The Calendaring and Scheduling Consortium, Inc.

4390 Chaffin Lane
McKinleyville
California 95519
United States of America

copyright@calconnect.org
www.calconnect.org

Contents

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.

This specification documents existing code deployed by multiple vendors. It is published as an Informational specification rather than Standards Track due to its noncompliance with multiple best current practices of HTTP.

Introduction

The iCalendar IETF RFC 5545 data format is used to represent calendar data and is used with iCalendar Transport-independent Interoperability Protocol (iTIP) IETF RFC 5546 to handle scheduling operations between calendar users.

IETF RFC 4791 defines the Calendaring Extensions to WebDAV (CalDAV), based on HTTP IETF RFC 7230, 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.

Rationale for Informational Status

Although this extension to CalDAV has wide deployment, its design does not comply with some of the best current practices of HTTP, namely:

— All operations on attachments are modeled as HTTP POST operations, where the actual type of operation is specified using a query parameter, instead of using separate HTTP POST, PUT, and DELETE methods where appropriate.
— Specific query strings are hardwired into the protocol in violation of IETF RFC 7320, Section 2.4

Additionally, this extension misuses the Content-Disposition header field IETF RFC 6266 as a request header field to convey a filename for an attachment rather than using an existing request header field suitable for that purpose, such as "Slug" (see IETF RFC 5023, Section 9.7).

Rather than creating interoperability problems with deployed code by updating the design of this extension to be compliant with best current practices and to allow this specification to be placed on the Standards Track, it was decided to simply document how existing implementations interoperate and to publish the document as Informational.

CalDAV Managed Attachments

1. Scope

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

2. Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF RFC 2119, BRADNER, S. *Key words for use in RFCs to Indicate Requirement Levels*. 1997. RFC Publisher. https://www.rfc-editor.org/info/rfc2119.

IETF RFC 2518, GOLAND, Y., E. WHITEHEAD, A. FAIZI, S. CARTER, D. JENSEN and Internet Engineering Task Force. *HTTP Extensions for Distributed Authoring — WEBDAV*. 1999. RFC Publisher. https://www.rfc-editor.org/info/rfc2518.

IETF RFC 3864, KLYNE, G., M. NOTTINGHAM and J. MOGUL. *Registration Procedures for Message Header Fields*. 2004. RFC Publisher. https://www.rfc-editor.org/info/rfc3864.

IETF RFC 4331, KORVER, B., L. DUSSEAULT and Internet Engineering Task Force. *Quota and Size Properties for Distributed Authoring and Versioning (DAV) Collections*. 2006. RFC Publisher. https://www.rfc-editor.org/info/rfc4331.

IETF RFC 4791, DABOO, C., B. DESRUISSEAUX and L. DUSSEAULT. *Calendaring Extensions to WebDAV (CalDAV)*. 2007. RFC Publisher. https://www.rfc-editor.org/info/rfc4791.

IETF RFC 4918, Internet Engineering Task Force (committee). *HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)*. 2007. RFC Publisher. https://www.rfc-editor.org/info/rfc4918.

IETF RFC 5234, OVERELL, P. *Augmented BNF for Syntax Specifications: ABNF*. 2008. RFC Publisher. https://www.rfc-editor.org/info/rfc5234.

IETF RFC 5545, Internet Engineering Task Force (committee). *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. 2009. RFC Publisher. https://www.rfc-editor.org/info/rfc5545.

IETF RFC 6266, RESCHKE, J. and Internet Engineering Task Force. *Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP)*. 2011. RFC Publisher. https://www.rfc-editor.org/info/rfc6266.

IETF RFC 6638, DABOO, C. and B. DESRUISSEAUX. *Scheduling Extensions to CalDAV*. 2012. RFC Publisher. https://www.rfc-editor.org/info/rfc6638.

IETF RFC 7230, Internet Engineering Task Force (committee). *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. 2014. RFC Publisher. https://www.rfc-editor.org/info/rfc7230.

IETF RFC 7231, Internet Engineering Task Force (committee). *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. 2014. RFC Publisher. https://www.rfc-editor.org/info/rfc7231.

IETF RFC 7240, SNELL, J. *Prefer Header for HTTP*. 2014. RFC Publisher. https://www.rfc-editor.org/info/rfc7240.

IETF RFC 7538, RESCHKE, J. and Internet Engineering Task Force. *The Hypertext Transfer Protocol Status Code 308 (Permanent Redirect)*. 2015. RFC Publisher. https://www.rfc-editor.org/info/rfc7538.

IETF RFC 8174, LEIBA, B. *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*. 2017. RFC Publisher. https://www.rfc-editor.org/info/rfc8174.

3. Terms and definitions

No terms and definitions are listed in this document.

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 (see Clause 4.2, Paragraph 3). In the case of the remove operation, the client can alternatively directly update the calendar object resource and remove the relevant "ATTACH" properties (see Clause 4.8). The retrieve operation is accomplished by simply issuing an HTTP GET request targeting the attachment URI specified by the calendar resource's "ATTACH" property (see Clause 4.9).

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" IETF RFC 4791 features.

In addition, such a server SHOULD support the "return=representation" Prefer header field IETF RFC 7240 preference 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 field 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 token in the DAV response header field (as defined in IETF RFC 4918, Section 10.1) from an OPTIONS request on a calendar home collection.

A server might choose to not support storing managed attachments on a per-recurrence instance basis (i.e., they can only be added to all instances as a whole). If that is the case, the server MUST also include "calendar-managed-attachments-no-recurrence" as a token in the DAV response header field from an OPTIONS request on a calendar home collection. When that field is present, clients MUST NOT attempt any managed attachment operations that target specific recurrence instances.

=== POST Request for Managing Attachments

An HTTP POST request is used to add, update, or remove attachments. These requests are subject to the preconditions listed in Clause 4.10. The request-URI will contain various query parameters to specify the behavior.

==== 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 Clause 4.3 for more details.
`attachment-update`	Indicates an operation that is updating an existing attachment on a calendar object resource. See Clause 4.4 for more details.
`attachment-remove`	Indicates an operation that is removing an attachment from a calendar object resource. See Clause 4.5 for more details.
Example	https://calendar.example.com/events/1.ics?action=attachment-add

4.2.1. `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 (0×2C). 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. For example, to target a recurrence defined by property `RECURRENCE-ID;TZID=America/Montreal:20111022T160000`, the query parameter `rid=20111022T160000` would be used. 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, or if the server chooses not to support per-recurrence instance managed attachments (see Clause 4.1).
Example (targeting the master instance and a specific overridden instance)	https://calendar.example.com/events/1.ics?action=attachment-add&rid=M,20111022T160000

4.2.2. `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	https://calendar.example.com/events/1.ics?action=attachment-update&managed-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 Clause 4.2, Paragraph 5).
   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 Clause 4.2.1).
   c) The body of the request contains the data for the attachment.
   d) The client MUST include a valid Content-Type header field describing the media type of the attachment (as required by HTTP).
   e) The client SHOULD include a Content-Disposition header field IETF RFC 6266 with a "type" parameter set to "attachment", and a "filename" parameter that indicates the name of the attachment. Note that the use of Content-Disposition as a request header field is nonstandard and specific to this protocol.
   f) The client MAY include a Prefer header field IETF RFC 7240 with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.
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 field value from the request. The "ATTACH" property SHOULD contain an "FILENAME" parameter whose value matches the Content-Disposition header field "filename" parameter value from the request, taking into account the restrictions expressed in Clause 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 an overridden component with the appropriate "RECURRENCE-ID" property.
   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" header field containing the "MANAGED-ID" parameter value of the newly created "ATTACH" property. The client can use the "Cal-Managed-ID" header field value to correlate the attachment with "ATTACH" properties added to the calendar object resource. If the client included a Prefer header field with the "return=representation" preference in the request, the server SHOULD return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent GET request to refresh any local cache.

In the following example, the client adds a new attachment to a non recurring event and asks the server (via the Prefer IETF RFC 7240 header field) to return the modified 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: https://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
```

4.4. Adding attachments

When an attachment is updated the server MUST change the associated "MANAGED-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 Clause 4.2, Paragraph 5).
   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 Clause 4.2.2).
   c) The body of the request contains the updated data for the attachment.
   d) The client MUST include a valid Content-Type header field describing the media type of the attachment (as required by HTTP).
   e) The client SHOULD include a Content-Disposition header field IETF RFC 6266 with a "type" parameter set to "attachment", and a "filename" parameter that indicates the name of the attachment.
   f) The client MAY include a Prefer header field IETF RFC 7240 with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.
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 field 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 field "filename" parameter value from the request, taking into account the restrictions expressed in "FILENAME-parameter". 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" header field containing the new value of the "MANAGED-ID" parameter. The client can use the "Cal-Managed-ID" header field value to correlate the attachment with "ATTACH" properties added to the calendar object resource. If the client included a Prefer header field with the "return=representation" preference in the request, the server SHOULD return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent GET request to refresh any local cache.

The update operation does not take a "rid" parameter and does not add, or remove, any "ATTACH" property in the targeted 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 Clause 4.6).

In the following example, the client updates an existing attachment and asks the server (via the Prefer IETF RFC 7240 header field) 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: https://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
```

4.5. 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 Clause 4.2, Paragraph 5).
   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 Clause 4.2.1).
   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 Clause 4.2.2).
   d) The body of the request will be empty.
   e) The client MAY include a Prefer header field IETF RFC 7240 with the "return=representation" preference to request that the modified calendar object resource be returned as the body of a successful response to the POST request.
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 an overridden component with the appropriate "RECURRENCE-ID" property, 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. If the client included a Prefer header field with the "return=representation" preference in the request, the server SHOULD return the modified calendar object resource as the body of the response. Otherwise, the server can expect that the client will reload the calendar object resource with a subsequent GET request to refresh any local cache.

In the following example, the client deletes an existing attachment by passing its managed-id in the request. The Prefer IETF RFC 7240 header field 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/1.1 204 No Content
Content-Length: 0
```

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 Clause 4.11.2).

If a managed attachment is used in more than calendar resource, servers `SHOULD NOT` change either the "`MANAGED-ID" parameter value or the "ATTACH`" property value for these attachments — this ensures that clients do not have to download the attachment data again if they already have it cached. Additionally, servers SHOULD validate "SIZE" parameter values and replace incorrect values with the actual sizes of existing attachments.

These PUT requests are subject to the preconditions listed in Clause 4.10.

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). Instead, attachments can only be updated via use of POST requests on the calendar data.

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. Servers MUST NOT allow clients to delete attachments directly via a DELETE request on the attachment URI.

4.9. 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 Clause 7).

4.10. 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 (Clause 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 (Clause 7.3) on the calendar collection of the target calendar resource;
(CALDAV:valid-action)	The action query parameter in the POST request MUST contain one of "attachment-add", "attachment-update", or "attachment-remove".

(CALDAV:valid-rid)	The rid query parameter in the POST request MUST NOT` be present for an attachment-update action, and `MUST contain the value "M" and/or values corresponding to "RECURRENCE-ID" property values in the iCalendar data targeted by the request.
(CALDAV:valid-managed-id)	The managed-id query parameter in the POST request MUST NOT be present for an attachment-add action, and MUST contain a value corresponding to a "MANAGED-ID" property parameter value in the iCalendar data targeted by the request.

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 IETF RFC 4791, Section 5.3.2.1)
— (DAV:quota-not-exceeded) (from IETF RFC 4331, Section 6)</t>
— (DAV:sufficient-disk-space) (from IETF RFC 4331, Section 6)

A PUT request to add or modify and existing calendar object resource can make reference to an existing managed attachment. The following new preconditions is defined:

(CALDAV:valid-managed-id-parameter)	a "MANAGED-ID" property parameter value in the iCalendar data in the PUT request is not valid (e.g., does not match any existing managed attachment).

If a precondition for a request is not satisfied:

1) The response status of the request MUST either be 403 (Forbidden), if the request should not be repeated because it will always fail, or 409 (Conflict), if it is expected that the user might be able to resolve the conflict and resubmit the request.
2) The appropriate XML element MUST be returned as the child of a top-level DAV:error element in the response body.

4.11. Additional Considerations

4.11.1. Quotas

The WebDAV Quotas IETF RFC 4331 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.

4.11.2. Access Control

Access to the managed attachments store in a calendar object resource SHOULD be restricted to only those calendar users who have access to that calendar object either directly, or indirectly (via being an attendee who would receive a scheduling message).

When accessing a managed attachment, clients SHOULD be prepared to authenticate with the server storing the attachment resource. The credentials required to access the managed attachment store could be different from the ones used to access the CalDAV server.

This specification only allows organizers of scheduled events to add managed attachments. Servers MUST prevent attendees of scheduled events from adding, updating or removing managed attachments. In addition, the server MUST prevent a calendar user from re-using a managed attachment (based on its managed-id value), unless that user is the one who originally created the managed attachment.

4.11.3. Redirects

For POST requests that add or update attachment data, the server MAY issue a 307 (Temporary Redirect) IETF RFC 7231 or 308 (Permanent Redirect) IETF RFC 7538 response to require the client to re-issue the POST request using a different request-URI. As a result, clients SHOULD use the "100-continue" expectation defined in IETF RFC 7231, Section 5.1.1. Using this mechanism ensures that, if a redirect does occur, the client does not needlessly send the attachment data.

4.11.4. Processing Time

Clients can expect servers to take a while to respond to POST requests that include large attachment bodies. Servers SHOULD use the "102 (Processing)" interim response defined in IETF RFC 2518, Section 10.1 to keep the client connection alive if the POST request will take significant time to complete.

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 respond with "410 (Gone)" 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 IETF RFC 6638.

4.11.7. Migrating Calendar Data

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:https://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 field "filename" parameter and exposes the same security risks. As a consequence, clients MUST follow the guidelines expressed in IETF RFC 6266, Section 4.3 when consuming this parameter value. Similarly, servers MUST follow those same guidelines before storing a value.

Example:

```
ATTACH;FILENAME=agenda.html:https://attachments.example.com/
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 within the scope of the CalDAV server. This property parameter MUST NOT be present in the case of non-managed attachments.

Example:

```
ATTACH;MANAGED-ID=aUNhbGVuZGFy:https://attachments.example.
com/abcd.txt
```

6. Additional Message Header Fields

6.1. Cal-Managed-ID Response Header Field

The Cal-Managed-ID response header field provides the value of the MANAGED-ID parameter corresponding to a newly added ATTACH property.

ABNF

```
Cal-Managed-ID = "Cal-Managed-ID" ":"
paramtext
; "paramtext" is defined in <<RFC5545,
section=3.1>>
```

Example


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

The Cal-Managed-ID header field MUST only be sent by an origin server in response to a successful POST request with an action set to attachment-add or attachment-update. It MUST only appear once in a response and MUST NOT appear in trailers.

The Cal-Managed-ID header field is end to end and MUST be forwarded by intermediaries. Intermediaries MUST NOT insert, delete, or modify a Cal-Managed-ID header field.

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 (Clause 4.10) 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 (Clause 4.10) 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. Implementation Status

EDITORIAL NOTE —
RFC Editor: before publication please remove this section, the reference to IETF RFC 7942, and any resulting “URIs” references sub-section.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in IETF RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to IETF RFC 7942, “this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit”.

8.1. Calendar and Contacts Server

The open source Calendar and Contacts Server project is a standards-compliant server implementing the CalDAV protocol. This production level implementation supports all of the requirements described in this document and successfully interoperates with the Clause 8.4, Clause 8.5, Clause 8.7, and Clause 8.6 client implementations described below. This implementation is freely distributable under the terms of the Apache License, Version 2.0.

8.2. Cyrus Server

The open source Cyrus Server project is a highly scalable enterprise mail system which also supports calendaring. This production level CalDAV implementation supports all of the requirements described in this document and successfully interoperates with the Clause 8.4 and Clause 8.6 client implementations described below. This implementation is freely distributable under a BSD style license from Computing Services at Carnegie Mellon University.

8.3. Oracle Communications Calendar Server

The Oracle Communications Calendar Server project is a standards-compliant, scalable, enterprise-ready calendaring solution. This production level CalDAV implementation supports all of the requirements described in this document and successfully interoperates with the Clause 8.4 and Clause 8.6 client implementations described below. This implementation is proprietary and available for a free trial and/or purchase from the vendor.

8.4. Apple Calendar

The widely used Apple Calendar client is a standards-compliant client implementing the CalDAV protocol. This production level implementation supports all the requirements described in this document and successfully interoperates with the Clause 8.1, Clause 8.2, and Clause 8.3

implementations described above. This client implementation is proprietary and is distributed as part of Apple's desktop operating systems.

8.5. BusyCal

BusyCal is a standards-compliant calendar client for MacOS implementing the CalDAV protocol. This implementation supports all of the requirements described in this document and successfully interoperates with the Clause 8.1 and Clause 8.2 implementations described above. This implementation is proprietary and available for a free trial and/or purchase from the vendor.

8.6. CalDAVTester

CalDAVTester is an open source test and performance application designed to work with CalDAV servers and tests various aspects of their protocol handling as well as performance. This widely used implementation supports all of the requirements described in this document and successfully interoperates with the server implementations described above. This implementation is freely distributable under the terms of the Apache License, Version 2.0.

8.7. 2Do

2Do is a standards-complient calendar client for iOS which uses the CalDAV standard for communication. This implementation supports all of the requirements described in this document and successfully interoperates with the Clause 8.1 implementation described above. This implementation is proprietary and available for purchase from the vendor.

9. Security Considerations

The security considerations in IETF RFC 4791 and IETF RFC 4918 apply to this extension. Additionally, servers need to be aware that a client could attack underlying storage by POSTing extremely large attachments and could attack processing time by uploading a recurring event with a large number of overrides and then repeatedly adding, updating, and deleting attachments.

Malicious content could be introduced into the calendar server by way of a managed attachment, and propagated to many end users via scheduling. Servers SHOULD check managed attachments for malicious or inappropriate content. Upon detecting of such content, servers SHOULD remove the attachment, following the rules described in Clause 4.11.5.

10. IANA Considerations

10.1. Parameter Registrations

This specification defines the following new iCalendar property parameters to be added to the registry defined in IETF RFC 5545, Section 8.2.3:

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

10.2. Message Header Field Registrations

The message header fields below should be added to the Permanent Message Header Field Registry (see IETF RFC 3864).

10.2.1. Cal-Managed-ID

Header field name	Cal-Managed-ID
Applicable protocol	http
Status	standard
Author/Change controller	IETF
Specification document(s)	this specification (Clause 6.1)
Related information	none

11. Acknowledgments

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

Appendix A
(normative)
Example Involving Recurring Events

In the following example, the organizer of a recurring meeting makes an unsuccessful attempt to add an agenda (HTML attachment) to the corresponding calendar resource with a conditional request. Note that the client includes both the Expect and Prefer header fields in the request, thereby preventing itself from needlessly sending the attachment data, and requesting that the current resource be returned in the failure response (see IETF RFC 8144, Section 3.2).

```
>> 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
If-Match: "abcdefg-000"
Expect: 100-continue
Prefer: return=representation

>> Final Response <<

HTTP/1.1 412 Precondition Failed
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://cal.example.com/events/65.ics
ETag: "123456789-000-000"

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
END:VEVENT
END:VCALENDAR
```

The organizer of a recurring meeting successfully 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
If-Match: "123456789-000-000"
Expect: 100-continue
Prefer: return=representation

>> Interim Response <<

HTTP/1.1 100 Continue

>> Request Body <<

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

>> Final Response <<

HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://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. Note that an overridden instance is created with the `RECURRENCE-ID` property value matching the value of the "`rid`" query parameter in the request. Also note that the server takes significant time to complete the request and notifies the client accordingly.

```
>> 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
If-Match: "123456789-000-111"
Expect: 100-continue
Prefer: return=representation

>> Interim Response <<

HTTP/1.1 100 Continue

>> Request Body <<

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

>> Interim Response <<

HTTP/1.1 102 Processing

>> Final Response <<
```



```
HTTP/1.1 201 Created
Content-Type: text/calendar; charset="utf-8"
Content-Length: yyyy
Content-Location: https://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
```


Bibliography

[1] IETF RFC 5023, Internet Engineering Task Force (committee). *The Atom Publishing Protocol*. 2007. RFC Publisher. https://www.rfc-editor.org/info/rfc5023.

[2] IETF RFC 5546, Internet Engineering Task Force (committee). *iCalendar Transport-Independent Interoperability Protocol (iTIP)*. 2009. RFC Publisher. https://www.rfc-editor.org/info/rfc5546.

[3] IETF RFC 7320, NOTTINGHAM, M. and Internet Engineering Task Force. *URI Design and Ownership*. 2014. RFC Publisher. https://www.rfc-editor.org/info/rfc7320.

[4] IETF RFC 7942, SHEFFER, Y. and A. FARREL. *Improving Awareness of Running Code: The Implementation Status Section*. 2016. RFC Publisher. https://www.rfc-editor.org/info/rfc7942.

[5] IETF RFC 8144, MURCHISON, K. *Use of the Prefer Header Field in Web Distributed Authoring and Versioning (WebDAV)*. 2017. RFC Publisher. https://www.rfc-editor.org/info/rfc8144.

