WebDAV Ordered Collections Protocol

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

To view the list Internet-Draft Shadow Directories, see http://www.ietf.org/shadow.html.

Distribution of this document is unlimited. Please send comments to the Distributed Authoring and Versioning (WebDAV) working group at <w3c-dist-auth@w3.org>, which may be joined by sending a message with subject "subscribe" to <w3c-dist-auth-request@w3.org>.

Discussions of the WEBDAV working group are archived at URL: <http://lists.w3.org/Archives/Public/w3c-dist-auth/>.

Abstract

The WebDAV Distributed Authoring Protocol provides basic support for collections, offering the ability to create and list unordered collections.

This specification is one of a group of three specifications that supplement the WebDAV Distributed Authoring Protocol to increase the power of WebDAV collections. This specification defines a protocol supporting server-side ordering of collection members. The companion specifications "WebDAV Bindings"[B] and "WebDAV Redirect Reference Resources"[RR] define two mechanisms for allowing a single resource to appear in more than one collection.

Table of Contents

1 Notational Conventions.................................2
2 Introduction..........................................3
3 Terminology........................................3
4 Overview of Ordered Collections.....................4
5 Creating an Ordered Collection.......................5
Since this document describes a set of extensions to the WebDAV Distributed Authoring Protocol [WebDAV], itself an extension to the HTTP/1.1 protocol, the augmented BNF used here to describe protocol elements is exactly the same as described in Section 2.1 of [HTTP].
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].

2 Introduction

The simple collections that the WebDAV Distributed Authoring Protocol specification supports are powerful enough to be widely useful. They provide for the hierarchical organization of resources, with mechanisms for creating and deleting collections, copying and moving them, locking them, adding members to them and removing members from them, and getting listings of their members. Delete, copy, move, list, and lock operations can be applied recursively, so that a client can operate on whole hierarchies with a single request.

This specification is one of a family of three specifications that build on the infrastructure defined in [HTTP] and [WebDAV] to extend the capabilities of collections. The companion specifications "WebDAV Bindings"[B] and "WebDAV Redirect Reference Resources"[RR] define mechanisms for allowing the same resource to appear in multiple collections. The present specification defines protocol extensions to support ordered collections.

There are many scenarios where it is useful to impose an ordering on a collection at the server, such as expressing a recommended access order, or a revision history order. Orderings may be based on property values, but they may be completely independent of any properties on the resources identified by the collection's internal member URIs. Orderings based on properties can be obtained using a search protocol, but orderings not based on properties need some other mechanism. These orderings generally need to be maintained by a human user. The ordering protocol defined here focuses on support for such human-maintained orderings, but also allows for server-maintained orderings.

The remainder of this document is structured as follows: Section 3 defines terminology that will be used throughout the specification. Section 4 provides an overview of ordered collections. Section 5 describes how to create an ordered collection, and Section 6 discusses how to set a member's position in the ordering of a collection. Section 7 explains how to change a collection ordering. Section 8 discusses listing the members of an ordered collection. Sections 9 through 12 define the status codes, headers, properties, and XML elements needed to support ordered collections. Section 13 describes capability discovery. Sections 14 through 16 discuss security, internationalization, and IANA considerations. The remaining sections provide supporting information.

3 Terminology

The terminology used here follows that in the WebDAV Distributed Authoring Protocol specification [WebDAV]. Definitions of the terms resource, Uniform Resource Identifier (URI), and Uniform Resource Locator (URL) are provided in [URI]. Definitions of the terms URI mapping, path segment, binding, collection, and internal member URI are provided in [B].

Ordered Collection
A collection for which the results from a PROPFIND request are guaranteed to be in the order specified for that collection

Unordered Collection
A collection for which the client cannot depend on the repeatability of the ordering of results from a PROPFIND request

Client-Maintained Ordering
An ordering of collection members that is maintained on the server based on client requests specifying the position of each collection member in the ordering

Server-Maintained Ordering
An ordering of collection members that is maintained automatically by the server, based on a client’s choice of ordering semantics

4 Overview of Ordered Collections

If a collection is unordered, the client cannot depend on the repeatability of the ordering of results from a PROPFIND request. By specifying an ordering for a collection, a client requires the server to follow that ordering whenever it responds to a PROPFIND request on that collection.

These server-side orderings may be client-maintained or server-maintained. For client-maintained orderings, a client must specify the position of each of the collection’s bindings in the ordering, either when the binding is added to the collection (using the Position header) or later (using the ORDERPATCH method). For server-maintained orderings, the server automatically positions each of the collection’s bindings according to the ordering semantics.

A collection that supports ordering may be ordered, but is not required to be. It is up to the client to decide whether a given collection is ordered and, if so, to specify the semantics to be used for ordering its bindings. If a collection is ordered, each of its bindings, and hence internal member URIs, MUST be in the ordering exactly once, and the ordering MUST NOT include any binding that is not contained by the collection. Only one ordering can be attached to any collection. An ordering is considered to be part of the state of a collection resource, and hence is the same across all URI mappings to the collection. Multiple orderings of the same resources can be achieved by creating multiple collections referencing those resources, and attaching a different ordering to each collection.

The server is responsible for enforcing these constraints on orderings. The server MUST remove a binding (and its derived internal member URI) from the ordering when it is removed from the collection. The server MUST add a binding (and its derived internal member URI) to the ordering when it is added to the collection.

5 Creating an Ordered Collection

5.1 Overview

When a collection is created, the client MAY request that it be ordered and specify the semantics of the ordering by using the new Ordered
header (defined in Section 10.1) with a MKCOL request.

For collections that are ordered, the client SHOULD identify the semantics of the ordering with a URI in the Ordered header. This URI may identify a server-maintained ordering. Clients can discover the available server-maintained orderings using the mechanism defined in Section 13.2. The URI may identify a semantics for a client-maintained ordering, providing the information a human user or software package needs to insert new collection members into the ordering intelligently. Although the URI in the Ordered header MAY point to a resource that contains a definition of the semantics of the ordering, clients are discouraged from accessing that resource, in order to avoid overburdening its server. The client MAY set the header value to DAV:custom to indicate that the collection is ordered, but the semantics of the ordering are not being advertised. If the client does not want the collection to be ordered, it may omit the Ordered header, or use it with the value DAV:unordered.

If the server does not recognize the value of the Ordered header as one of its server-maintained orderings, it MUST assume that a client-maintained ordering is intended. If the value of the Ordered header is one of the server-maintained orderings that the server supports, it MUST maintain the collection’s ordering according to that ordering semantics as new members are added.

Every collection MUST have a DAV:orderingtype property (defined in Section 11.1), which indicates whether the collection is ordered and, if so, identifies the semantics of the ordering. The server sets the initial value of this property based on the value of the Ordering header in the MKCOL request. If the collection is unordered, the DAV:orderingtype property MUST have the value DAV:unordered. An ordering-aware client interacting with an ordering-unaware server (e.g., one that is implemented only according to [WebDAV]) SHOULD assume that if a collection does not have the DAV:orderingtype property, the collection is unordered.

5.2 Example: Creating an Ordered Collection

>>Request:

MKCOL /theNorth/ HTTP/1.1
Host: www.server.org
Ordered: <http://www.server.org/orderings/compass.html>

>>Response:

HTTP/1.1 201 Created

In this example a new, ordered collection was created. Its

DAV:orderingtype property has as its value the URI from the Ordered header, http://www.server.org/orderings/compass.html. In this case, the URI identifies the semantics governing a client-maintained ordering. As new members are added to the collection, clients or end users can use the semantics to determine where to position the new members in the ordering.

6 Setting the Position of a Collection Member
6.1 Overview

When a new member is added to a collection with a client-maintained ordering (for example, with PUT, MKREF, or MKCOL), its position in the ordering can be set with the new Position header (defined in Section 10.2). The Position header allows the client to specify that the member should be first in the collection’s ordering, last in the collection’s ordering, immediately before some other binding in the collection’s ordering, or immediately after some other binding in the collection’s ordering.

6.2 Status Codes

409 (Conflict): The request specifies a position that is before or after a URI that is not an internal member URI of the collection, or before or after itself.

418 (Unordered Collection): The request specifies a collection position in an unordered collection or in a collection with a server-maintained ordering.

6.3 Examples: Setting the Position of a Collection Member

>> Request:

MKREF /~whitehead/dav/spec08.ref HTTP/1.1
HOST: www.ics.uci.edu
Ref-Target: <http://www.ics.uci.edu/i-d/draft-webdav-protocol-08.txt>
Position: after <requirements.html>

>> Response:

HTTP/1.1 201 Created

This request resulted in the creation of a new referential resource at www.ics.uci.edu/~whitehead/dav/spec08.ref, which points to the resource identified by the Ref-Target header. The Position header in this example caused the server to set its position in the ordering of the /~whitehead/dav/ collection immediately after requirements.html.

>> Request:

MOVE /i-d/draft-webdav-protocol-08.txt HTTP/1.1
Host: www.ics.uci.edu
Destination: http://www.ics.uci.edu/~whitehead/dav/draft-webdav-protocol-08.txt
Position: first

>> Response:

HTTP/1.1 418 Unordered Collection

In this case, the server returned a 418 (Unordered Collection) status code because the /~whitehead/dav/ collection is an unordered collection. Consequently, the server was unable to satisfy the Position header.
7 Changing a Collection Ordering

7.1 ORDERPATCH Method

The ORDERPATCH method is used to change the ordering semantics of a collection or to change the order of bindings in a client-maintained ordering or both.

The ORDERPATCH method changes the ordering semantics of the collection identified by the Request-URI, based on the value of DAV:orderingtype submitted in the request entity body. If the new value identifies a client-maintained ordering, the client is responsible for updating the collection’s ordering according to the new semantics. If it identifies a server-maintained ordering, the server MUST reorder the collection according to the new semantics.

The ORDERPATCH method alters the ordering of bindings in the collection identified by the Request-URI, based on instructions in the ordermember XML elements in the request entity body. The ordermember XML elements identify the bindings whose positions are to be changed, and describes their new positions in the ordering. Each new position can be specified as first in the ordering, last in the ordering, immediately before some other binding, or immediately after some other binding.

The server MUST apply the changes in the order they appear in the order XML element. The server MUST either apply all the changes or apply none of them. If any error occurs during processing, all executed changes MUST be undone and a proper error result returned.

7.1.1 Status Codes

Since multiple changes can be requested in a single ORDERPATCH request, the server MUST return a 207 (Multi-Status) response, as defined in [WebDAV].

The following are examples of response codes one would expect to be used in a 207 (Multi-Status) response for this method:

200 (OK): The change in ordering was successfully made.

409 (Conflict): The request specifies a position that is before or after a URI that is not an internal member URI of the collection, or before or after itself.

418 (Unordered Collection): The request specifies a collection position in an unordered collection or in a collection with a server-maintained ordering.

A request to reposition a binding at the same place in the ordering is not an error.

7.1.2 Example: Changing a Collection Ordering

Consider a collection /coll-1/ whose DAV:orderingtype is DAV:unordered, with bindings ordered as follows: nunavut.map
nunavut.img
baffin.map
baffin.desc
baffin.img
iqaluit.map
nunavut.desc
iqaluit.img
iqaluit.desc

>> Request:

ORDERPATCH /coll-1/ HTTP/1.1
Host: www.nunanet.com
Content-Type: text/xml
Content-Length: xxx

<?xml version="1.0" ?>
<d:order xmlns:d="DAV:">
  <d:orderingtype>
  </d:orderingtype>
  <d:ordermember>
    <d:href>nunavut.desc</d:href>
    <d:position>
      <d:after>
        <d:href>nunavut.map</d:href>
      </d:after>
    </d:position>
  </d:ordermember>
  <d:ordermember>
    <d:href>iqaluit.img</d:href>
    <d:position>
      <d:last/>
    </d:position>
  </d:ordermember>
</d:order>

>> Response:

HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: xxx

<?xml version="1.0" ?>
<d:multistatus xmlns:d="DAV:">
  <d:response>
    <d:propstat>
      <d:prop>
        <d:orderingtype/>
      </d:prop>
      <d:status>HTTP/1.1 200 OK</d:status>
    </d:propstat>
  </d:response>
  <d:response>
    <d:status>HTTP/1.1 200 OK</d:status>
  </d:response>
  <d:response>
    <d:status>HTTP/1.1 200 OK</d:status>
  </d:response>
</d:multistatus>
In this example, after the request has been processed, the previously unordered collection has become an ordered collection whose ordering semantics are identified by the URI http://www.nunanet.com/geog.ord. The value of the collection’s DAV:orderingtype property has been set to this URI. Since this is a client-maintained ordering, the request also contained instructions for changing the positions of the bindings in the ordering to comply with the new ordering semantics. If href elements are relative URIs, as in this example, they are interpreted relative to the collection whose ordering is being modified. After the request has been processed, the collection’s ordering is as follows:

```
nunavut.map
nunavut.desc
nunavut.img
baffin.map
baffin.desc
baffin.img
iqaluit.map
iqaluit.desc
iqaluit.img
```

7.1.3 Example: Failure of an ORDERPATCH Request

Consider a collection /coll-1/ with bindings ordered as follows:

```
nunavut.map
nunavut.img
baffin.map
baffin.desc
baffin.img
iqaluit.map
iqaluit.desc
iqaluit.img
```

>> Request:

```
ORDERPATCH /coll-1/ HTTP/1.1
Host: www.nunanet.com
Content-Type: text/xml
Content-Length: xxx

<?xml version="1.0" ?>
<d:order xmlns:d="DAV:"
  <d:href>nunavut.desc</d:href>
  <d:position>
    <d:after>
      <d:href>nunavut.map</d:href>
    </d:after>
  </d:position>
</d:ordermember>
```
In this example, the client attempted to position iqaluit.map after a binding that is not contained in the collection /coll-1/. The server responded to this client error with a 409 (Conflict) status code. Because ORDERPATCH is an atomic method, the request to reposition nunavut.desc (which would otherwise have succeeded) failed with a 424 (Failed Dependency) status code.

8 Listing the Members of an Ordered Collection

A PROPFIND request is used to retrieve a listing of the members of an ordered collection, just as it is used to retrieve a listing of the members of an unordered collection.

However, when responding to a PROPFIND on an ordered collection, the server MUST order the response elements according to the ordering defined on the collection. If a collection is unordered, the client cannot depend on the repeatability of the ordering of results from a PROPFIND request.

When responding to a PROPFIND on an ordered collection, the server SHOULD include the DAV:orderingtype property in the DAV:response element for the collection, even if the client did not explicitly request it.

8.1 Example: PROPFIND on an Ordered Collection

Suppose a PROPFIND request is submitted to the following collection, which has its members ordered according to their distance from the
equator.

/MyCollection/
  lakehazen.html
  siorapaluk.html
  iqaluit.html
  newyork.html

>> Request:

PROPFIND /MyCollection/ HTTP/1.1
Host: www.svr.com
Depth: 1
Content-Type: text/xml
Content-Length: xxxx

<?xml version="1.0" ?>
<D:propfind xmlns:D="DAV:"
  xmlns:J="http://www.svr.com/jsprops/">
  <D:prop>
    <J:latitude/>
  </D:prop>
</D:propfind>

>> Response:

HTTP/1.1 207 Multi-Status
Content-Type: text/xml
Content-Length: xxxx

<?xml version="1.0" ?>
<D:multistatus xmlns:D="DAV:"
  xmlns:J="http://www.svr.com/jsprops/">
  <D:response>
    <D:href>http://www.svr.com/MyCollection/</D:href>
    <D:propstat>
      <D:prop>
        <D:resourcetype/>
        <D:orderingtype>
          <D:href>http://www.svr.com/jslatitudedesc</D:href>
        </D:orderingtype>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
    <D:propstat>
      <D:prop>
        <J:latitude>82N</J:latitude>
      </D:prop>
      <D:status>HTTP/1.1 404 Not Found</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:propstat>
      <D:prop>
        <D:resourcetype/>
        <J:latitude>82N</J:latitude>
      </D:prop>
      <D:status>HTTP/1.1 404 Not Found</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
In this example, the server responded with a list of the collection members ordered according to their distance from the equator, as specified by the value of DAV:orderingtype. Although the client did not explicitly ask for the value of DAV:orderingtype, the server provided it as one of the collection properties, allowing the client to tell that the collection is ordered and to identify the ordering semantics.

9 Status Codes

9.1 418 Unordered Collection

The 418 (Unordered Collection) status code indicates that the client attempted to set the position of an internal collection member in an unordered collection or in a collection with a server-maintained ordering.

10 Headers

10.1 Ordered Entity Header

Ordered = "Ordered" :: ("DAV:unordered" | "DAV:custom" | Coded-url)
The Ordered header may be used with MKCOL to request that the new collection be ordered and to specify its ordering semantics. A value of "DAV:unordered" indicates that the collection is not ordered. A value of "DAV:custom" indicates that the collection is to be ordered, but the semantics of the ordering is not being advertised. Any other Coded-url value indicates that the collection is ordered, and identifies the semantics of the ordering.

10.2 Position Request Header

Position = "Position" ":" ("first" | "last" |
(("before" | "after") Generic-Coded-url))
Generic-Coded-url = "<" (absoluteURI | relativeURI)">
absoluteURI is defined in Section 3 of [URI].
relativeURI is defined in Section 5 of [URI].

The Position header may be used with any method that adds a binding to a collection with a client-maintained ordering, to tell the server where in the collection ordering to position the new binding being added to the collection. Examples of methods that add bindings to collections are BIND, PUT, COPY, MOVE, etc.

If the Generic-Coded-url is a relative URL, it is interpreted relative to the collection to which the new binding is being added.

The server MUST insert the new binding into the ordering at the location specified in the Position header, if one is present (and if the collection has a client-maintained ordering).

The "first" keyword indicates the new binding is put in the beginning position in the collection’s ordering, while "last" indicates the new binding is put in the final position in the collection’s ordering. The "before" keyword indicates the new binding is added to the collection’s ordering immediately prior to the position of the binding identified in the Generic-Coded-url. Likewise, the "after" keyword indicates the new binding is added to the collection’s ordering immediately following the position of the binding identified in the Generic-Coded-url.

If the request is replacing an existing resource, and the Position header is present, the server MUST remove the binding from its previous position, and then insert it at the requested position.

If the Position request header is not used when adding a binding to a collection with a client-maintained ordering, then:

- If the request is replacing an existing resource, the server MUST preserve the present ordering.
- If the request is adding a new binding to the collection, the server MUST append the new binding to the end of the ordering.

If an attempt is made to use the Position header on a collection that is unordered or that has a server-maintained ordering, the server MUST fail the request with a 418 (Unordered) status code.

11 Properties
11.1 orderingtype Property

Name: orderingtype
Namespace: DAV:
Purpose: Indicates whether the collection is ordered and, if so, uniquely identifies the semantics of the ordering being used. May also point to an explanation of the semantics in human and/or machine-readable form. At a minimum, this allows human users who add members to the collection to understand where to position them in the ordering. This property cannot be set using PROPPATCH. Its value can only be set by including the Ordered header with a MKCOL request or by submitting an ORDERPATCH request.

Value: The value unordered indicates that the collection is not ordered. The value custom indicates that the collection is ordered, but the semantics governing the ordering are not being advertised. If the value is an href element, it contains a URI that uniquely identifies the semantics of the collection’s ordering.

<!ELEMENT orderingtype (unordered | custom | href) >

12 XML Elements

12.1 unordered XML Element

Name: unordered
Namespace: DAV:
Purpose: A value of the DAV:orderingtype property that indicates that the collection is not ordered. That is, the client cannot depend on the repeatability of the ordering of results from a PROPFIND request.

<!ELEMENT unordered EMPTY >

12.2 custom XML Element

Name: custom
Namespace: DAV:
Purpose: A value of the DAV:orderingtype property that indicates that the collection is ordered, but the semantics of the ordering are not being advertised.

<!ELEMENT custom EMPTY >

12.3 order XML Element

Name: order
Namespace: DAV:
Purpose: For use with the new ORDERPATCH method. Describes a change to be made in a collection’s ordering semantics or in the positions of its bindings in the ordering or both.

Value: An optional identifier of an ordering semantics for the collection, followed by a list of changes to be made in the positions of the bindings in the collection’s ordering.
12.4 ordermember XML Element

Name: ordermember
Namespace: DAV:
Purpose: Occurs in the order XML element, and describes the new position of a single binding in the collection’s ordering.
Value: An href containing a binding’s path segment, and a description of its new position in the ordering. The href XML element is defined in [WebDAV], Section 11.3.

12.5 position XML Element

Name: position
Namespace: DAV:
Purpose: Occurs in the ordermember XML element. Describes the new position in a collection’s ordering of one of the bindings it contains.
Value: The new position can be described as first in the collection’s ordering, last in the collection’s ordering, immediately before some other binding, or immediately after some other binding.

12.6 first XML Element

Name: first
Namespace: DAV:
Purpose: Occurs in the position XML element. Specifies that the binding should be placed first in the collection’s ordering.

12.7 last XML Element

Name: last
Namespace: DAV:
Purpose: Occurs in the position XML element. Specifies that the binding should be placed last in the collection’s ordering.

12.8 before XML Element

Name: before
Namespace: DAV:
Purpose: Occurs in the position XML element. Specifies that the binding should be placed immediately before the binding in the enclosed href XML element in the collection’s ordering.
Value: href of the member it precedes in the ordering
12.9 after XML Element

Name: after
Namespace: DAV:
Purpose: Occurs in the position XML element. Specifies that the binding should be placed immediately after the binding in the enclosed href XML element in the collection’s ordering.
Value: href of the member it follows in the ordering

<!ELEMENT after href >

12.10 options XML Element

Name: options
Namespace: DAV:
Purpose: Used in OPTIONS requests to ask for more detailed information about capabilities than can be provided in the DAV: response header. Used in OPTIONS responses to provide that information.
Value: List of elements identifying or providing the additional information desired.

<!ELEMENT options (orderingoptions | ANY)+ >

12.11 orderingoptions XML Element

Name: orderingoptions
Namespace: DAV:
Purpose: Used in OPTIONS requests to ask for the list of server-maintained orderings that can be supported at the request-URI. Used in OPTIONS responses to provide that information. These values can be used in the Ordered header or the DAV:orderingtype property to request that a particular server-maintained ordering be applied to the collection.
Value: EMPTY on requests. On responses, it is the list of server-maintained orderings available for the request-URI.

<!ELEMENT orderingoptions ( (#PCDATA)+ | EMPTY) >

13 Capability Discovery

Sections 9.1 and 15 of [WebDAV] describe the use of compliance classes with the DAV header in responses to OPTIONS, to indicate which parts of the Web Distributed Authoring protocols the resource supports. This specification defines an OPTIONAL extension to [WebDAV]. It defines a new compliance class, called orderedcoll, for use with the DAV header in responses to OPTIONS requests. If a collection resource does support ordering, its response to an OPTIONS request MUST indicate that it does, by listing the new ORDERPATCH method as one it supports, and by listing the new orderedcoll compliance class in the DAV header.

When responding to an OPTIONS request, only a collection or a null resource can include orderedcoll in the value of the DAV header. By including orderedcoll, the resource indicates that its bindings can be ordered. It implies nothing about whether any collections identified by its internal member URIs can be ordered.
13.1 Example: Discovery of Support for Ordering

>> Request:

OPTIONS /somecollection/ HTTP/1.1
HOST: somehost.org

>> Response:

HTTP/1.1 200 OK
Date: Tue, 20 Jan 1998 20:52:29 GMT
Connection: close
Accept-Ranges: none
Allow: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, ORDERPATCH
Public: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, BIND, MKREF, ORDERPATCH
DAV: 1, 2, orderedcoll

The DAV header in the response indicates that the resource /somecollection/ is level 1 and level 2 compliant, as defined in [WebDAV]. In addition, /somecollection/ supports ordering. The Allow header indicates that ORDERPATCH requests can be submitted to /somecollection/. The Public header shows that other Request-URIs on the server support additional methods.

13.2 Additional Capabilities

Clients may need detailed information about specific areas of Web Distributed Authoring functionality. This information can be requested by sending an OPTIONS request with an XML body that includes a DAV:options element. The DAV:options element contains a list of empty elements identifying the information the client needs.

As described in Section 3, servers may offer a set of server-maintained orderings on collections. Clients can discover the list of server-maintained orderings available for the request-URI by including an empty DAV:orderingoptions element in the DAV:options element. The response will include a DAV:orderingoptions element with the list of supported server-maintained orderings. Servers SHOULD advertise the server-maintained orderings available using this mechanism.

13.3 Example: Discovery of Ordering Options

>> Request:

OPTIONS /somecollection/ HTTP/1.1
HOST: somehost.org

<?xml version="1.0" ?>
<D:options xmlns:D="DAV:"
  <D:orderingoptions/>
</D:options>

>> Response:

HTTP/1.1 200 OK
This response indicates that the resource /somecollection/ is level 1 compliant, as defined in [WebDAV]. In addition, /somecollection/ supports ordering. The client also asked for a list of the server-maintained orderings that are supported for /somecollection/. The response indicates that the orderings Xerox:author-ascending, Xerox:title-ascending, and Xerox:date-descending are supported.

### 14 Security Considerations

This section is provided to make WebDAV applications aware of the security implications of this protocol.

All of the security considerations of HTTP/1.1 and the WebDAV Distributed Authoring Protocol specification also apply to this protocol specification. In addition, ordered collections introduce a new security concern. This issue is detailed here.

#### 14.1 Denial of Service and DAV:orderingtype

There may be some risk of denial of service at sites that are advertised in the DAV:orderingtype property of collections. However, it is anticipated that widely-deployed applications will use hard-coded values for frequently-used ordering semantics rather than looking up the semantics at the location specified by DAV:orderingtype. In addition, Section 3 discourages clients from looking up the semantics at that location.

### 15 Internationalization Considerations

This specification follows the practices of [WebDAV] in encoding all human-readable content using XML [XML] and in the treatment of names. Consequently, this specification complies with the IETF Character Set Policy [Alvestrand].

WebDAV applications MUST support the character set tagging, character set encoding, and the language tagging functionality of the XML specification. This constraint ensures that the human-readable content of this specification complies with [Alvestrand].
As in [WebDAV], names in this specification fall into three categories: names of protocol elements such as methods and headers, names of XML elements, and names of properties. Naming of protocol elements follows the precedent of HTTP, using English names encoded in USASCII for methods and headers. The names of XML elements used in this specification are English names encoded in UTF-8.

For error reporting, [WebDAV] follows the convention of HTTP/1.1 status codes, including with each status code a short, English description of the code (e.g., 423 Locked). Internationalized applications will ignore this message, and display an appropriate message in the user’s language and character set.

For rationales for these decisions and advice for application implementors, see [WebDAV].

16 IANA Considerations

This document uses the namespaces defined by [WebDAV] for properties and XML elements. All other IANA considerations mentioned in [WebDAV] also apply to this document.

17 Copyright

To be supplied by the RFC Editor.

18 Intellectual Property

To be supplied by the RFC Editor.

19 Acknowledgements

This draft has benefited from thoughtful discussion by Jim Amsden, Steve Carter, Ken Coar, Ellis Cohen, Bruce Cragun, Spencer Dawkins, Mark Day, Rajiv Dulepet, David Durand, Roy Fielding, Yaron Goland, Fred Hitt, Alex Hopmann, Marcus Jager, Chris Kaler, Manoj Kasichainula, Rohit Khare, Daniel LaLiberte, Steve Martin, Larry Masinter, Jeff McAffer, Surendra Koduru Reddy, Max Rible, Sam Ruby, Bradley Sergeant, Nick Shelness, John Stracke, John Tigue, John Turner, and others.

20 References


21 Authors’ Addresses

J. Slein
Xerox Corporation
800 Phillips Road, 105-50C
Webster, NY 14580
Email: jslein@crt.xerox.com

E. J. Whitehead, Jr.
Dept. of Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
Email: ejw@ics.uci.edu

J. Davis
CourseNet Systems
170 Capp Street
San Francisco, CA 94110
Email: jrd3@alum.mit.edu

G. Clemm
Rational Software Corporation
20 Maguire Road
Lexington, MA 02173-3104
Email: gclemm@rational.com

C. Fay
FileNet Corporation
3565 Harbor Boulevard
Costa Mesa, CA 92626-1420
Email: cfay@filenet.com

J. Crawford
IBM
Email: ccjason@us.ibm.com

T. Chihaya
DataChannel, Inc.
155 108th Ave. N.E., Suite 400
Bellevue, WA 98004
22 Appendices

22.1 Appendix 1: Extensions to the WebDAV Document Type Definition

<!ELEMENT unordered EMPTY >
<!ELEMENT custom EMPTY >
<!ELEMENT order (orderingtype?, ordermember*) >
<!ELEMENT ordermember (href, position) >
<!ELEMENT position (first | last | before | after)>
<!ELEMENT first EMPTY >
<!ELEMENT last EMPTY >
<!ELEMENT before href >
<!ELEMENT after href >
<!ELEMENT options (refintegrityoptions | orderingoptions)+ >