Internet Engineering Task Force (IETF)
Request for Comments: 8144
Updates: 7240
Category: Standards Track
ISSN: 2070-1721

K. Murchison
CMU
April 2017

Use of the Prefer Header Field in Web Distributed Authoring and Versioning (WebDAV)

Abstract

This document defines how the Prefer header field (RFC 7240) can be used by a Web Distributed Authoring and Versioning (WebDAV) client to request that certain behaviors be employed by a server while constructing a response to a request. Furthermore, it defines the new "depth-noroot" preference.

This document updates RFC 7240.

Status of this Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc8144[1].

Copyright Notice

[1] http://www.rfc-editor.org/info/rfc8144
[2] http://trustee.ietf.org/license-info

Table of Contents

1. Introduction

[RFC7240] defines the Prefer header field and the "return=minimal" preference, which indicate that a client wishes for the server to return a minimal response to a successful request but states that what constitutes an appropriate minimal response is left solely to the discretion of the server. Section 2 of this specification defines precisely what is expected of a server when constructing minimal responses to successful WebDAV [RFC4918] requests.

[RFC7240] also defines the "return=representation" preference, which indicates that a client wishes for the server to include an entity representing the current state of the resource in the response to a successful request. Section 3 of this specification makes recommendations on when this preference should be used by clients and extends its applicability to 412 (Precondition Failed) [RFC7232] responses.

Finally, Section 4 of this specification defines the "depth-noroot" preference that can be used with HTTP methods that support the Depth header field.

1.1. Notational Conventions

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

This document references XML element types in the "DAV:" [RFC4918], "urn:ietf:params:xml:ns:caldav" [RFC4791], and "urn:ietf:params:xml:ns:carddav" [RFC6352] namespaces outside of the context of an XML fragment. When doing so, the strings "DAV:", "CALDAV:", and "CARDDAV:" will be prepended to the XML element types, respectively.

2. Reducing WebDAV Response Verbosity with "return=minimal"

Some payload bodies in responses to WebDAV requests, such as 207 (Multi-Status) [RFC4918] responses, can be quite verbose or even unnecessary at times. This specification defines how the Prefer header field, in conjunction with its "return=minimal" preference, can be used by clients to reduce the verbosity of such responses by requesting that the server omit those portions of the response that can be inferred by their absence.

2.1. Minimal PROPFIND and REPORT Responses

When a PROPFIND [RFC4918] request, or a REPORT [RFC3253] request whose report type results in a 207 (Multi-Status) response, contains a Prefer header field with a preference of "return=minimal", the server SHOULD omit all DAV:propstat XML elements containing a DAV:status XML element of value 404 (Not Found) [RFC7231] from the 207 (Multi-Status) response. If the omission of such a DAV:propstat element would result in a DAV:response XML element containing zero DAV:propstat elements, the server MUST substitute one of the following in its place:

- a DAV:propstat element consisting of an empty DAV:prop element and a DAV:status element of value 200 (OK) [RFC7231]
- a DAV:status element of value 200 (OK)

The following report types are candidates that could benefit from use of the "return=minimal" preference. NOTE: This list is not intended to be normative or exhaustive.

- DAV:expand-property [RFC3253]
- DAV:acl-principal-prop-set [RFC3744]
- DAV:principal-property-search [RFC3744]
- DAV:sync-collection [RFC6578]
- CALDAV:calendar-query [RFC4791]
- CALDAV:calendar-multiget [RFC4791]
- CARDDAV:addressbook-query [RFC6352]
- CARDDAV:addressbook-multiget [RFC6352]

See Appendices B.1 and B.2 for examples.

2.2. Minimal PROPPATCH Response

When a PROPPATCH [RFC4918] request contains a Prefer header field with a preference of "return=minimal", and all instructions are processed successfully, the server SHOULD return one of the following responses rather than a 207 (Multi-Status) response:

- 204 (No Content) [RFC7231]
- 200 (OK) [RFC7231] (preferably with a zero-length message body)

See Appendix B.3 for examples.

2.3. Minimal MKCALENDAR and MKCOL Responses

Both the MKCALENDAR [RFC4791] and Extended MKCOL [RFC5689] specifications indicate that a server MAY return a message body in response to a successful request. This specification explicitly defines the intended behavior in the presence of the Prefer header field.

When a MKCALENDAR or an extended MKCOL request contains a Prefer header field with a preference of "return=minimal", and the collection is created with all requested properties being set successfully, the server SHOULD return a 201 (Created) [RFC7231] response with an empty (zero-length) message body.

Note that the rationale for requiring that a minimal success response have an empty body is twofold:

- [RFC4791], Section 5.3.1 states: "If a response body for a successful request is included, it MUST be a CALDAV:mkcalendar-response XML element."

- [RFC5689], Section 3 states: "When an empty response body is returned with a success request status code, the client can assume that all properties were set."

See Appendix B.4 for examples.

3. Reducing WebDAV Roundtrips with "return=representation"

[RFC7240] describes the "return=representation" preference as being intended to provide a means of optimizing communication between the client and server by eliminating the need for a subsequent GET request to retrieve the current representation of the resource following a modification. This preference is equally applicable to situations where the server itself modifies a resource, and where a resource has been modified by another client.

3.1. Successful State-Changing Requests

The state-changing methods PUT [RFC7231], COPY/MOVE [RFC4918], PATCH [RFC5789], and POST [RFC5995] can be used to create or update a resource. In some instances, such as with Calendaring Extensions to WebDAV (CalDAV) Scheduling [RFC6638], the created or updated resource representation may differ from the representation sent in the body of the request or from that referenced by the effective request URI. In cases where the client, upon receiving a 2xx (Successful) [RFC7231] response to its state-changing request, would normally issue a subsequent GET request to retrieve the current representation of the resource, the client can instead include a Prefer header field with the "return=representation" preference in the state-changing request.

When a state-changing request contains a Prefer header field with a preference of "return=representation", and the resource is created or updated successfully, the server SHOULD include an entity representing the current state of the resource in the resulting 201 (Created) or 200 (OK) [RFC7231] response. In addition to coalescing the create/update and retrieve operations into a single roundtrip, by returning the current representation of the resource in the response, the client will know that any changes to the resource were produced by the server rather than a concurrent client, thus providing a level of atomicity to the operation.

See Appendix B.5 for examples.

3.2. Unsuccessful Conditional State-Changing Requests

Frequently, clients using a state-changing method such as those listed above will make them conditional by including either an If#Match or an If-None-Match [RFC7232] header field in the request. This is done to prevent the client from accidentally overwriting a resource whose current state has been modified by another client acting in parallel. In cases where the client, upon receiving a 412 (Precondition Failed) [RFC7232] response to its conditional state-changing request, would normally issue a subsequent GET request to retrieve the current representation of the resource, the client can instead include a Prefer header field with the "return=representation" preference in the conditional state-changing request.

When a conditional state-changing request contains a Prefer header field with a preference of "return=representation", and the specified condition evaluates to false, the server SHOULD include an entity representing the current state of the resource in the resulting 412 (Precondition Failed) [RFC7232] response.

See Appendix B.6 for examples.

4. The "depth-noroot" Processing Preference

The "depth-noroot" preference indicates that the client wishes for the server to exclude the target (root) resource from processing by the HTTP method and only apply the HTTP method to the target resource's subordinate resources.

This preference is only intended to be used with HTTP methods whose definitions explicitly provide support for the Depth [RFC4918] header field. Furthermore, this preference only applies when the Depth header field has a value of "1" or "infinity" (either implicitly or explicitly).

The "depth-noroot" preference MAY be used in conjunction with the "return=minimal" preference in a single request.

See Appendix B.1 for examples.

5. Security Considerations

No new security considerations are introduced by use of the Prefer header field with WebDAV requests, beyond those discussed in [RFC7240] and those already inherent in those requests.

6. IANA Considerations

6.1. Preference Registration

The following preference has been added to the HTTP Preferences Registry defined in Section 5.1 of [RFC7240].

Preference:	depth-noroot
Description:	The "depth-noroot" preference indicates that the client wishes for the server to exclude the target (root) resource from processing by the HTTP method and only apply the HTTP method to the target resource's subordinate resources.
Reference:	RFC 8144, Section 4
Notes:	This preference is only intended to be used with HTTP methods whose definitions explicitly provide support for the Depth [RFC4918] header field. Furthermore, this preference only applies when the Depth header field has a value of "1" or "infinity" (either implicitly or explicitly).

6.2. Method References

The following methods have had their references updated in the "HTTP Method Registry" (http://www.iana.org/assignments/http-methods).

Method Name	Safe	Idempotent	References
MKCALENDAR	no	yes	RFC 4791, Section 5.3.1; RFC 8144, Section 2.3
MKCOL	no	yes	RFC 4918, Section 9.3; RFC 5689, Section 3; RFC 8144, Section 2.3
PROPFIND	yes	yes	RFC 4918, Section 9.1; RFC 8144, Section 2.1
PROPPATCH	no	yes	RFC 4918, Section 9.2; RFC 8144, Section 2.2
REPORT	yes	yes	RFC 3253, Section 3.6; RFC 8144, Section 2.1

6.3. Status Code References

The following status code has had its references updated in the "HTTP Status Codes" registry (http://www.iana.org/assignments/http-status-codes).

Value	Description	References
412	Precondition Failed	RFC 7232, Section 4.2; RFC 8144, Section 3.2

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3253] Clemm, G., Amsden, J., Ellison, T., Kaler, C., and J. Whitehead, "Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning)", RFC 3253, DOI 10.17487/RFC3253, March 2002, <https://www.rfc-editor.org/info/rfc3253>.

[RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, DOI 10.17487/RFC4791, March 2007, <https://www.rfc-editor.org/info/rfc4791>.

[RFC4918] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, DOI 10.17487/RFC4918, June 2007, <https://www.rfc-editor.org/info/rfc4918>.

[RFC5689] Daboo, C., "Extended MKCOL for Web Distributed Authoring and Versioning (WebDAV)", RFC 5689, DOI 10.17487/RFC5689, September 2009, <https://www.rfc-editor.org/info/rfc5689>.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, DOI 10.17487/RFC5789, March 2010, <https://www.rfc-editor.org/info/rfc5789>.

[RFC5995] Reschke, J., "Using POST to Add Members to Web Distributed Authoring and Versioning (WebDAV) Collections", RFC 5995, DOI 10.17487/RFC5995, September 2010, <https://www.rfc-editor.org/info/rfc5995>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/info/rfc7232>.

[RFC7240] Snell, J., "Prefer Header for HTTP", RFC 7240, DOI 10.17487/RFC7240, June 2014, <https://www.rfc-editor.org/info/rfc7240>.

7.2. Informative References

[MSDN.aa493854] Microsoft Developer Network, "PROPPATCH Method", June 2006, <http://msdn.microsoft.com/en-us/library/aa493854.aspx>.

[MSDN.aa563501] Microsoft Developer Network, "Brief Header", June 2006, <http://msdn.microsoft.com/en-us/library/aa563501.aspx>.

[MSDN.aa563950] Microsoft Developer Network, "Depth Header", June 2006, <http://msdn.microsoft.com/en-us/library/aa563950.aspx>.

[MSDN.aa580336] Microsoft Developer Network, "PROPFIND Method", June 2006, <http://msdn.microsoft.com/en-us/library/aa580336.aspx>.

[RFC3744] Clemm, G., Reschke, J., Sedlar, E., and J. Whitehead, "Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol", RFC 3744, DOI 10.17487/RFC3744, May 2004, <https://www.rfc-editor.org/info/rfc3744>.

[RFC6352] Daboo, C., "CardDAV: vCard Extensions to Web Distributed Authoring and Versioning (WebDAV)", RFC 6352, DOI 10.17487/RFC6352, August 2011, <https://www.rfc-editor.org/info/rfc6352>.

[RFC6578] Daboo, C. and A. Quillaud, "Collection Synchronization for Web Distributed Authoring and Versioning (WebDAV)", RFC 6578, DOI 10.17487/RFC6578, March 2012, <https://www.rfc-editor.org/info/rfc6578>.

[RFC6638] Daboo, C. and B. Desruisseaux, "Scheduling Extensions to CalDAV", RFC 6638, DOI 10.17487/RFC6638, June 2012, <https://www.rfc-editor.org/info/rfc6638>.

A. The Brief and Extended Depth Header Fields

This document is based heavily on the Brief [MSDN.aa563501] and extended Depth [MSDN.aa563950] header fields. The behaviors described in Sections 2.1 and 2.2 are identical to those provided by the Brief header field when used with the PROPFIND [MSDN.aa580336] and PROPPATCH [MSDN.aa493854] methods, respectively. The behavior described in Section 4 is identical to that provided by the "1,noroot" [MSDN.aa563950] and "infinity,noroot" [MSDN.aa563950] Depth header field values.

Client and server implementations that already support the Brief header field can add support for the "return=minimal" preference with nominal effort.

If a server supporting the Prefer header field receives both the Brief and Prefer header fields in a request, clients can expect the server to ignore the Brief header field and only use the Prefer header field preferences.

B. Examples

B.1. PROPFIND

B.1.1. Typical PROPFIND Request/Response with Depth:1

This example tries to fetch one known and one unknown property from child resources.

>> Request <<

```
PROPFIND /container/ HTTP/1.1
Host: webdav.example.com
Content-Type: application/xml; charset=utf-8
Content-Length: 189
Depth: 1

<?xml version="1.0" encoding="UTF-8"?>
<D:propfind xmlns:D="DAV:" xmlns:X="http://ns.example.com/foobar/">
  <D:prop>
    <D:resourcetype/>
    <X:foobar/>
  </D:prop>
</D:propfind>
```

>> Response <<

```
HTTP/1.1 207 Multi-Status
Content-Type: application/xml; charset=utf-8
Content-Length: 1722

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:"
               xmlns:X="http://ns.example.com/foobar/">
  <D:response>
    <D:href>/container/</D:href>
    <D:propstat>
      <D:prop>
        <D:resourcetype>
          <D:collection/>
        </D:resourcetype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
    <D:propstat>
      <D:prop>
        <X:foobar/>
      </D:prop>
      <D:status>HTTP/1.1 404 Not Found</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:href>/container/work/</D:href>
    <D:propstat>
      <D:prop>
        <D:resourcetype>
          <D:collection/>
        </D:resourcetype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
    <D:propstat>
      <D:prop>
        <X:foobar/>
      </D:prop>
      <D:status>HTTP/1.1 404 Not Found</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:href>/container/home/</D:href>
    <D:propstat>
      <D:prop>
        <D:resourcetype>
          <D:collection/>
        </D:resourcetype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
    <D:propstat>
      <D:prop>
        <X:foobar/>
      </D:prop>
      <D:status>HTTP/1.1 404 Not Found</D:status>
    </D:propstat>
  </D:response>
```

B.1.2. Minimal PROPFIND Request/Response with Depth:1

This example tries to fetch one known and one unknown property from child resources only.

>> Request <<

```
PROPFIND /container/ HTTP/1.1
Host: webdav.example.com
Content-Type: application/xml; charset=utf-8
Content-Length: 189
Depth: 1
Prefer: return=minimal, depth-noroot

<?xml version="1.0" encoding="UTF-8"?>
<D:propfind xmlns:D="DAV:" xmlns:X="http://ns.example.com/foobar/">
  <D:prop>
    <D:resourcetype/>
    <X:foobar/>
  </D:prop>
</D:propfind>
```

>> Response <<

```
HTTP/1.1 207 Multi-Status
Content-Type: application/xml; charset=utf-8
Content-Length: 837
Preference-Applied: return=minimal, depth-noroot

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>/container/work/</D:href>
    <D:propstat>
      <D:prop>
        <D:resourcetype>
          <D:collection/>
        </D:resourcetype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:href>/container/home/</D:href>
    <D:propstat>
      <D:prop>
        <D:resourcetype>
          <D:collection/>
        </D:resourcetype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:href>/container/foo.txt</D:href>
    <D:propstat>
      <D:prop>
        <D:resourcetype/>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

B.1.3. Minimal PROPFIND Request/Response with an Empty DAV:propstat Element

This example tries to fetch an unknown property from a collection.

>> Request <<

```
PROPFIND /container/ HTTP/1.1
Host: webdav.example.com
Content-Type: application/xml; charset=utf-8
Content-Length: 166
Prefer: return=minimal

<?xml version="1.0" encoding="UTF-8"?>
<D:propfind xmlns:D="DAV:" xmlns:X="http://ns.example.com/foobar/">
  <D:prop>
    <X:foobar/>
  </D:prop>
</D:propfind>
```

>> Response <<

```
HTTP/1.1 207 Multi-Status
Content-Type: application/xml; charset=utf-8
Content-Length: 255
Preference-Applied: return=minimal

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>/container/</D:href>
    <D:propstat>
      <D:prop/>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

B.2. REPORT

B.2.1. Typical REPORT Request/Response

This example tries to fetch an unknown property from several resources via the DAV:expand-property [RFC3253] REPORT type.

>> Request <<

```
REPORT /dav/principals/ HTTP/1.1
Host: webdav.example.com
Content-type: text/xml; charset=utf-8
Content-length: 847

<?xml version="1.0" encoding="utf-8"?>
<D:expand-property xmlns:D="DAV:">
  <D:property name="current-user-principal">
    <D:property name="resourcetype"/>
    <D:property name="displayname"/>
    <D:property name="foobar"
                namespace="http://ns.example.com/foobar"/>
    <D:property name="calendar-home-set"
                namespace="urn:ietf:params:xml:ns:caldav">
      <D:property name="resourcetype"/>
      <D:property name="foobar"
                  namespace="http://ns.example.com/foobar"/>
    </D:property>
    <D:property name="addressbook-home-set"
                namespace="urn:ietf:params:xml:ns:carddav">
      <D:property name="resourcetype"/>
      <D:property name="foobar"
                  namespace="http://ns.example.com/foobar"/>
    </D:property>
  </D:property>
</D:expand-property>
```

>> Response <<

```
HTTP/1.1 207 Multi-Status
Content-Type: application/xml; charset=utf-8
Content-Length: 2664

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:"
               xmlns:C="urn:ietf:params:xml:ns:caldav"
               xmlns:R="urn:ietf:params:xml:ns:carddav"
               xmlns:X="http://ns.example.com/foobar">
  <D:response>
    <D:href>/dav/principals/</D:href>
    <D:propstat>
      <D:prop>
        <D:current-user-principal>
          <D:response>
            <D:href>/dav/principals/user/ken/</D:href>
            <D:propstat>
              <D:prop>
                <D:resourcetype>
                  <D:principal/>
                </D:resourcetype>
                <D:displayname>ken</D:displayname>
                <C:calendar-home-set>
                  <D:response>
                    <D:href>/dav/calendars/user/ken/</D:href>
                    <D:propstat>
                      <D:prop>
                        <D:resourcetype>
                          <D:collection/>
                        </D:resourcetype>
                      </D:prop>
                      <D:status>HTTP/1.1 200 OK</D:status>
                    </D:propstat>
                    <D:propstat>
                      <D:prop>
                        <X:foobar/>
                      </D:prop>
                      <D:status>HTTP/1.1 404 Not Found</D:status>
                    </D:propstat>
                  </D:response>
                </C:calendar-home-set>
                <R:addressbook-home-set>
                  <D:response>
                    <D:href>/dav/addressbooks/user/ken/</D:href>
                    <D:propstat>
                      <D:prop>
                        <D:resourcetype>
                          <D:collection/>
                        </D:resourcetype>
                      </D:prop>
                      <D:status>HTTP/1.1 200 OK</D:status>
                    </D:propstat>
                    <D:propstat>
                      <D:prop>
                        <X:foobar/>
                      </D:prop>
                      <D:status>HTTP/1.1 404 Not Found</D:status>
                    </D:propstat>
```

B.2.2. Minimal REPORT Request/Response

This example tries to fetch an unknown property from several resources via the DAV:expand-property [RFC3253] REPORT type.

>> Request <<

```
REPORT /dav/principals/ HTTP/1.1
Host: webdav.example.com
Content-Type: application/xml; charset=utf-8
Content-Length: 847
Prefer: return=minimal

<?xml version="1.0" encoding="utf-8"?>
<D:expand-property xmlns:D="DAV:">
  <D:property name="current-user-principal">
    <D:property name="resourcetype"/>
    <D:property name="displayname"/>
    <D:property name="foobar"
                namespace="http://ns.example.com/foobar"/>
    <D:property name="calendar-home-set"
                namespace="urn:ietf:params:xml:ns:caldav">
      <D:property name="resourcetype"/>
      <D:property name="foobar"
                  namespace="http://ns.example.com/foobar"/>
    </D:property>
    <D:property name="addressbook-home-set"
                namespace="urn:ietf:params:xml:ns:carddav">
      <D:property name="resourcetype"/>
      <D:property name="foobar"
                  namespace="http://ns.example.com/foobar"/>
    </D:property>
  </D:property>
</D:expand-property>
```

>> Response <<

```
HTTP/1.1 207 Multi-Status
Content-Type: application/xml; charset=utf-8
Content-Length: 1998
Preference-Applied: return=minimal

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:"
               xmlns:C="urn:ietf:params:xml:ns:caldav"
               xmlns:R="urn:ietf:params:xml:ns:carddav"
               xmlns:X="http://ns.example.com/foobar">
  <D:response>
    <D:href>/dav/principals/</D:href>
    <D:propstat>
      <D:prop>
        <D:current-user-principal>
          <D:response>
            <D:href>/dav/principals/user/ken/</D:href>
            <D:propstat>
              <D:prop>
                <D:resourcetype>
                  <D:principal/>
                </D:resourcetype>
                <D:displayname>ken</D:displayname>
                <C:calendar-home-set>
                  <D:response>
                    <D:href>/dav/calendars/user/ken/</D:href>
                    <D:propstat>
                      <D:prop>
                        <D:resourcetype>
                          <D:collection/>
                        </D:resourcetype>
                      </D:prop>
                      <D:status>HTTP/1.1 200 OK</D:status>
                    </D:propstat>
                  </D:response>
                </C:calendar-home-set>
                <R:addressbook-home-set>
                  <D:response>
                    <D:href>/dav/addressbooks/user/ken/</D:href>
                    <D:propstat>
                      <D:prop>
                        <D:resourcetype>
                          <D:collection/>
                        </D:resourcetype>
                      </D:prop>
                      <D:status>HTTP/1.1 200 OK</D:status>
                    </D:propstat>
                  </D:response>
                </R:addressbook-home-set>
              </D:prop>
              <D:status>HTTP/1.1 200 OK</D:status>
            </D:propstat>
          </D:response>
        </D:current-user-principal>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
```

B.3. PROPPATCH

B.3.1. Typical PROPPATCH Request/Response

>> Request <<

```
PROPPATCH /container/ HTTP/1.1
Host: webdav.example.com
Content-Type: application/xml; charset=utf-8
Content-Length: 199

<?xml version="1.0" encoding="utf-8"?>
<D:propertyupdate xmlns:D="DAV:">
  <D:set>
    <D:prop>
      <D:displayname>My Container</D:displayname>
    </D:prop>
  </D:set>
</D:propertyupdate>
```

>> Response <<

```
HTTP/1.1 207 Multi-Status
Content-Type: application/xml; charset=utf-8
Content-Length: 297

<?xml version="1.0" encoding="utf-8"?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>/container/</D:href>
    <D:propstat>
      <D:prop>
        <D:displayname/>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

B.3.2. Minimal PROPPATCH Request/Response

>> Request <<

```
PROPPATCH /container/ HTTP/1.1
Host: webdav.example.com
Content-Type: application/xml; charset=utf-8
Content-Length: 199
Prefer: return=minimal

<?xml version="1.0" encoding="utf-8"?>
<D:propertyupdate xmlns:D="DAV:">
  <D:set>
    <D:prop>
      <D:displayname>My Container</D:displayname>
    </D:prop>
  </D:set>
</D:propertyupdate>
```

>> Response <<

```
HTTP/1.1 200 OK
Content-Length: 0
Preference-Applied: return=minimal
```

B.4. MKCOL

B.4.1. Verbose MKCOL Request/Response

>> Request <<

```
MKCOL /container/ HTTP/1.1
Host: webdav.example.com
Content-Type: application/xml; charset=utf-8
Content-Length: 181

<?xml version="1.0" encoding="utf-8"?>
<D:mkcol xmlns:D="DAV:">
  <D:set>
    <D:prop>
      <D:displayname>My Container</D:displayname>
    </D:prop>
  </D:set>
</D:mkcol>
```

>> Response <<

```
HTTP/1.1 201 Created
Cache-Control: no-cache
Content-Type: application/xml; charset=utf-8
Content-Length: 224

<?xml version="1.0" encoding="utf-8"?>
<D:mkcol-response xmlns:D="DAV:">
  <D:propstat>
    <D:prop>
      <D:displayname/>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:mkcol-response>
```

B.4.2. Minimal MKCOL Request/Response

>> Request <<

```
MKCOL /container/ HTTP/1.1
Host: webdav.example.com
Content-Type: application/xml; charset=utf-8
Content-Length: 181
Prefer: return=minimal

<?xml version="1.0" encoding="utf-8"?>
<D:mkcol xmlns:D="DAV:">
  <D:set>
    <D:prop>
      <D:displayname>My Container</D:displayname>
    </D:prop>
  </D:set>
</D:mkcol>
```

>> Response <<

```
HTTP/1.1 201 Created
Cache-Control: no-cache
Content-Length: 0
Preference-Applied: return=minimal
```

B.5. POST

B.5.1. Typical Resource Creation and Retrieval via POST + GET

Note that this request is not conditional because by using the POST [RFC5995] method, the client lets the server choose the resource URI, thereby guaranteeing that it will not modify an existing resource.

>> Request <<

```
POST /container/work;add-member/ HTTP/1.1
Host: caldav.example.com
Content-Type: text/calendar; charset=utf-8
Content-Length: 521

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Client//EN
BEGIN:VEVENT
UID:CD87465FA
SEQUENCE:0
DTSTAMP:20120602T185254Z
DTSTART:20120602T160000Z
DTEND:20120602T170000Z
TRANSP:OPAQUE
SUMMARY:Lunch
ORGANIZER;CN="Ken Murchison":mailto:murch@example.com
ATTENDEE;CN="Ken Murchison";CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:
 mailto:murch@example.com
ATTENDEE;CN="John Doe";CUTYPE=INDIVIDUAL;PARTSTAT
 =NEEDS-ACTION;ROLE=REQ-PARTICIPANT;RSVP=TRUE:mailto:jdoe@
 example.com
END:VEVENT
END:VCALENDAR
```

>> Response <<

```
HTTP/1.1 201 Created
Location: /container/work/abc.ics
Content-Length: 0
```

Note that the server did not include any validator header fields (e.g., ETag) in the response, signaling that the created representation differs from the representation sent in the body of the request. The client has to send a separate GET request to retrieve the current representation:

>> Request <<

```
GET /container/work/abc.ics HTTP/1.1
Host: caldav.example.com
```

>> Response <<

```
HTTP/1.1 200 OK
Content-Type: text/calendar; charset=utf-8
Content-Length: 541
ETag: "nahduyejc"
Schedule-Tag: "jfd84hgbcn"

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VEVENT
UID:CD87465FA
SEQUENCE:0
DTSTAMP:20120602T185300Z
DTSTART:20120602T160000Z
DTEND:20120602T170000Z
TRANSP:OPAQUE
SUMMARY:Lunch
ORGANIZER;CN="Ken Murchison":mailto:murch@example.com
ATTENDEE;CN="Ken Murchison";CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:
 mailto:murch@example.com
ATTENDEE;CN="John Doe";CUTYPE=INDIVIDUAL;PARTSTAT
 =NEEDS-ACTION;ROLE=REQ-PARTICIPANT;RSVP=TRUE;SCHEDULE-STATUS=
 1.2:mailto:jdoe@example.com
END:VEVENT
END:VCALENDAR
```

B.5.2. Streamlined Resource Creation and Retrieval via POST

Note that this request is not conditional because by using the POST [RFC5995] method, the client lets the server choose the resource URI, thereby guaranteeing that it will not modify an existing resource.

>> Request <<

```
POST /container/work;add-member/ HTTP/1.1
Host: caldav.example.com
Content-Type: text/calendar; charset=utf-8
Content-Length: 521
Prefer: return=representation

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Client//EN
BEGIN:VEVENT
UID:CD87465FA
SEQUENCE:0
DTSTAMP:20120602T185254Z
DTSTART:20120602T160000Z
DTEND:20120602T170000Z
TRANSP:OPAQUE
SUMMARY:Lunch
ORGANIZER;CN="Ken Murchison":mailto:murch@example.com
ATTENDEE;CN="Ken Murchison";CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:
 mailto:murch@example.com
ATTENDEE;CN="John Doe";CUTYPE=INDIVIDUAL;PARTSTAT
 =NEEDS-ACTION;ROLE=REQ-PARTICIPANT;RSVP=TRUE:mailto:jdoe@
 example.com
END:VEVENT
END:VCALENDAR
```

>> Response <<

```
HTTP/1.1 201 Created
Location: /container/work/abc.ics
Content-Type: text/calendar; charset=utf-8
Content-Length: 541
Content-Location: /container/work/abc.ics
ETag: "nahduyejc"
Schedule-Tag: "jfd84hgbcn"
Preference-Applied: return=representation

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VEVENT
UID:CD87465FA
SEQUENCE:0
DTSTAMP:20120602T185300Z
DTSTART:20120602T160000Z
DTEND:20120602T170000Z
TRANSP:OPAQUE
SUMMARY:Lunch
ORGANIZER;CN="Ken Murchison":mailto:murch@example.com
ATTENDEE;CN="Ken Murchison";CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED:
 mailto:murch@example.com
ATTENDEE;CN="John Doe";CUTYPE=INDIVIDUAL;PARTSTAT
 =NEEDS-ACTION;ROLE=REQ-PARTICIPANT;RSVP=TRUE;SCHEDULE-STATUS=
 1.2:mailto:jdoe@example.com
END:VEVENT
END:VCALENDAR
```

B.6. PUT

B.6.1. Typical Conditional Resource Update Failure and Retrieval via PUT + GET

>> Request <<

```
PUT /container/motd.txt HTTP/1.1
Host: dav.example.com
Content-Type: text/plain
Content-Length: 69
If-Match: "asd973"

Either write something worth reading or do something worth writing.
```

>> Response <<

```
HTTP/1.1 412 Precondition Failed
Content-Length: 0
```

The resource has been modified by another user agent (ETag mismatch); therefore, the client has to send a separate GET request to retrieve the current representation:

>> Request <<

```
GET /container/motd.txt HTTP/1.1
Host: dav.example.com

```

>> Response <<

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 52
ETag: "789sdas"

An investment in knowledge pays the best interest.
```

B.6.2. Streamlined Conditional Resource Update Failure and Retrieval via PUT

>> Request <<

```
PUT /container/motd.txt HTTP/1.1
Host: dav.example.com
Content-Type: text/plain
Content-Length: 69
If-Match: "asd973"
Prefer: return=representation

Either write something worth reading or do something worth writing.
```

>> Response <<

```
HTTP/1.1 412 Precondition Failed
Content-Type: text/plain
Content-Length: 52
Content-Location: /container/motd.txt
ETag: "789sdas"
Preference-Applied: return=representation

An investment in knowledge pays the best interest.
```

Acknowledgements

The author would like to thank the following individuals for contributing their ideas and support for writing this specification: Cyrus Daboo, Helge Hess, Andrew McMillan, Arnaud Quillaud, and Julian Reschke.

The author would also like to thank the Calendaring and Scheduling Consortium for advice with this specification and for organizing interoperability testing events to help refine it.

Author's Address

Kenneth Murchison
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
United States of America
Phone: +1-412-268-1982
EMail: murch@andrew.cmu.edu