WEBDAV Working Group J. Slein, Xerox
INTERNET DRAFT E.J. Whitehead Jr., UC Irvine
<draft-ietf-webdav-redirectref-protocol-00.txt> J. Davis, CourseNet
G. Clemm, Rational
C. Fay, FileNet
J. Crawford, IBM
T. Chihaya, DataChannel
August 20, 1999

Expires February 20, 2000

WebDAV Redirect Reference Resources

Status of this Memo

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 redirect reference resources, one mechanism for allowing a single resource to appear in more than one collection. A redirect reference resource is a resource in one collection that responds to most requests by redirecting the request (using an HTTP 1.1 302 Moved Temporarily response) to a

different resource, possibly in a different collection. [B] defines bindings, another approach to allowing a single resource to be accessed from multiple collections. [OC] provides ordered collections.

Table of Contents

1 Notational Conventions

Since this document describes a set of extensions 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]. Since this augmented BNF uses the basic production rules provided in Section 2.2 of [HTTP], these rules apply to this document as well.

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 specification [OC] defines protocol extensions to support ordered collections. The present specification and the companion specification [B] define mechanisms for allowing the same resource to appear in multiple collections. This capability is useful for several reasons:

Organizing resources into hierarchies places them into smaller groupings, known as collections, which are more easily browsed and manipulated than a flat namespace. However, hierarchies require categorization decisions that locate resources at a single location in the hierarchy, a drawback when a resource has multiple valid categories. For example, in a hierarchy of vehicle descriptions containing collections for cars and boats, a description of a combination car/boat vehicle could belong in either collection. Ideally, the description should be accessible from both.

Hierarchies also make resource sharing more difficult, since resources that have utility across many collections are still forced into a single collection. For example, the mathematics department at one university might create a collection of information on fractals that contains bindings to some local resources, but also provides access to some resources at other universities. For many reasons, it may be undesirable to make physical copies of the shared resources on the local server - to conserve disk space, to respect copyright constraints, or to make any changes in the shared resources visible automatically.

The BIND method defined in [B] provides one mechanism for allowing a single resource to appear in multiple collections. It lets clients associate a new URI with an existing resource. This URI can then be used to submit requests to the resource. Since URIs in WebDAV are hierarchical, and correspond to a hierarchy of collections in resource space, the BIND method also has the effect of adding the resource to a collection. As new URIs are associated with the resource, it appears in additional collections.

The redirect reference resources defined here are a different mechanism for allowing a single resource to appear in multiple collections. A redirect reference resource is a resource in one collection whose purpose is to forward requests to another resource (its target), usually in a different collection. In this way, it allows clients to submit requests to the target resource from another collection. It redirects most requests to the target resource using the HTTP 302 (Moved Temporarily) status code, thereby providing a form of mediated access to the target resource.

These two approaches to allowing clients to add a single resource to multiple collections have very different characteristics:

A redirect reference is a resource, and so can have properties of its own. Such information as who created the reference, when, and why can be stored on the redirect reference resource. Since redirect references are implemented using HTTP 302 responses, it generally takes two round trips to submit a request to the intended resource. Servers are not required to enforce the integrity of redirect references. Redirect references work equally well for local resources and for resources that reside on a different server from the reference.

By contrast, a BIND request does not create a new resource, but simply makes available a new URI for submitting requests to an existing resource. The new URI can be used like any other URI to submit a request to a resource. Only one round trip is needed to submit a request to the intended target. Servers are required to enforce the integrity of the relationships between the new URIs clients create and the resources associated with them. Consequently, it is unlikely that servers will support BIND requests that cross server boundaries.

3 Terminology

The terminology used here follows and extends 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].

Reference Resource
A resource whose purpose is to forward requests to another resource. Reference resources are an alternative mechanism to bindings (defined in [B]) for allowing clients to create multiple URIs that can be used to submit requests to the same resource.

Redirect Reference Resource

A resource that forwards requests to another resource using the HTTP 1.1 302 (Moved Temporarily) response mechanism. The client is aware that this type of reference resource is mediating between it and the target resource.

Non-Reference Resource
A resource that is not a reference to another resource.

Target Resource
The resource to which requests are forwarded by a reference resource.

4 Overview of Redirect Reference Resources

For most operations submitted to a redirect reference resource, the response is a 302 (Moved Temporarily), accompanied by the Resource-Type header (defined in Section 14.2 below) set to "DAV:redirectref" and the Location header set to the URI of the target resource. With this information, the client can resubmit the request to the URI of the target resource. The methods COPY (for collections containing redirect reference resources), DELETE, MOVE, and LOCK, for reasons that will be explained, are exceptions to this general behavior. These exceptional operations are applied to the reference resource itself and do not result in a 302 response.

If the client is aware that it is operating on a redirect reference resource, it can resolve the reference by retrieving the reference resource's DAV:reftarget property (defined in Section 15.1 below), whose value contains the URI of the target resource. It can then submit requests to the target resource.

A redirect reference resource is a new type of resource. To distinguish redirect reference resources from non-reference resources, a new value of the DAV:resourcetype property (defined in [WebDAV]), DAV:redirectref, is defined in Section 16.1 below.

Since a redirect reference resource is a resource, it is possible to apply methods to the reference resource rather than to its target resource. The Passthrough request header (defined in Section 14.3 below) is provided so that referencing-aware clients can control whether an operation is applied to the redirect reference resource or to its target resource. The Passthrough header can be used with most requests to redirect reference resources. This header is particularly useful with PROPFIND, to retrieve the reference resource's own properties.

5 MKREF Method

5.1 Overview of MKREF

The MKREF method creates a redirect reference resource identified by the Request-URI, whose target is identified by the REQUIRED Ref-Target header. MKREF sets the value of the REQUIRED DAV:reftarget property to the value of the Ref-Target header.

The MKREF method creates a new binding between the new redirect

reference resource and the last path segment of the Request-URI. The new binding is added to its parent collection, identified by the Request-URI minus its trailing slash (if present) and final segment.

MKREF requests MAY include an entity body. This specification does not define the action to be taken if a request entity body is present, but allows it for extensibility.

By default, if the Request-URI of the MKREF request identifies an existing resource, the request MUST fail with a 405 (Method Not Allowed) response code. This default behavior can be overridden using the Overwrite header defined in Section 9.6 of [WebDAV].

5.2 Status Codes

201 (Created): The redirect reference resource was successfully created.

400 (Bad Request): The client set an invalid value for the Ref-Target header.

405 (Method Not Allowed): A resource already exists at the Request-URI.

409 (Conflict): Several conditions may produce this response. There may be no resource at the location specified in Ref-Target, on a server that prohibits dangling reference resources. The request may be attempting to create the reference resource in a collection that does not exist.

412 (Precondition Failed): The Overwrite header is "F" or absent, and a resource already exists at the request-URI.

5.3 Example: MKREF

>> Request:

```
MKREF /~whitehead/dav/spec08.ref HTTP/1.1
Host: www.ics.uci.edu
Ref-Target: </i-d/draft-webdav-protocol-08.txt>
```

>> Response:

```
HTTP/1.1 201 Created
```

This request resulted in the creation of a new redirect reference resource at www.ics.uci.edu/~whitehead/dav/spec08.ref, which points to the resource identified by the Ref-Target header. In this example, the target resource of the referential resource is identified by the URI http://www.ics.uci.edu/~whitehead/dav/i-d/draft-webdav-protocol-08.txt. The referential resource's DAV:resourcetype property is set to DAV:redirectref. Its DAV:reftarget property is set to the value of the Ref-Target header, "/i-d/draft-webdav-protocol-08.txt".

6 Listing the Redirect Reference Resources in a Collection

A URI of a redirect reference resource can be an internal member URI of a collection just as the URI of a non-reference resource can. A listing of the internal member URIs of a collection shows all of the URIs that are internal members of the collection, whether they identify redirect reference resources or non-reference resources. That is, a WebDAV PROPFIND request on a collection resource with the Depth header set to 1 or infinity MUST return a response XML element for each member URI in the collection, whether it identifies a non-reference resource or a redirect reference resource.

For each redirect reference resource, the response element MUST contain a 302 (Moved Temporarily) status code unless a Passthrough header with the value "F" is included with the PROPFIND request. The DAV:location pseudo-property and the DAV:resourcetype property MUST be included with the 302 status code, extending the syntax of the DAV:response element that was defined in [WebDAV] as described in Section 17 below. A referencing-aware client can tell from the DAV:resourcetype property that the collection contains a redirect reference resource. The DAV:location pseudo-property contains the absolute URI of the target resource. A referencing-aware client can either use the URI value of the DAV:location pseudo-property to retrieve the properties of the target resource, or it can submit a PROPFIND to the redirect reference resource with "Passthrough: F" to retrieve its properties. It is recommended that future editors of [WebDAV] define the DAV:location pseudo-property in [WebDAV], so that non-referencing clients will also be able to use the response to retrieve the properties of the target resource.

If the Depth header is set to infinity in the PROPFIND request, the server MUST NOT follow redirect reference resources into any collections to which they may refer.

The Passthrough header (defined in Section 14.3) MAY be used with a PROPFIND request on a collection.

6.1 Example: PROPFIND on a Collection with Redirect Reference Resources

Suppose a PROPFIND request with Depth = infinity is submitted to the following collection, with the members shown here:

```
http://www.svr.com/MyCollection/
     (non-reference resource) diary.html
     (redirect reference resource) nunavut
```

>> Request:

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

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

```
   </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:collection/></D:resourcetype>
            <J:keywords>diary, interests, hobbies</J:keywords>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
   </D:response>
   <D:response>
      <D:href>http://www.svr.com/MyCollection/diary.html</D:href>
      <D:propstat>
         <D:prop>
            <D:resourcetype/>
            <J:keywords>diary, travel, family, history</J:keywords>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
   </D:response>
   <D:response>
      <D:href>http://www.svr.com/MyCollection/nunavut</D:href>
      <D:status>HTTP/1.1 302 Moved Temporarily</D:status>
      <D:prop>
         <D:location>
            <D:href>http://www.inac.gc.ca/art/inuit/</D:href>
         </D:location>
         <D:resourcetype><D:redirectref/></D:resourcetype>
      </D:prop>
   </D:response>
</D:multistatus>
```

In this example the Depth header is set to infinity, and the Passthrough header is not used. The collection contains one URI that identifies a redirect reference resource. The response element for the redirect reference resource has a status of 302 (Moved Temporarily), and includes a DAV:prop element with the DAV:location pseudo-property and the DAV:resourcetype property to allow clients to retrieve the properties of its target resource. (The response element for the redirect reference resource does not include the requested properties. The client can submit another PROPFIND request to the URI in the DAV:location pseudo-property to retrieve those properties.)

6.2 Example: PROPFIND with Passthrough: F on a Collection with Redirect Reference Resources

Suppose a PROPFIND request with Passthrough = F and Depth = infinity is submitted to the following collection, with the members shown here:

```
/MyCollection/
     (non-reference resource) diary.html
     (redirect reference resource) nunavut
```

```
>> Request:

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

<?xml version="1.0" ?>
<D:propfind xmlns:D="DAV:">
   <D:prop>
      <D:resourcetype/>
      <D:reftarget/>
   </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:">
   <D:response>
      <D:href>http://www.svr.com/MyCollection/</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> <D:reftarget/> </D:prop>
         <D:status>HTTP/1.1 404 Not Found</D:status>
      </D:propstat>
   </D:response>
<D:response>
      <D:href>http://www.svr.com/MyCollection/diary.html</D:href>
      <D:propstat>
         <D:prop>
            <D:resourcetype/>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
```

```
      </D:propstat>
      <D:propstat>
         <D:prop> <D:reftarget/> </D:prop>
         <D:status>HTTP/1.1 404 Not Found</D:status>
      </D:propstat>
   </D:response>
   <D:response>
      <D:href>http://www.svr.com/MyCollection/nunavut</D:href>
      <D:propstat>
         <D:prop>
            <D:resourcetype><D:redirectref/></D:resourcetype>
            <D:reftarget>
               <D:href>http://www.inac.gc.ca/art/inuit/</D:href>
            </D:reftarget>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
   </D:response>
</D:multistatus>
```

Since the Passthrough header has the value "F", the response shows the properties of the redirect reference resource in the collection rather than the properties of its target. The value of the Passthrough header also prevents a 302 response from being returned for the redirect reference resource.

7 Copying Redirect Reference Resources

A client's intent in performing a COPY operation is to create a new resource that is similar to the original resource and behaves like the original resource, and that can be modified without affecting the original resource. For a COPY request to a redirect reference resource, the expectation would be a 302 response that the client could use to copy the target resource. This would yield an independent resource that could be modified without affecting the original resource. For COPY requests to collections that contain redirect reference resources, the situation is less clear. There is tension between two expectations. On the one hand, the client may expect the new copy of the collection to behave like the old one (which implies having reference resources where the old one had reference resources). On the other hand, the client may expect that it will be possible to modify the resources in the new collection without affecting the resources in the old collection (which implies having copies of the target resources where the original collection had reference resources).

For a COPY request on an individual reference resource, the response MUST be a 302 (Moved Temporarily) status code, with the URI of the target resource in the Location header, and "Resource-Type: DAV:redirectref" to distinguish the response from an ordinary HTTP redirect. This is the normal behavior for redirect reference resources, allowing the client to resubmit the request to the target resource identified in the Location header. This also yields intuitively correct behavior for a COPY request to an individual reference resource. Reference-aware clients can use the Passthrough header with the value "F" to copy the redirect reference resource itself.

For COPY on a collection containing redirect reference resources, different semantics may be desirable in different scenarios. Consequently, this specification makes a fairly arbitrary choice to take the simplest path. When a COPY request is submitted to a collection containing redirect reference resources, the server MUST copy the redirect reference resources to the new collection rather than returning **302 status codes for them.** This will result in a new collection that behaves like the old one, and avoids responding with multiple 302 status codes, each of which the client would have to process separately. Reference-aware clients can force the server to respond with 302 status codes rather than copying the reference resources by using the Passthrough header with the value "T".

7.1 Example: COPY on a Redirect Reference Resource

```
>> Request:

COPY /MyCollection/tuva HTTP/1.1
Host: www.svr.com
Destination: http://www.svr.com/OtherCollection/tuva.html

>> Response:

HTTP/1.1 302 Moved Temporarily
Location: http://www.svr.com/Asia/History/tuva.html
Resource-Type: DAV:redirectref
```

In this example, the request-URI identifies a redirect reference resource whose target resource is identified by http://www.svr.com/Asia/History/tuva.html. In this case, the server responded with a 302, and provided the URL of the target resource in the Location header. The Resource-Type header indicates to a reference-aware client that this is not an HTTP 1.1 redirect, but a reference to the resource identified by the Location header. The client can now resubmit the COPY request to the target resource, producing the desired result: a duplicate of the original target resource that can be modified independently of the original.

7.2 Example: COPY on a Collection That Contains a Redirect Reference Resource

Suppose a COPY request is submitted to the following collection, with the members shown:

```
/MyCollection/
     (non-reference resource) diary.html
     (redirect reference resource) nunavut with target
                           /Someplace/nunavut.map

>> Request:

COPY /MyCollection/ HTTP/1.1
Host: www.svr.com
Destination: http://www.svr.com/OtherCollection/
```

```
>> Response:

HTTP/1.1 201 Created
```

In this case, since /MyCollection/nunavut is a redirect reference resource, the reference resource itself, and not its target resource, was copied into the new collection. So the resulting collection is as follows:

```
/OtherCollection/
      (non-reference resource) diary.html
      (redirect reference resource) nunavut with target
                                 /Someplace/nunavut.map
```

8 Deleting and Moving Redirect Reference Resources

The DELETE method is used to delete bindings to redirect reference resources. DELETE MUST affect bindings to the reference resource itself, unless "Passthrough: T" is used, in which case it generates a 302 (Moved Temporarily) response. Similarly, when a DELETE on a collection encounters a redirect reference resource in the subtree under that collection, it MUST delete bindings to the reference resource, unless "Passthrough: T" is used, in which case it generates a 302 (Moved Temporarily) response. Whether deleting an individual resource or a collection, DELETE on a redirect reference resource does not affect the target of the reference resource.

A MOVE operation on a redirect reference resource MUST move the reference resource to a different location, and MUST NOT change the location of its target resource, unless "Passthrough: T" is used, in which case a 302 (Moved Temporarily) response is generated. The DAV:reftarget property is unchanged after a MOVE. Similarly, when a MOVE on a collection encounters a redirect reference resource in the subtree under that collection, it MUST move the reference resource, and not its target, unless "Passthrough: T" is used, in which case a 302 (Moved Temporarily) response is generated.

DELETE and MOVE differ from other methods in that they do not alter the resource that is being deleted or moved, but rather the collection that contains its binding. They change the membership of that collection.

When a redirect reference resource is added to a collection, the aim is to make it look as if the target resource were a member of that collection. When the reference resource is removed from that collection, the aim is to change the membership of that collection. Membership of the target resource in any other collections, either internally or by reference, should not be affected. Consequently, DELETE and MOVE do not follow the normal rules of behavior for reference resources. Instead, they are applied by default to the reference resource itself, not to its target resource, and by default do not result in 302 status codes.

9 Locking Redirect Reference Resources

The semantics of LOCK described here resulted from balancing a set of incompatible considerations:

- o Ideally, a LOCK on a redirect reference resource should lock both the reference resource and its target resource. The owner of an exclusive write lock, for example, would be surprised if anyone else could modify the content of the target resource while he held the lock. He would also be surprised if anyone else could delete the reference to it, or replace the reference resource with one pointing to a different target resource.
- o Non-referencing clients should be able to use redirect reference resources without encountering surprising results.
- o The basic characteristics of redirect reference resources should be honored. Redirect reference resources should be simple for servers to implement. In particular, a server should never have to resolve a redirect reference. A server should not have to provide proxy capabilities in order to implement redirect references.
- o There should be consistency between the behavior of LOCK on a single redirect reference resource and the behavior of LOCK on a collection that contains redirect reference resources.
- o The behavior of all requests to redirect reference resources should be as consistent as possible. In the absence of a Passthrough header, all methods should return a 302 when sent to a redirect reference resource.
- o LOCK semantics for redirect reference resources should be consistent with the LOCK semantics defined in [WebDAV].

We have compromised the intuitive locking behavior and support for non-referencing clients in order to preserve various sorts of consistency.

The behavior of LOCK for redirect reference resources was determined by what is possible for the case of locking collections that contain redirect reference resources.

The default behavior for any operation on a redirect reference resource is that a 302 (Moved Temporarily) response will be returned, unless the Passthrough header with a value of "F" is used. However, this policy has unacceptable consequences when locking a collection that contains redirect reference resources. Since [WebDAV] requires LOCK on a collection to be an atomic operation, if a 302 response is received for any member of the collection, the entire LOCK must fail. This would make it impossible to lock any collection that contained a redirect reference resource.

To avoid this result, a LOCK with Depth > 0 on a collection MUST lock any redirect reference resources it encounters, and not return 302 responses for them, unless the Passthrough header with a value of "T" is used. Use of the Passthrough header with a value of "T" in a LOCK request on a collection will cause the entire lock to fail if a redirect reference resource is encountered.

This gives part of the expected default lock behavior without forcing the server to resolve the redirect reference or become a proxy server in cases where the target resides on a different server.

There will be no hint in any response code that there are redirect reference resources whose targets need to be locked. The client will most likely not lock any target resources until it attempts an operation on the target resource and gets a 302 response. It is possible that a non-referencing client may never realize that the reference resource's target has not been locked.

Clearly, a LOCK with Depth = infinity on a collection MUST NOT follow any redirect reference resources whose targets are collections into the target collections; it MUST NOT cause any resources in those target collections to be locked.

The behavior of LOCK for individual redirect reference resources is designed to be consistent with LOCK behavior for collections that contain redirect reference resources. By default a LOCK on a redirect reference resource MUST lock only the reference resource, not its target resource, and it MUST NOT return a 302 response. A reference-aware client can use the Passthrough header with a value of "T" to get a 302 response with the URI of the target resource in the Location header.

UNLOCK behaves as specified in [WebDAV], unlocking all resources included in the lock identified by the Lock-Token header.

9.1 Example: LOCK on a Redirect Reference Resource

>> Request:

```
LOCK /MyCollection/tuva HTTP/1.1
Host: www.svr.com
Content-Type: text/xml
Content-Length: nnnn
Authorizaton: Digest username="jas",
   realm=jas@webdav.sb.aol.com, nonce=". . . ",
   uri="/MyCollection/tuva",
   response=". . . ", opaque=". . . "

<?xml version="1.0" ?>
<D:lockinfo xmlns:D="DAV:">
   <D:lockscope><D:exclusive/></D:lockscope>
   <D:locktype><D:write/></D:locktype>
   <D:owner>
```

```
      <D:href>http://www.svr.com/~jas/contact.html</D:href>
   </D:owner>
</D:lockinfo>
```

>> Response:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<?xml version="1.0" ?>
<D:prop xmlns:D="DAV:">
   <D:lockdiscovery>
      <D:activelock>
```

```
         <D:lockscope><D:exclusive/></D:lockscope>
         <D:locktype><D:write/></D:locktype>
         <D:depth>0</D:depth>
         <D:owner>
            <D:href>http://www.svr.com/~jas/contact.html</D:href>
         </D:owner>
         <D:locktoken>
            opaquelocktoken:e71dfae-5dec-22d6-fea5-00a0c91e6be4
         </D:locktoken>
      </D:activelock>
   </D:lockdiscovery>
</D:prop>
```

The request and response look exactly as specified in [WebDAV]. In this example, the request-URI, http://www.svr.com/MyCollection/tuva, identifies a redirect reference resource, which was successfully locked. The target resource of the redirect reference resource is not locked.

9.2 Example: LOCK on a Collection That Contains a Redirect Reference Resource, with Passthrough: T

Suppose a LOCK request is submitted to the following collection, with the members shown:

```
/MyCollection/
     (non-reference resource) diary.html
     (redirect reference resource) nunavut
```

>> Request:

```
LOCK /MyCollection/ HTTP/1.1
Host: www.svr.com
Passthrough: T
```

```
Content-Type: text/xml
Content-Length: nnnn
Authorizaton: Digest username="jas",
   realm=jas@webdav.sb.aol.com, nonce=". . . ",
   uri="/MyCollection/tuva",
   response=". . . ", opaque=". . . "

<?xml version="1.0" ?>
<D:lockinfo xmlns:D="DAV:">
   <D:lockscope><D:exclusive/></D:lockscope>
   <D:locktype><D:write/></D:locktype>
   <D:owner>
      <D:href>http://www.svr.com/~jas/contact.html</D:href>
   </D:owner>
</D:lockinfo>
```

>> Response:

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

```
<?xml version="1.0" ?>
<D:multistatus xmlns:D="Dav:">
   <D:response>
      <D:href>http://www.svr.com/MyCollection/</D:href>
      <D:propstat>
         <D:prop><D:lockdiscovery/></D:prop>
         <D:status>HTTP/1.1 424 Failed Dependency</D:status>
      </D:propstat>
   </D:response>
   <D:response>
      <D:href>http://www.svr.com/MyCollection/diary.html</D:href>
      <D:status>HTTP/1.1 424 Failed Dependency</D:status>
   </D:response>
   <D:response>
      <D:href>http://www.svr.com/MyCollection/nunavut</D:href>
      <D:status>HTTP/1.1 302 Moved Temporarily</D:status>
      <D:prop>
         <D:location>
            <D:href>http://www.inac.gc.ca/art/inuit/</D:href>
         </D:location>
         <D:resourcetype><D:redirectref/></D:resourcetype>
      </D:prop>
   </D:response>
</D:multistatus>
```

The "Passthrough: T" header caused the server to return a 302 response code for the redirect reference resource in the collection. Consequently, neither the collection nor any of the resources identified by its internal member URIs were locked. A referencing-aware client can submit a separate LOCK request to the URI in the DAV:location pseudo-property returned for the redirect reference resource, and can resubmit the LOCK request with "Passthrough: F" to the collection. At that point both the reference resource and its target resource will be locked (as well as the collection and all the resources identified by its other members).

10 Other Operations on Redirect Reference Resources

Although non-referencing-aware clients cannot create reference resources, they should be able to submit requests through the reference resources created by reference-aware WebDAV clients. They should be able to follow any references to their targets. To make this possible, a server that receives a GET, HEAD, PUT, POST, OPTIONS, PROPFIND, PROPPATCH, MKCOL, MKREF, BIND, or ORDERPATCH request made via a redirect reference resource MUST return a 302 (Moved Temporarily) status code. The client and server MUST follow [HTTP] Section 10.3.3 "302 Moved Temporarily," but with these additional rules:

o The Location response header MUST contain the absolute target URI of the reference resource.

o The response MUST include the Resource-Type header. This header allows reference-aware WebDAV clients to recognize the resource as a reference resource and understand the reason for the redirection.

A reference-aware WebDAV client can act on this response in one of two ways. It can, like a non-referencing client, resubmit the request to the URI in the Location header in order to operate on the target resource. Alternatively, it can resubmit the request to the URI of the redirect reference resource with the Passthrough header set to "F" in order to operate on the reference resource itself. If the Passthrough header is present with a value of "F", the request MUST be applied to the reference resource itself, and a 302 response MUST NOT be returned.

If a reference-aware client knows before submitting its request that the request-URI identifies a redirect reference resource, and if the client wants to apply the method to the reference resource, it can save the round trip caused by the 302 response by using "Passthrough: F" in its initial request to the URI.

"Passthrough: F" can be used with GET or HEAD to retrieve the entity headers of a redirect reference resource. When "Passthrough: F" is used with GET or HEAD, the referencing entity headers (Ref-Type and Ref-Target) MUST be returned, along with all HTTP headers that make sense for reference resources (for example, Cache-Control, Age, ETag, Expires, and Last-Modified).

"Passthrough: F" can be used with PUT to replace the redirect reference resource with a non-reference resource. It can be used with OPTIONS to retrieve the capabilities of a redirect reference resource.

Clients MUST NOT, however, use "Passthrough: F" with POST. Since a reference resource cannot accept another entity as its subordinate, an attempt to POST to a reference resource with "Passthrough: F" will also fail. If a server receives a POST request with "Passthrough: F" on a redirect reference resource, it MUST fail the request with a 400 (Bad Request) status code.

Since MKCOL fails when applied to existing resources, if the client attempts to resubmit the request to the target resource, the request MUST fail (unless the reference resource is a dangling reference). Similarly, if the client attempts to resubmit the request to the reference resource with "Passthrough: F", the request MUST fail.

Since ORDERPATCH applies only to collections, an ORDERPATCH request with a Passthrough header with the value "F" on a redirect reference resource MUST fail.

10.1 Example: GET on a Redirect Reference Resource

```
>> Request:

GET /bar.html HTTP/1.1
Host: www.foo.com

>> Response:

HTTP/1.1 302 Moved Temporarily
Location: http://www.svr.com/Internet/xxspec08.html
Resource-Type: DAV:redirectref
```

Since /bar.html is a redirect reference resource and the Passthrough header is not included in the request, the response is a 302 (Moved Temporarily). The Resource-Type header informs a reference-aware client that this is not an ordinary HTTP 1.1 redirect, but is a redirect reference resource. The URI of the target resource is provided in the Location header so that the client can resubmit the request to the target resource.

10.2 Example: PUT on a Redirect Reference Resource with "Passthrough: F"

>> Request:

```
PUT /bar.html HTTP/1.1
Host: www.foo.com
Passthrough: F
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

. . . some content . . .
```

>> Response:

```
HTTP/1.1 200 OK
```

Although /bar.html is a redirect reference resource, the presence of the "Passthrough: F" header prevents a 302 response, and instead causes the request to be applied to the reference resource. The result in this case is that the reference resource is replaced by a non-reference resource having the content submitted with the request.

10.3 Example: PROPPATCH on a Redirect Reference Resource

Request:

```
PROPPATCH /bar.html HTTP/1.1
Host: www.foo.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:propertyupdate xmlns:D="DAV:"
   xmlns:Z="http://www.w3.com/standards/z39.50/">
     <D:set>
          <D:prop>
               <Z:authors>
                    <Z:Author>Jim Whitehead</Z:Author>
                    <Z:Author>Roy Fielding</Z:Author>
               </Z:authors>
          </D:prop>
     </D:set>
     <D:remove>
          <D:prop><Z:Copyright-Owner/></D:prop>
     </D:remove>
   </D:propertyupdate>
```

Response:

```
HTTP/1.1 302 Moved Temporarily
Location: http://www.svr.com/Internet/xxspec08.html
Resource-Type: DAV:redirectref
```

Since /bar.html is a redirect reference resource and the Passthrough header is not included in the request, the response is a 302 (Moved Temporarily). The Resource-Type header informs a reference-aware client that this is not an ordinary HTTP 1.1 redirect, but is a redirect reference resource. The URI of the target resource is provided in the Location header so that the client can resubmit the request to the target resource.

11 Operations on Targets of Redirect Reference Resources

Operations on targets of redirect reference resources have no effect on the reference resource.

12 Relative URIs in Ref-Target and DAV:reftarget

The URI in a Ref-Target header MAY be a relative URI. Similarly, the href in a DAV:reftarget property MAY be a relative URI. In both cases, the base URI to be used for resolving the relative URI to absolute form is the URI used in the HTTP message to identify the redirect reference resource to which the Ref-Target entity header or DAV:reftarget property belongs.

In the case of a Ref-Target header, the base URI is constructed as follows: Its scheme component is "http", its authority component is the value of the Host header in the request, and its path component is the request-URI in the request. See Section 5 of [URI] for a discussion of relative URI references and how to resolve them.

The DAV:reftarget property appears in the protocol in the context of a Multi-Status response, in a DAV:response element that contains a single DAV:href element. The value of this DAV:href element serves as the base URI for resolving a relative URI in DAV:reftarget. The value of DAV:href may itself be relative, in which case it must be resolved first in order to serve as the base URI for the relative URI in DAV:reftarget. If the DAV:href element is relative, its base URI is constructed from the scheme component "http", the value of the Host header in the request, and the request-URI.

12.1 Example: Resolving a Relative URI in Ref-Target

>> Request:

```
MKREF /north/inuvik HTTP/1.1
Host: www.somehost.edu
Ref-Target: <mapcollection/inuvik.gif>
```

```
>> Response:
```

```
HTTP/1.1 201 Created
```

In this example, the base URI is http://www.somehost.edu/north/inuvik. Then, following the rules in [URI] Section 5, the relative URI in Ref-Target resolves to the absolute URI http://www.somehost.edu/north/mapcollection/inuvik.gif.

12.2 Example: Resolving a Relative URI in DAV:reftarget

```
>> Request:

PROPFIND /geog/ HTTP/1.1
Host: www.xxsvr.com
Passthrough: F
Depth: 1
Content-Type: text/xml
Content-Length: nnn

<?xml version="1.0" ?>
<D:propfind xmlns:D="DAV:">
   <D:prop>
      <D:resourcetype/>
      <D:reftarget/>
   </D:prop>
</D:propfind>

>> Response:

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

<?xml version="1/0" ?>
<D:multistatus xmlns:D="DAV:">
   <D:response>
      <D:href>/geog/</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><D:reftarget/></D:prop>
         <D:status>HTTP/1.1 404 Not Found</D:status>
     </D:propstat>
   </D:response>
   <D:response>
      <D:href>/geog/stats.html</D:href>
      <D:propstat>
         <D:prop>
            <D:resourcetype><D:redirectref/></D:resourcetype>
            <D:reftarget><D:href>statistics/population/1997.html
                 </D:href></D:reftarget>
         </D:prop>
```

```
         <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
   </D:response>
</D:multistatus>
```

In this example, the relative URI statistics/population/1997.html is returned as the value of reftarget for the reference resource identified by href /geog/stats.html. The href is itself a relative URI, which resolves to http://www.xxsrv.com/geog/stats.html. This is the base URI for resolving the relative URI in reftarget. The absolute URI of reftarget is http://www.xxsrv.com/geog/statistics/population/1997.html.

13 Redirect References to Collections

In a Request-URI /segment1/segment2/segment3, any of the three segments may identify a redirect reference resource. (See [URI], Section 3.3, for definitions of "path" and "segment".) If any segment in a Request-URI identifies a redirect reference resource, the response is a 302. The value of the Location header in the 302 response is as follows:

The leftmost path segment of the request-URI that identifies a redirect reference resource, together with all path segments and separators to the left of it, is replaced by the value of the redirect reference resource's DAV:reftarget property (resolved to an absolute URI). The remainder of the request-URI is concatenated to this path.

Note: If the DAV:reftarget property ends with a "/" and the remainder of the Request-URI is non-empty (and therefore must begin with a "/"), the final "/" in the DAV:reftarget property is dropped before the remainder of the Request-URI is appended.

Consider Request-URI /x/y/z.html. Suppose that /x/ is a redirect reference resource whose target resource is collection /a/, which contains redirect reference resource y whose target resource is collection /b/, which contains redirect reference resource z.html whose target resource is /c/d.html.

```
/x/ -----> /a/
           /a/y/ -----> /b/
                        /b/z.html -----> /c/d.html
```

In this case the client must follow up three separate 302 responses before finally reaching the target resource. The server responds to the initial request with a 302 with Location: /a/y/z.html, and the client resubmits the request to /a/y/z.html. The server responds to this request with a 302 with Location: /b/z.html, and the client resubmits the request to /b/z.html. The server responds to this request with a **302 with Location: /c/d.html, and the client resubmits the request to** /c/d.html. This final request succeeds.

14 Headers

14.1 Ref-Target Entity Header

```
Ref-Target = "Ref-Target" ":" 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 Ref-Target header is defined primarily for use with MKREF requests to identify the target resource of the new redirect reference resource being created.

14.2 Resource-Type Entity Header

```
Resource-Type = "Resource-Type" ":" ("DAV:redirectref" |
                                      ext-resource-type)
ext-resource-type = coded-URL
```

The Resource-Type header is defined primarily for use in 302 responses, to allow reference-aware clients to distinguish between HTTP 1.1 redirects and 302 responses for redirect reference resources. The possible values of this header are DAV:redirectref, and ext-resource-type. The ext-resource-type production is provided for extensibility.

14.3 Passthrough Request Header

```
Passthrough = "Passthrough" ":" ("T" | "F")
```

The optional Passthrough header can be used on any request to a redirect reference resource. If the Passthrough header has the value "F", the request MUST be applied to the reference resource itself, and a 302 response MUST NOT be returned. If the Passthrough header has the value "T", a 302 response MUST be returned, with the URI of the target resource in the Location header and the Resource-Type header with a value "DAV:redirectref".

If the Passthrough header is used on a request to any other sort of resource besides a reference resource, the server SHOULD ignore it. If the Passthrough header with the value "F" appears in a POST or ORDERPATCH request to a reference resource, the server MUST respond with a 400 (Bad Request).

15 Properties

15.1 reftarget Property

```
Name:        reftarget
Namespace:   DAV:
Purpose:     A property of redirect reference resources that provides an
             efficient way for clients to discover the URI of the target
             resource.  This is a read-only property, whose value can
             only be set by using the Ref-Target header with a MKREF
             request.
Value:       href containing the URI of the target resource.  This value
             MAY be a relative URI.  The reftarget property can occur in
             the entity bodies of responses to PROPFIND requests.
```

```
<!ELEMENT reftarget href >
```

15.2 location Pseudo-Property

```
Name:        location
Namespace:   DAV:
Purpose:     For use with 302 (Moved Temporarily) response codes in
             Multi-Status responses.  It contains the absolute URI of the
             temporary location of the resource.  In the context of
             redirect reference resources, this value is the absolute URI
             of the target resource.  It is analogous to the Location
             header in HTTP 302 responses defined in [HTTP] Section
             10.3.3 "302 Moved Temporarily."  Including the location
             pseudo-property in a Multi-Status response requires an
             extension to the syntax of the DAV:response element defined
             in [WebDAV], which is defined in Section 17 below.  This
             pseudo-property is not expected to be stored on the
             reference resource. It is modeled as a property only so that
             it can be returned inside a DAV:prop element in a Multi-
             Status response.
Value:       href containing the absolute URI of the target resource.
```

```
<!ELEMENT location href >
```

16 XML Elements

16.1 redirectref XML Element

```
Name:       redirectref
Namespace:  DAV:
Purpose:    Used as the value of the DAV:resourcetype property to
            specify that the resource type is a redirect reference
            resource.
```

```
<!ELEMENT redirectref EMPTY >
```

17 Extensions to the DAV:response XML Element for Multi-Status Responses

As described in Sections 6 and 9, the DAV:location pseudo-property and the DAV:reftype property may be returned in the DAV:response element of a 207 Multi-Status response, to allow clients to resubmit their requests to the target resource of a redirect reference resource.

Whenever these properties are included in a Multi-Status response, they are placed in a DAV:prop element associated with the href to which they apply. This structure provides a framework for future extensions by other standards that may need to include additional properties in their responses.

Consequently, the definition of the DAV:response XML element changes to the following:

```
<!ELEMENT response (href, ((href*, status, prop?) | (propstat+)),
responsedescription?) >
```

18 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 redirectrefs, for use with the DAV header in responses to OPTIONS requests. If a resource does support redirect references, its response to an OPTIONS request MUST indicate that it does, by listing the new MKREF method as one it supports, and by listing the new redirectrefs compliance class in the DAV header.

When responding to an OPTIONS request, any type of resource can include redirectrefs in the value of the DAV header. Doing so indicates that the server permits a redirect reference resource at the request URI.

18.1 Example: Discovery of Support for Redirect Reference Resources

```
>> Request:

OPTIONS /somecollection/someresource 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, MKREF
Public: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, COPY, MOVE, MKCOL,
PROPFIND, PROPPATCH, LOCK, UNLOCK, BIND, MKREF, ORDERPATCH
DAV: 1, 2, redirectrefs
```

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

19 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, redirect reference resources introduce several new security concerns and increase the risk of some existing threats. These issues are detailed below.

19.1 Privacy Concerns

By creating redirect reference resources on a trusted server, it is

possible for a hostile agent to induce users to send private information to a target on a different server. This risk is mitigated somewhat, since clients are required to notify the user of the redirection for any request other than GET or HEAD. (See [HTTP], Section 10.3.3 Moved Temporarily.)

19.2 Redirect Loops

Although redirect loops were already possible in HTTP 1.1, the introduction of the MKREF method creates a new avenue for clients to create loops accidentally or maliciously. If the reference resource and its target are on the same server, the server may be able to detect MKREF requests that would create loops. See also [HTTP], Section 10.3 "Redirection 3xx."

19.3 Redirect Reference Resources and Denial of Service

Denial of service attacks were already possible by posting URLs that were intended for limited use at heavily used Web sites. The introduction of MKREF creates a new avenue for similar denial of service attacks. Clients can now create redirect reference resources at heavily used sites to target locations that were not designed for heavy usage.

19.4 Private Locations May Be Revealed

There are several ways that redirect reference resources may reveal information about directory structures. First, the DAV:reftarget property of every redirect reference resource contains the URI of the target resource. Anyone who has access to the reference resource can discover the directory path that leads to the target resource. The owner of the target resource may have wanted to limit knowledge of this directory structure.

Sufficiently powerful access control mechanisms can control this risk to some extent. Property-level access control could prevent users from examining the DAV:reftarget property. (The Ref-Target and Location headers, which are returned in some responses to requests on redirect reference resources, reveal the same information, however.) In some environments, the owner of a resource might be able to use access control to prevent others from creating references to that resource.

20 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].

21 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.

22 Copyright

To be supplied by the RFC Editor.

23 Intellectual Property

To be supplied by the RFC Editor.

24 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.

25 References

[URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax." RFC 2396. MIT/LCS, U.C. Irvine, Xerox. August, 1998.

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels." RFC 2119, BCP 14. Harvard University. March, 1997.

[XML] T. Bray, J. Paoli, C.M. Sperberg-McQueen, "Extensible Markup Language (XML)." World Wide Web Consortium Recommendation REC-xml-**19980210**. **http://www.w3.org/TR/1998/REC-xml-19980210**.

[HTTP] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1." RFC **2616**. **UC Irvine, Compaq, W3C, Xerox, Microsoft.** June, 1999.

[WebDAV] Y. Y. Goland, E. J. Whitehead, Jr., A. Faizi, S. R. Carter, D. Jensen, "HTTP Extensions for Distributed Authoring - WebDAV." RFC 2518. Microsoft, U.C. Irvine, Netscape, Novell. February, 1999.

[B] J. Slein, E.J. Whitehead Jr., J. Davis, G. Clemm, C. Fay, J. Crawford, T. Chihaya, "WebDAV Bindings." Internet Draft (work in progress) draft-ietf-webdav-binding-protocol-00. Xerox, UC Irvine, CourseNet, Rational, FileNet, IBM, DataChannel. August, 1999.

[OC] J. Slein, E.J. Whitehead Jr., J. Davis, G. Clemm, C. Fay, J. Crawford, T. Chihaya, "WebDAV Ordered Collections." Internet Draft (work in progress) draft-ietf-webdav-ordering-protocol-00. Xerox, UC Irvine, CourseNet, Rational, FileNet, IBM, DataChannel. August, 1999.

26 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
Email: Tyson@DataChannel.com

27 Appendices

27.1 Appendix 1: Extensions to the WebDAV Document Type Definition

```
<!--============= XML Elements from Section 16 ================-->
<!ELEMENT redirectref EMPTY >
<!--============= Property Elements from Section 15 =================-->
<!ELEMENT reftarget href>
<!ELEMENT location href>
<!--====== Changes to the DAV:response Element from Section 17 ====-->
<!ELEMENT response (href, ((href*, status, prop?) | (propstat+)),
responsedescription?) >
```