                     Collection Synchronization
        for Web Distributed Authoring and Versioning (WebDAV)

Abstract

   This specification defines an extension to Web Distributed Authoring
   and Versioning (WebDAV) that allows efficient synchronization of the
   contents of a WebDAV collection.

Copyright Notice

Table of Contents

1. Introduction

   WebDAV [RFC4918] defines the concept of 'collections', which are
   hierarchical groupings of WebDAV resources on an HTTP [RFC2616]
   server. Collections can be of arbitrary size and depth (i.e.,
   collections within collections). WebDAV clients that cache resource
   content need a way to synchronize that data with the server (i.e.,
   detect what has changed and update their cache). Currently, this can
   be done using a WebDAV PROPFIND request on a collection to list all
   members of a collection along with their DAV:getetag property values,
   which allows the client to determine which were changed, added, or
   deleted. However, this does not scale well to large collections, as
   the XML response to the PROPFIND request will grow with the
   collection size.

   This specification defines a new WebDAV report that results in the
   server returning to the client only information about those member
   URLs that were added or deleted, or whose mapped resources were
   changed, since a previous execution of the report on the collection.

   Additionally, a new property is added to collection resources that is
   used to convey a "synchronization token" that is guaranteed to change
   when the collection's member URLs or their mapped resources have
   changed.

2. Conventions Used in This Document

   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 uses XML DTD fragments ([W3C.REC-xml-20081126], Section
   3.2) as a purely notational convention. WebDAV request and response
   bodies cannot be validated by a DTD due to the specific extensibility
   rules defined in Section 17 of [RFC4918] and due to the fact that all
   XML elements defined by this specification use the XML namespace name
   "DAV:". In particular:

   1. Element names use the "DAV:" namespace.

   2. Element ordering is irrelevant unless explicitly stated
       otherwise.

   3. Extension elements (elements not already defined as valid child
       elements) may be added anywhere, except when explicitly stated
       otherwise.

   4. Extension attributes (attributes not already defined as valid for
       this element) may be added anywhere, except when explicitly
       stated otherwise.

   When an XML element type in the "DAV:" namespace is referenced in
   this document outside of the context of an XML fragment, the string
   "DAV:" will be prefixed to the element type.

   This document inherits, and sometimes extends, DTD productions from
   Section 14 of [RFC4918].

3. WebDAV Synchronization

3.1. Overview

   One way to synchronize data between two entities is to use some form
   of synchronization token. The token defines the state of the data
   being synchronized at a particular point in time. It can then be
   used to determine what has changed between one point in time and
   another.

   This specification defines a new WebDAV report that is used to enable
   client-server collection synchronization based on such a token.

   In order to synchronize the contents of a collection between a server
   and client, the server provides the client with a synchronization
   token each time the synchronization report is executed. That token
   represents the state of the data being synchronized at that point in
   time. The client can then present that same token back to the server
   at some later time, and the server will return only those items that
   are new, have changed, or were deleted since that token was
   generated. The server also returns a new token representing the new
   state at the time the report was run.

   Typically, the first time a client connects to the server it will
   need to be informed of the entire state of the collection (i.e., a
   full list of all member URLs that are currently in the collection).
   That is done by the client sending an empty token value to the
   server. This indicates to the server that a full listing is
   required.

   As an alternative, the client might choose to do its first
   synchronization using some other mechanism on the collection (e.g.,
   some other form of batch resource information retrieval such as
   PROPFIND, SEARCH [RFC5323], or specialized REPORTs such as those
   defined in CalDAV [RFC4791] and CardDAV [RFC6352]) and ask for the

DAV:sync-token property to be returned. This property (defined in
Section 4) contains the same token that can be used later to issue a
DAV:sync-collection report.

In some cases, a server might only wish to maintain a limited amount
of history about changes to a collection. In that situation, it will
return an error to the client when the client presents a token that
is "out of date". At that point, the client has to fall back to
synchronizing the entire collection by re-running the report request
using an empty token value.

Typically, a client will use the synchronization report to retrieve
the list of changes and will follow that with requests to retrieve
the content of changed resources. It is possible that additional
changes to the collection could occur between the time of the
synchronization report and resource content retrieval, which could
result in an inconsistent view of the collection. When clients use
this method of synchronization, they need to be aware that such
additional changes could occur and track them, e.g., by differences
between the ETag values returned in the synchronization report and
those returned when actually fetching resource content, by using
conditional requests as described in Section 5, or by repeating the
synchronization process until no changes are returned.

3.2. DAV:sync-collection Report

If the DAV:sync-collection report is implemented by a WebDAV server,
then the server MUST list the report in the
"DAV:supported-report-set" property on any collection that supports
synchronization.

To implement the behavior for this report, a server needs to keep
track of changes to any member URLs and their mapped resources in a
collection (as defined in Section 3 of [RFC4918]). This includes
noting the addition of new member URLs, the changes to the mapped
resources of existing member URLs, and the removal of member URLs.
The server will track each change and provide a synchronization
"token" to the client that describes the state of the server at a
specific point in time. This "token" is returned as part of the
response to the "sync-collection" report. Clients include the last
token they got from the server in the next "sync-collection" report
that they execute, and the server provides the changes from the
previous state (represented by the token) to the current state
(represented by the new token returned).

The synchronization token itself MUST be treated as an "opaque"
string by the client, i.e., the actual string data has no specific
meaning or syntax. However, the token MUST be a valid URI to allow

its use in an If precondition request header (see Section 5). For
example, a simple implementation of such a token could be a numeric
counter that counts each change as it occurs and relates that change
to the specific object that changed. The numeric value could be
appended to a "base" URI to form the valid sync-token.

Marshalling:

   The request-URI MUST identify a collection. The request body MUST
   be a DAV:sync-collection XML element (see Section 6.1), which MUST
   contain one DAV:sync-token XML element, one DAV:sync-level
   element, and one DAV:prop XML element, and MAY contain a DAV:limit
   XML element.

   This report is only defined when the Depth header has value "0";
   other values result in a 400 (Bad Request) error response. Note
   that [RFC3253], Section 3.6, states that if the Depth header is
   not present, it defaults to a value of "0".

   The response body for a successful request MUST be a
   DAV:multistatus XML element, which MUST contain one DAV:sync-token
   element in addition to one DAV:response element for each member
   URL that was added, has had its mapped resource changed, or was
   deleted since the last synchronization operation as specified by
   the DAV:sync-token provided in the request. A given member URL
   MUST appear only once in the response. In the case where multiple
   member URLs of the request-URI are mapped to the same resource, if
   the resource is changed, each member URL MUST be returned in the
   response.

   The content of each DAV:response element differs depending on how
   the member was altered:

      For members that have changed (i.e., are new or have had their
      mapped resource modified), the DAV:response MUST contain at
      least one DAV:propstat element and MUST NOT contain any
      DAV:status element.

      For members that have been removed, the DAV:response MUST
      contain one DAV:status with a value set to '404 Not Found' and
      MUST NOT contain any DAV:propstat element.

      For members that are collections and are unable to support the
      DAV:sync-collection report, the DAV:response MUST contain one
      DAV:status with a value set to '403 Forbidden', a DAV:error
      containing DAV:supported-report or DAV:sync-traversal-supported
      (see Section 3.3 for which is appropriate) and MUST NOT contain
      any DAV:propstat element.

The conditions under which each type of change can occur are
further described in Section 3.5.

Preconditions:

(DAV:valid-sync-token): The DAV:sync-token element value MUST be a
valid token previously returned by the server for the collection
targeted by the request-URI. Servers might need to invalidate
tokens previously returned to clients. Doing so will cause the
clients to fall back to doing full synchronization using the
report, though that will not require clients to download resources
that are already cached and have not changed. Even so, servers
MUST limit themselves to invalidating tokens only when absolutely
necessary. Specific reasons include:

* Servers might be unable to maintain all of the change data for
  a collection due to storage or performance reasons, e.g.,
  servers might only be able to maintain up to 3 weeks worth of
  changes to a collection, or only up to 10,000 total changes, or
  not wish to maintain changes for a deleted collection.

* Change to server implementation: servers might be upgraded to a
  new implementation that tracks the history in a different
  manner, and thus previous synchronization history is no longer
  valid.

Postconditions:

(DAV:number-of-matches-within-limits): The number of changes
reported in the response must fall within the client-specified
limit. This condition might be triggered if a client requests a
limit on the number of responses (as per Section 3.7), but the
server is unable to truncate the result set at or below that
limit.

3.3. Depth Behavior

Servers MUST support only Depth:0 behavior with the
DAV:sync-collection report, i.e., the report targets only the
collection being synchronized in a single request. However, clients
do need to "scope" the synchronization to different levels within
that collection -- specifically, immediate children (level "1") and
all children at any depth (level "infinite"). To specify which level
to use, clients MUST include a DAV:sync-level XML element in the
request.

   o When the client specifies the DAV:sync-level XML element with a
      value of "1", only appropriate internal member URLs (immediate
      children) of the collection specified as the request-URI are
      reported.

   o When the client specifies the DAV:sync-level XML element with a
      value of "infinite", all appropriate member URLs of the collection
      specified as the request-URI are reported, provided child
      collections themselves also support the DAV:sync-collection
      report.

   o DAV:sync-token values returned by the server are not specific to
      the value of the DAV:sync-level XML element used in the request.
      As such, clients MAY use a DAV:sync-token value from a request
      with one DAV:sync-level XML element value for a similar request
      with a different DAV:sync-level XML element value; however, the
      utility of this is limited.

   Note that when a server supports a DAV:sync-level XML element with a
   value of "infinite", it might not be possible to synchronize some
   child collections within the collection targeted by the report. When
   this occurs, the server MUST include a DAV:response element for the
   child collection with status 403 (Forbidden). The 403 response MUST
   be sent once, when the collection is first reported to the client.
   In addition, the server MUST include a DAV:error element in the
   DAV:response element, indicating one of two possible causes for this:

      The DAV:sync-collection report is not supported at all on the
      child collection. The DAV:error element MUST contain the
      DAV:supported-report element.

      The server is unwilling to report results for the child collection
      when a DAV:sync-collection report with the DAV:sync-level XML
      element set to "infinite" is executed on a parent resource. This
      might happen when, for example, the synchronization state of the
      collection resource is controlled by another subsystem. In such
      cases clients can perform the DAV:sync-collection report directly
      on the child collection instead. The DAV:error element MUST
      contain the DAV:sync-traversal-supported element.

3.4. Types of Changes Reported on Initial Synchronization

   When the DAV:sync-collection request contains an empty DAV:sync-token
   element, the server MUST return all member URLs of the collection
   (taking account of the DAV:sync-level XML element value as per
   Section 3.3, and optional truncation of the result set as per
   Section 3.6) and it MUST NOT return any removed member URLs. All
   types of member (collection or non-collection) MUST be reported.

3.5. Types of Changes Reported on Subsequent Synchronizations

   When the DAV:sync-collection request contains a valid value for the
   DAV:sync-token element, two types of member URL state changes can be
   returned (changed or removed). This section defines what triggers
   each of these to be returned. It also clarifies the case where a
   member URL might have undergone multiple changes between two
   synchronization report requests. In all cases, the DAV:sync-level
   XML element value (as per Section 3.3) and optional truncation of the
   result set (as per Section 3.6) are taken into account by the server.

3.5.1. Changed Member

   A member URL MUST be reported as changed if it has been newly mapped
   as a member of the target collection since the request sync-token was
   generated (e.g., when a new resource has been created as a child of
   the collection). For example, this includes member URLs that have
   been newly mapped as the result of a COPY, MOVE, BIND [RFC5842], or
   REBIND [RFC5842] request. All types of member URL (collection or
   non-collection) MUST be reported.

   In the case where a mapping between a member URL and the target
   collection was removed, then a new mapping with the same URI was
   created, the member URL MUST be reported as changed and MUST NOT be
   reported as removed.

   A member URL MUST be reported as changed if its mapped resource's
   entity tag value (defined in Section 3.11 of [RFC2616]) has changed
   since the request sync-token was generated.

   A member URL MAY be reported as changed if the user issuing the
   request was granted access to this member URL, due to access control
   changes.

   Collection member URLs MUST be returned as changed if they are mapped
   to an underlying resource (i.e., entity body) and if the entity tag
   associated with that resource changes. There is no guarantee that
   changes to members of a collection will result in a change in any
   entity tag of that collection, so clients cannot rely on a series of
   reports using the DAV:sync-level XML element value set to "1" at
   multiple levels to track all changes within a collection. Instead, a
   DAV:sync-level XML element with a value of "infinite" has to be used.

3.5.2. Removed Member

   A member MUST be reported as removed if its mapping under the target
   collection has been removed since the request sync-token was
   generated, and it has not been remapped since it was removed. For

example, this includes members that have been unmapped as the result
of a MOVE, UNBIND [RFC5842], or REBIND [RFC5842] operation. This
also includes collection members that have been removed, including
ones that themselves do not support the DAV:sync-collection report.

If a member was added (and its mapped resource possibly modified),
then removed between two synchronization report requests, it MUST be
reported as removed. This ensures that a client that adds a member
is informed of the removal of the member, if the removal occurs
before the client has had a chance to execute a synchronization
report.

A member MAY be reported as removed if the user issuing the request
no longer has access to this member, due to access control changes.

For a report with the DAV:sync-level XML element value set to
"infinite", where a collection is removed, the server MUST NOT report
the removal of any members of the removed collection. Clients MUST
assume that if a collection is reported as being removed, then all
members of that collection have also been removed.

3.6. Truncation of Results

A server MAY limit the number of member URLs in a response, for
example, to limit the amount of work expended in processing a
request, or as the result of an explicit limit set by the client. If
the result set is truncated, the response MUST use status code 207
(Multi-Status), return a DAV:multistatus response body, and indicate
a status of 507 (Insufficient Storage) for the request-URI. That
DAV:response element SHOULD include a DAV:error element with the
DAV:number-of-matches-within-limits precondition, as defined in
[RFC3744] (Section 9.2). DAV:response elements for all the changes
being reported are also included.

When truncation occurs, the DAV:sync-token value returned in the
response MUST represent the correct state for the partial set of
changes returned. That allows the client to use the returned
DAV:sync-token to fetch the next set of changes. In this way, the
client can effectively "page" through the entire set of changes in a
consistent manner.

Clients MUST handle the 507 status on the request-URI in the response
to the report.

For example, consider a server that records changes using a strictly
increasing integer to represent a "revision number" and uses that
quantity as the DAV:sync-token value (appropriately encoded as a
URI). Assume the last DAV:sync-token used by the client was

"http://example.com/sync/10", and since then 15 additional changes to
different resources have occurred. If the client executes a
DAV:sync-collection request with a DAV:sync-token of
"http://example.com/sync/10", without a limit, the server would
return 15 DAV:response elements and a DAV:sync-token with value
"http://example.com/sync/25". But if the server chooses to limit
responses to at most 10 changes, then it would return only 10
DAV:response elements and a DAV:sync-token with value
"http://example.com/sync/20", together with an additional
DAV:response element for the request-URI with a status code of 507.
Subsequently, the client can reissue the request with the
DAV:sync-token value returned from the server and fetch the remaining
5 changes.

3.7. Limiting Results

   A client can limit the number of results returned by the server
   through use of the DAV:limit element ([RFC5323], Section 5.17) in the
   request body. This is useful when clients have limited space or
   bandwidth for the results. If a server is unable to truncate the
   result at or below the requested number, then it MUST fail the
   request with a DAV:number-of-matches-within-limits postcondition
   error. When the results can be correctly limited by the server, the
   server MUST follow the rules above for indicating a result set
   truncation to the client.

3.8. Example: Initial DAV:sync-collection Report

   In this example, the client is making its first synchronization
   request to the server, so the DAV:sync-token element in the request
   is empty. It also asks for the DAV:getetag property and for a
   proprietary property. The server responds with the items currently
   in the targeted collection. The current synchronization token is
   also returned.

   >> Request <<

   REPORT /home/cyrusdaboo/ HTTP/1.1
   Host: webdav.example.com
   Depth: 0
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:sync-collection xmlns:D="DAV:">
     <D:sync-token/>
     <D:sync-level>1</D:sync-level>

```
      <D:prop xmlns:R="urn:ns.example.com:boxschema">
        <D:getetag/>
        <R:bigbox/>
      </D:prop>
    </D:sync-collection>


    >> Response <<


    HTTP/1.1 207 Multi-Status
    Content-Type: text/xml; charset="utf-8"
    Content-Length: xxxx

    <?xml version="1.0" encoding="utf-8" ?>
    <D:multistatus xmlns:D="DAV:">
      <D:response>
        <D:href
    >http://webdav.example.com/home/cyrusdaboo/test.doc</D:href>
        <D:propstat>
          <D:prop>
            <D:getetag>"00001-abcd1"</D:getetag>
            <R:bigbox xmlns:R="urn:ns.example.com:boxschema">
              <R:BoxType>Box type A</R:BoxType>
            </R:bigbox>
          </D:prop>
          <D:status>HTTP/1.1 200 OK</D:status>
        </D:propstat>
      </D:response>
      <D:response>
        <D:href
    >http://webdav.example.com/home/cyrusdaboo/vcard.vcf</D:href>
        <D:propstat>
          <D:prop>
            <D:getetag>"00002-abcd1"</D:getetag>
          </D:prop>
          <D:status>HTTP/1.1 200 OK</D:status>
        </D:propstat>
        <D:propstat>
          <D:prop>
            <R:bigbox xmlns:R="urn:ns.example.com:boxschema"/>
          </D:prop>
          <D:status>HTTP/1.1 404 Not Found</D:status>
        </D:propstat>
      </D:response>
      <D:response>
        <D:href
    >http://webdav.example.com/home/cyrusdaboo/calendar.ics</D:href>
```

```
      <D:propstat>
        <D:prop>
          <D:getetag>"00003-abcd1"</D:getetag>
        </D:prop>
        <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
      <D:propstat>
        <D:prop>
          <R:bigbox xmlns:R="urn:ns.example.com:boxschema"/>
        </D:prop>
        <D:status>HTTP/1.1 404 Not Found</D:status>
      </D:propstat>
    </D:response>
    <D:sync-token>http://example.com/ns/sync/1234</D:sync-token>
  </D:multistatus>
```

3.9. Example: DAV:sync-collection Report with Token

   In this example, the client is making a synchronization request to
   the server and is using the DAV:sync-token element returned from the
   last report it ran on this collection. The server responds, listing
   the items that have been added, changed, or removed. The (new)
   current synchronization token is also returned.

   >> Request <<

   REPORT /home/cyrusdaboo/ HTTP/1.1
   Host: webdav.example.com
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

```
   <?xml version="1.0" encoding="utf-8" ?>
   <D:sync-collection xmlns:D="DAV:">
     <D:sync-token>http://example.com/ns/sync/1234</D:sync-token>
     <D:sync-level>1</D:sync-level>
     <D:prop xmlns:R="urn:ns.example.com:boxschema">
       <D:getetag/>
       <R:bigbox/>
     </D:prop>
   </D:sync-collection>
```

```
   >> Response <<


   HTTP/1.1 207 Multi-Status
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:multistatus xmlns:D="DAV:">
     <D:response>
       <D:href
   >http://webdav.example.com/home/cyrusdaboo/file.xml</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"00004-abcd1"</D:getetag>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
       </D:propstat>
       <D:propstat>
         <D:prop>
           <R:bigbox xmlns:R="urn:ns.example.com:boxschema"/>
         </D:prop>
         <D:status>HTTP/1.1 404 Not Found</D:status>
       </D:propstat>
     </D:response>
     <D:response>
       <D:href
   >http://webdav.example.com/home/cyrusdaboo/vcard.vcf</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"00002-abcd2"</D:getetag>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
       </D:propstat>
       <D:propstat>
         <D:prop>
           <R:bigbox xmlns:R="urn:ns.example.com:boxschema"/>
         </D:prop>
         <D:status>HTTP/1.1 404 Not Found</D:status>
       </D:propstat>
     </D:response>
     <D:response>
       <D:href
   >http://webdav.example.com/home/cyrusdaboo/test.doc</D:href>
       <D:status>HTTP/1.1 404 Not Found</D:status>
     </D:response>
     <D:sync-token>http://example.com/ns/sync/1238</D:sync-token>
   </D:multistatus>
```

3.10. Example: Initial DAV:sync-collection Report with Truncation

   In this example, the client is making its first synchronization
   request to the server, so the DAV:sync-token element in the request
   is empty. It also asks for the DAV:getetag property. The server
   responds with the items currently in the targeted collection but
   truncated at two items. The synchronization token for the truncated
   result set is returned.

   >> Request <<

   REPORT /home/cyrusdaboo/ HTTP/1.1
   Host: webdav.example.com
   Depth: 0
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:sync-collection xmlns:D="DAV:">
     <D:sync-token/>
     <D:sync-level>1</D:sync-level>
     <D:prop>
       <D:getetag/>
     </D:prop>
   </D:sync-collection>

   >> Response <<

   HTTP/1.1 207 Multi-Status
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:multistatus xmlns:D="DAV:">
     <D:response>
       <D:href
   >http://webdav.example.com/home/cyrusdaboo/test.doc</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"00001-abcd1"</D:getetag>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
       </D:propstat>
     </D:response>
     <D:response>

```
      <D:href
   >http://webdav.example.com/home/cyrusdaboo/vcard.vcf</D:href>
      <D:propstat>
        <D:prop>
          <D:getetag>"00002-abcd1"</D:getetag>
        </D:prop>
        <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
    </D:response>
    <D:response>
      <D:href
   >http://webdav.example.com/home/cyrusdaboo/</D:href>
      <D:status>HTTP/1.1 507 Insufficient Storage</D:status>
      <D:error><D:number-of-matches-within-limits/></D:error>
    </D:response>
    <D:sync-token>http://example.com/ns/sync/1233</D:sync-token>
  </D:multistatus>
```

3.11. Example: Initial DAV:sync-collection Report with Limit

   In this example, the client is making its first synchronization
   request to the server, so the DAV:sync-token element in the request
   is empty. It requests a limit of 1 for the responses returned by the
   server. It also asks for the DAV:getetag property. The server
   responds with the items currently in the targeted collection, but
   truncated at one item. The synchronization token for the truncated
   result set is returned.

   >> Request <<

   REPORT /home/cyrusdaboo/ HTTP/1.1
   Host: webdav.example.com
   Depth: 0
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:sync-collection xmlns:D="DAV:">
     <D:sync-token/>
     <D:sync-level>1</D:sync-level>
     <D:limit>
       <D:nresults>1</D:nresults>
     </D:limit>
     <D:prop>
       <D:getetag/>
     </D:prop>
   </D:sync-collection>
```

```
   >> Response <<


   HTTP/1.1 207 Multi-Status
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:multistatus xmlns:D="DAV:">
     <D:response>
       <D:href
   >http://webdav.example.com/home/cyrusdaboo/test.doc</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"00001-abcd1"</D:getetag>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
       </D:propstat>
     </D:response>
     <D:response>
       <D:href
   >http://webdav.example.com/home/cyrusdaboo/</D:href>
       <D:status>HTTP/1.1 507 Insufficient Storage</D:status>
       <D:error><D:number-of-matches-within-limits/></D:error>
     </D:response>
     <D:sync-token>http://example.com/ns/sync/1232</D:sync-token>
   </D:multistatus>
```

3.12. Example: DAV:sync-collection Report with Unsupported Limit

   In this example, the client is making a synchronization request to
   the server with a valid DAV:sync-token element value. It requests a
   limit of 100 for the responses returned by the server. It also asks
   for the DAV:getetag property. The server is unable to limit the
   results to the maximum specified by the client, so it responds with a
   507 status code and appropriate postcondition error code.

```
   >> Request <<


   REPORT /home/cyrusdaboo/ HTTP/1.1
   Host: webdav.example.com
   Depth: 0
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx
```

```
      <?xml version="1.0" encoding="utf-8" ?>
      <D:sync-collection xmlns:D="DAV:">
        <D:sync-token>http://example.com/ns/sync/1232</D:sync-token>
        <D:sync-level>1</D:sync-level>
        <D:limit>
          <D:nresults>100</D:nresults>
        </D:limit>
        <D:prop>
          <D:getetag/>
        </D:prop>
      </D:sync-collection>


      >> Response <<


      HTTP/1.1 507 Insufficient Storage
      Content-Type: text/xml; charset="utf-8"
      Content-Length: xxxx

      <?xml version="1.0" encoding="utf-8" ?>
      <D:error xmlns:D="DAV:">
        <D:number-of-matches-within-limits/>
      </D:error>
```

3.13. Example: DAV:sync-level Set to Infinite, Initial
       DAV:sync-collection Report

   In this example, the client is making its first synchronization
   request to the server, so the DAV:sync-token element in the request
   is empty, and it is using DAV:sync-level set to "infinite". It also
   asks for the DAV:getetag property and for a proprietary property.
   The server responds with the items currently in the targeted
   collection. The current synchronization token is also returned.

   The collection /home/cyrusdaboo/collection1/ exists and has one child
   resource that is also reported. The collection /home/cyrusdaboo/
   collection2/ exists but has no child resources. The collection
   /home/cyrusdaboo/shared/ is returned with a 403 status indicating
   that a collection exists, but it is unable to report on changes
   within it in the scope of the current DAV:sync-level "infinite"
   report. Instead, the client can try a DAV:sync-collection report
   directly on the collection URI.

```
   >> Request <<


   REPORT /home/cyrusdaboo/ HTTP/1.1
   Host: webdav.example.com
   Depth: 0
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:sync-collection xmlns:D="DAV:">
     <D:sync-token/>
     <D:sync-level>infinite</D:sync-level>
     <D:prop xmlns:R="urn:ns.example.com:boxschema">
       <D:getetag/>
       <R:bigbox/>
     </D:prop>
   </D:sync-collection>


   >> Response <<


   HTTP/1.1 207 Multi-Status
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:multistatus xmlns:D="DAV:">
     <D:response>
       <D:href>/home/cyrusdaboo/collection1/</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"00001-abcd1"</D:getetag>
           <R:bigbox xmlns:R="urn:ns.example.com:boxschema">
             <R:BoxType>Box type A</R:BoxType>
           </R:bigbox>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
       </D:propstat>
     </D:response>
     <D:response>
       <D:href>/home/cyrusdaboo/collection1/test.doc</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"00001-abcd1"</D:getetag>
           <R:bigbox xmlns:R="urn:ns.example.com:boxschema">
             <R:BoxType>Box type A</R:BoxType>
```

```
                    </R:bigbox>
                  </D:prop>
                  <D:status>HTTP/1.1 200 OK</D:status>
                </D:propstat>
              </D:response>
              <D:response>
                <D:href>/home/cyrusdaboo/collection2/</D:href>
                <D:propstat>
                  <D:prop>
                    <D:getetag/>
                  </D:prop>
                  <D:status>HTTP/1.1 404 Not Found</D:status>
                </D:propstat>
                <D:propstat>
                  <D:prop>
                    <R:bigbox xmlns:R="urn:ns.example.com:boxschema"/>
                  </D:prop>
                  <D:status>HTTP/1.1 404 Not Found</D:status>
                </D:propstat>
              </D:response>
              <D:response>
                <D:href>/home/cyrusdaboo/calendar.ics</D:href>
                <D:propstat>
                  <D:prop>
                    <D:getetag>"00003-abcd1"</D:getetag>
                  </D:prop>
                  <D:status>HTTP/1.1 200 OK</D:status>
                </D:propstat>
                <D:propstat>
                  <D:prop>
                    <R:bigbox xmlns:R="urn:ns.example.com:boxschema"/>
                  </D:prop>
                  <D:status>HTTP/1.1 404 Not Found</D:status>
                </D:propstat>
              </D:response>
              <D:response>
                <D:href>/home/cyrusdaboo/shared/</D:href>
                <D:status>HTTP/1.1 403 Forbidden</D:status>
                <D:error><D:sync-traversal-supported/></D:error>
              </D:response>
              <D:sync-token>http://example.com/ns/sync/1234</D:sync-token>
            </D:multistatus>
```

4. DAV:sync-token Property

   Name: sync-token

   Namespace: DAV:

   Purpose: Contains the value of the synchronization token as it would
      be returned by a DAV:sync-collection report.

   Value: Any valid URI.

   Protected: MUST be protected because this value is created and
      controlled by the server.

   COPY/MOVE behavior: This property value is dependent on the final
      state of the destination resource, not the value of the property
      on the source resource.

   Description: The DAV:sync-token property MUST be defined on all
      resources that support the DAV:sync-collection report. It
      contains the value of the synchronization token as it would be
      returned by a DAV:sync-collection report on that resource at the
      same point in time. It SHOULD NOT be returned by a PROPFIND
      DAV:allprop request (as defined in Section 14.2 of [RFC4918]).

   Definition:

   <!ELEMENT sync-token #PCDATA>

   <!-- Text MUST be a valid URI -->

5. DAV:sync-token Use with If Header

   WebDAV provides an If precondition header that allows for "state
   tokens" to be used as preconditions on HTTP requests (as defined in
   Section 10.4 of [RFC4918]). This specification allows the
   DAV:sync-token value to be used as one such token in an If header.
   By using this, clients can ensure requests only complete when there
   have been no changes to the content of a collection, by virtue of an
   unchanged DAV:sync-token value. Servers MUST support use of
   DAV:sync-token values in If request headers.

5.1. Example: If Precondition with PUT

   In this example, the client has already used the DAV:sync-collection
   report to synchronize the collection /home/cyrusdaboo/collection/.
   Now it wants to add a new resource to the collection, but only if
   there have been no other changes since the last synchronization.

   Note that because the DAV:sync-token is defined on the collection and
   not on the resource targeted by the request, the If header value
   needs to use the "Resource_Tag" construct for the header syntax to
   correctly identify that the supplied state token refers to the
   collection resource.

   >> Request <<

   PUT /home/cyrusdaboo/collection/newresource.txt HTTP/1.1
   Host: webdav.example.com
   If: </home/cyrusdaboo/collection/>
     (<http://example.com/ns/sync/12345>)
   Content-Type: text/plain; charset="utf-8"
   Content-Length: xxxx

   Some content here...

   >> Response <<

   HTTP/1.1 201 Created

5.2. Example: If Precondition with MKCOL

   In this example, the client has already used the DAV:sync-collection
   report to synchronize the collection /home/cyrusdaboo/collection/.
   Now, it wants to add a new collection to the collection, but only if
   there have been no other changes since the last synchronization.
   Note that because the DAV:sync-token is defined on the collection and
   not on the resource targeted by the request, the If header value
   needs to use the "Resource_Tag" construct for the header syntax to
   correctly identify that the supplied state token refers to the
   collection resource. In this case, the request fails as another
   change has occurred to the collection corresponding to the supplied
   DAV:sync-token.

   >> Request <<

   MKCOL /home/cyrusdaboo/collection/child/ HTTP/1.1
   Host: webdav.example.com
   If: </home/cyrusdaboo/collection/>
     (<http://example.com/ns/sync/12346>)

>> Response <<

HTTP/1.1 412 Precondition Failed

6. XML Element Definitions

6.1. DAV:sync-collection XML Element

   Name: sync-collection

   Namespace: DAV:

   Purpose: WebDAV report used to synchronize data between client and
      server.

   Description: See Section 3.

   <!ELEMENT sync-collection (sync-token, sync-level, limit?, prop)>

   <!-- DAV:limit defined in RFC 5323, Section 5.17 -->
   <!-- DAV:prop defined in RFC 4918, Section 14.18 -->

6.2. DAV:sync-token XML Element

   Name: sync-token

   Namespace: DAV:

   Purpose: The synchronization token provided by the server and
      returned by the client.

   Description: See Section 3.

   <!ELEMENT sync-token CDATA>

   <!-- Text MUST be a URI -->

6.3. DAV:sync-level XML Element

   Name: sync-level

   Namespace: DAV:

   Purpose: Indicates the "scope" of the synchronization report
      request.

   Description: See Section 3.3.

```
<!ELEMENT sync-level CDATA>

<!-- Text MUST be either "1" or "infinite" -->
```

6.4. DAV:multistatus XML Element

   Name: multistatus

   Namespace: DAV:

   Purpose: Extends the DAV:multistatus element to include
      synchronization details.

   Description: See Section 3.

```
<!ELEMENT multistatus (response*, responsedescription?,
                       sync-token?) >

<!-- DAV:multistatus originally defined in RFC 4918, Section 14.16
     but overridden here to add the DAV:sync-token element -->
<!-- DAV:response defined in RFC 4918, Section 14.24 -->
<!-- DAV:responsedescription defined in RFC 4918, Section 14.25 -->
```

7. Security Considerations

   This extension does not introduce any new security concerns beyond
   those already described in HTTP and WebDAV.

8. Acknowledgments

   The following individuals contributed their ideas and support for
   writing this specification: Bernard Desruisseaux, Werner Donne, Mike
   Douglass, Ciny Joy, Andrew McMillan, Julian Reschke, and Wilfredo
   Sanchez. We would like to thank the Calendaring and Scheduling
   Consortium for facilitating interoperability testing for early
   implementations of this specification.

9. References

9.1. Normative References

   [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.

   [RFC3253] Clemm, G., Amsden, J., Ellison, T., Kaler, C., and J.
              Whitehead, "Versioning Extensions to WebDAV
              (Web Distributed Authoring and Versioning)", RFC 3253,
              March 2002.

   [RFC3744] Clemm, G., Reschke, J., Sedlar, E., and J. Whitehead, "Web
              Distributed Authoring and Versioning (WebDAV)
              Access Control Protocol", RFC 3744, May 2004.

   [RFC4918] Dusseault, L., "HTTP Extensions for Web Distributed
              Authoring and Versioning (WebDAV)", RFC 4918, June 2007.

   [RFC5323] Reschke, J., Reddy, S., Davis, J., and A. Babich, "Web
              Distributed Authoring and Versioning (WebDAV) SEARCH",
              RFC 5323, November 2008.

   [W3C.REC-xml-20081126]
              Sperberg-McQueen, C., Yergeau, F., Paoli, J., Maler, E.,
              and T. Bray, "Extensible Markup Language (XML) 1.0 (Fifth
              Edition)", World Wide Web Consortium
              Recommendation REC-xml-20081126, November 2008,
              <http://www.w3.org/TR/2008/REC-xml-20081126>.

9.2. Informative References

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

   [RFC5842] Clemm, G., Crawford, J., Reschke, J., and J. Whitehead,
              "Binding Extensions to Web Distributed Authoring and
              Versioning (WebDAV)", RFC 5842, April 2010.

   [RFC6352] Daboo, C., "CardDAV: vCard Extensions to Web Distributed
              Authoring and Versioning (WebDAV)", RFC 6352, August 2011.

Appendix A. Backwards-Compatible Handling of Depth

   In prior draft versions of this specification, the Depth request
   header was used instead of the DAV:sync-level element to indicate the
   "scope" of the synchronization request. Servers that wish to be
   backwards compatible with clients conforming to the older
   specification should do the following: if a DAV:sync-level element is
   not present in the request body, use the Depth header value as the
   equivalent value for the missing DAV:sync-level element.

Appendix B. Example of a Client Synchronization Approach

   This appendix gives an example of how a client might accomplish
   collection synchronization using the WebDAV sync report defined in
   this specification. Note that this is provided purely as an example,
   and is not meant to be treated as a normative "algorithm" for client
   synchronization.

   This example assumes a WebDAV client interacting with a WebDAV server
   supporting the sync report. The client keeps a local cache of
   resources in a targeted collection, "/collection/". Local changes
   are assumed to not occur. The client is only tracking changes to the
   immediate children of the collection resource.

      ** Initial State **

      The client starts out with an empty local cache.

      The client starts out with no DAV:sync-token stored for
      "/collection/".

      ** Initial Synchronization **

      The client issues a sync report request to the server with an
      empty DAV:sync-token element, and DAV:sync-level set to "1". The
      request asks for the server to return the DAV:getetag WebDAV
      property for each resource it reports.

The server returns a response containing the list of current
resources (with their associated DAV:getetag properties) as well
as a new DAV:sync-token value.

The client associates the new DAV:sync-token value with the
collection.

For each reported resource, the client creates a set of (resource
path, DAV:getetag) tuples.

For each tuple, the client issues an HTTP GET request to the
server to retrieve its content, and updates the (resource path,
DAV:getetag) entry in its local cache for that resource with the
ETag response header value returned in the GET request.

** Routine Synchronization **

The client issues a sync report request to the server with the
DAV:sync-token set to the current cached value from the last sync,
and DAV:sync-level set to "1". The request asks for the server to
return the DAV:getetag WebDAV property for each resource it
reports.

The server returns a response containing the list of changes as
well as a new DAV:sync-token value.

The client associates the new DAV:sync-token value with the
collection.

  * Process Removed Resources *

For each resource reported with a 404 response status, the client
removes the corresponding resource from its local cache.

  * Process Resources *

For each remaining reported resource, the client creates a new set
of (resource path, DAV:getetag) tuples.

The client then determines which resources are in the new set but
not in the current cache, and which resources are in the new set
and the current cache but have a different DAV:getetag value. For
each of those, the client issues an HTTP GET request to the server
to retrieve the resource content, and updates the (resource path,
DAV:getetag) entry in its local cache for that resource with the
ETag response header value returned in the GET request.

Authors' Addresses

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

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

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

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