INTERNET-DRAFT
draft-ietf-webdav-acl-06

Geoffrey Clemm, Rational Software
Anne Hopkins, Microsoft Corporation
Eric Sedlar, Oracle Corporation
Jim Whitehead, U.C. Santa Cruz

Expires December 21, 2001

June 21, 2001

WebDAV Access Control 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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

This document specifies a set of methods, headers, and message bodies that define Access Control extensions to the WebDAV Distributed Authoring Protocol. This protocol permits a client to remotely read and modify access control lists that instruct a server whether to grant or deny operations upon a resource (such as HTTP method invocations) by a given principal.

This document is a product of the Web Distributed Authoring and Versioning (WebDAV) working group of the Internet Engineering Task Force. Comments on this draft are welcomed, and should be addressed to the acl@webdav.org mailing list. Other related documents can be found at http://www.webdav.org/acl/, and http://www.ics.uci.edu/pub/ietf/webdav/.

Table of Contents

1 INTRODUCTION

The goal of the WebDAV access control extensions is to provide an interoperable mechanism for handling discretionary access control for content in WebDAV servers. WebDAV access control can be implemented on content repositories with security as simple as that of a UNIX file system, as well as more sophisticated models. The underlying principle of access control is that who you are determines how you can access a resource. The "who you are" is defined by a "principal" identifier; users, client software, servers, and groups of the previous have principal identifiers. The "how" is determined by a single "access control list" (ACL) associated with a resource. An ACL contains a set of "access control entries" (ACEs), where each ACE specifies a principal and a set of privileges that are either granted or denied to that principal. When a principal submits an operation (such as an HTTP or WebDAV method) to a resource for execution, the server evaluates the ACEs in the ACL to determine if the principal has permission for that operation.

This specification intentionally omits discussion of authentication, as the HTTP protocol already has a number of authentication mechanisms [RFC2617]. Some authentication mechanism (such as HTTP Digest Authentication, which all WebDAV compliant implementations are required to support) must be available to validate the identity of a principal.

The following issues are out of scope for this document:

* Access control that applies only to a particular property on a resource (excepting the access control properties DAV:acl and DAV:current-user-privilege-set), rather than the entire resource,

* Role-based security (where a role can be seen as a dynamically defined collection of principals),

* Specification of the ways an ACL on a resource is initialized,

* Specification of an ACL that applies globally to all resources , rather than to a particular resource.

* Creation and maintenance of resources representing people or computational agents (principals), and groups of these.

This specification is organized as follows. Section 1.1 defines key concepts used throughout the specification, and is followed

by more in-depth discussion of principals (Section 2), and privileges (Section 3). Properties defined on principals are specified in Section 4, and access control properties for content resources are specified in Section 5. The semantics of access control lists are described in Section 6, including sections on ACE combination (Section 6.1), ACE ordering

(Section 6.2), and principals required to be present in an ACE (Section 6.4). Client discovery of access control capability using OPTIONS is described in Section 7.1, and the access control setting method, ACL, is specified in Section 8. Internationalization considerations (Section 11) and security considerations (Section 12) round out the specification. An appendix (Section 19.1) provides an XML Document Type Definition (DTD) for the XML elements defined in the specification.

1.1 Terms

This draft uses the terms defined in HTTP [RFC2616] and WebDAV [RFC2518]. In addition, the following terms are defined:

principal

A "principal" is a distinct human or computational actor that initiates access to network resources. In this protocol, a principal is an HTTP resource that represents such an actor.

principal collection

A "principal collection" is a group of principals, and is represented in this protocol by a WebDAV collection containing HTTP resources that represent principals, and principal collections.

privilege

A "privilege" controls access to a particular set of HTTP operations on a resource.

aggregate privilege

An "aggregate privilege" is a privilege that contains a set of other privileges.

abstract privilege

The modifier "abstract", when applied to a privilege, means the privilege cannot be set in an access control element (ace).

access control list (ACL)

An "ACL" is a list of access control elements that define access control to a particular resource.

access control element (ace)

An "ace" either grants or denies a particular set of (non-abstract) privileges for a particular principal.

inherited ace

An "inherited ace" is an ace that is dynamically shared from the ACL of another resource. When a shared ACE changes on the primary resource, it is also changed on inheriting resources.

protected property

A "protected property" is one whose value cannot be updated except by a method explicitly defined as updating that specific property. In particular, a protected property cannot be updated with a PROPPATCH request.

1.2 Notational Conventions

The augmented BNF used by this document to describe protocol elements is described in Section 2.1 of [RFC2616]. Because this augmented BNF uses the basic production rules provided in Section 2.2 of [RFC2616], those 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].

Definitions of XML elements in this document use XML element type declarations (as found in XML Document Type Declarations), described in Section 3.2 of [REC-XML].

2 PRINCIPALS

A principal is a network resource that represents a distinct human or computational actor that initiates access to network resources. On many implementations, users and groups are represented as principals; other types of principals are also possible. A URI of any scheme MAY be used to identify a principal resource. However, servers implementing this specification MUST expose principal resources at an http(s) URL, which is a privileged scheme that points to resources that have additional properties, as described in Section 4. So, a principal resource can have multiple URI identifiers, one of which has to be an http(s) scheme URL. Although an implementation SHOULD support PROPFIND and MAY support PROPPATCH to access and modify information about a principal, it is not required to do so.

A principal resource may or may not be a collection. If a person or computational agent matches a principal resource that is contained by a collection principal, they also match the

collection principal. This definition is recursive, and hence if a person or computational agent matches a collection principal that is the child of another collection principal, they also match the parent collection principal. Membership in a collection principal is also recursive, so a principal in a collection principal GRPA contained by collection principal

GRPB is a member of both GRPA and GRPB. Implementations not supporting recursive membership in principal collections can return an error if the client attempts to bind collection principals into other collection principals.

Servers that support aggregation of principals (e.g. groups of users or other groups) MUST manifest them as collection principals. At minimum, principals and collection principals MUST support the OPTIONS and PROPFIND methods.

> Implementer's Note: Collection principals are first and foremost WebDAV collections. Therefore they contain resources as members. Since there is no requirement that all members of a collection principal need be principals, it is possible for a collection principal to have non-principals as members. When enumerating the principals-only membership of a collection principal, it is necessary to retrieve the DAV:resourcetype property and check it for the DAV:principal XML element (described in Section 4). If the DAV:principal XML element is not present, the resource is not a principal and may be ignored for the purposes of determining the principals-only membership of the collection principal.
>
> For example, the collection principal /FOO/ has two members, Bar and Baz. Bar is a principal but Baz is not. Therefore when determining which principals belong to the collection principal /FOO/, a client would enumerate the membership using PROPFIND while asking for the DAV:resourcetype property, and see that only Bar has the DAV:principal XML element. Therefore, only Bar is the only principal that is a member of the collection principal /FOO/.

3 PRIVILEGES

Ability to perform a given method on a resource SHOULD be controlled by one or more privileges. Authors of protocol extensions that define new HTTP methods SHOULD specify which privileges (by defining new privileges, or mapping to ones below) are required to perform the method. A principal with no privileges to a resource SHOULD be denied any HTTP access to that resource.

Privileges may be containers of other privileges, in which case they are termed aggregate privileges. If a principal is granted or denied an aggregate privilege, it is semantically equivalent to granting or denying each of the aggregated privileges individually. For example, an implementation may define add-member and remove-member privileges that control the

ability to add and remove an internal member of a collection. Since these privileges control the ability to update the state of a collection, these privileges would be aggregated by the DAV:write privilege on a collection, and granting the DAV:write privilege on a collection would also grant the add-member and remove-member privileges.

Privileges may have the quality of being abstract, in which case they cannot be set in an ACE. Aggregate and non-aggregate privileges are both capable of being abstract. Abstract privileges are useful for modeling privileges that otherwise would not be exposed via the protocol. Abstract privileges also provide server implementations with flexibility in implementing the privileges defined in this specification. For example, if a server is incapable of separating the read resource capability from the read ACL capability, it can still model the DAV:read and DAV:read-acl privileges defined in this specification by declaring them abstract, and containing them within a non-abstract aggregate privilege (say, read-all) that holds DAV:read, and DAV:read-acl. In this way, it is possible to set the aggregate privilege, read-all, thus coupling the setting of DAV:read and DAV:read-acl, but it is not possible to set DAV:read, or DAV:read-acl individually. Since aggregate privileges can be abstract, it is also possible to use abstract privileges to group or organize non-abstract privileges. Privilege containment loops are not allowed, hence a privilege MUST NOT contain itself. For example, DAV:read cannot contain DAV:read.

The set of privileges that apply to a particular resource may vary with the DAV:resourcetype of the resource, as well as between different server implementations. To promote interoperability, however, this specification defines a set of well-known privileges (e.g. DAV:read,DAV:write, DAV:read-acl, DAV:write-acl, DAV:read-current-user-privilege-set, and DAV:all), which can at least be used to classify the other privileges defined on a particular resource. The access permissions on null and lock-null resources (defined in [RFC2518], Sections 3 and 7.4) are solely those they inherit (if any), and they are not discoverable (i.e., the access control properties specified in Section 5 are not defined on null and lock-null resources). On the transition from null or lock-null to a stateful resource, the initial access control list is set by the server's default ACL value policy (if any).

3.1 DAV:read Privilege

The read privilege controls methods that return information about the state of the resource, including the resource's properties. Affected methods include GET and PROPFIND. Additionally, the read privilege MAY control the OPTIONS method.

```
<!ELEMENT read EMPTY>
```

3.2 DAV:write Privilege

The write privilege controls methods that modify the content, dead properties, or (in the case of a collection) membership of the resource, such as PUT and PROPPATCH. Note that state modification is also controlled via locking (see section 5.3 of

[WEBDAV]), so effective write access requires that both write privileges and write locking requirements are satisfied.

<!ELEMENT write EMPTY>

3.3 DAV:read-acl Privilege

The DAV:read-acl privilege controls the use of PROPFIND to retrieve the DAV:acl property of the resource.

<!ELEMENT read-acl EMPTY>

3.4 DAV:read-current-user-privilege-set Privilege

The DAV:read-current-user-privilege-set privilege controls the use of PROPFIND to retrieve the DAV:current-user-privilege-set property of the resource.

Clients are intended to use this property to visually indicate in their UI items that are dependent on the permissions of a resource, for example, by graying out resources that are not writeable.

This privilege is separate from DAV:read-acl because there is a need to allow most users access to the privileges permitted the current user (due to its use in creating the UI), while the full ACL contains information that may not be appropriate for the current authenticated user. As a result, the set of users who can view the full ACL is expected to be much smaller than those who can read the current user privilege set, and hence distinct privileges are needed for each.

<!ELEMENT read-current-user-privilege-set EMPTY>

3.5 DAV:write-acl Privilege

The DAV:write-acl privilege controls use of the ACL method to modify the DAV:acl property of the resource.

<!ELEMENT write-acl EMPTY>

3.6 DAV:all Privilege

DAV:all is an aggregate privilege that contains the entire set of privileges that apply to the resource.

<!ELEMENT all EMPTY>

3.7 Aggregation of Predefined Privileges

Server implementations are free to aggregate the predefined privileges (defined above in Sections 3.1-3.6) subject to the following limitations:

DAV:read-acl MUST NOT contain DAV:read, DAV:write, DAV:write-acl, or DAV:read-current-user-privilege-set.

DAV:write-acl MUST NOT contain DAV:write, DAV:read, DAV:read-acl, or DAV:read-current-user-privilege-set.

DAV:read-current-user-privilege-set MUST NOT contain DAV:write, DAV:read, DAV:read-acl, or DAV:write-acl.

DAV:write MUST NOT contain DAV:read, DAV:read-acl, or DAV:read-current-user-privilege-set.

DAV:read MUST NOT contain DAV:write, or DAV:write-acl.

4 PRINCIPAL PROPERTIES

Principals are manifested to clients as an HTTP resource, identified by a URL. A principal MUST have a DAV:displayname property (defined in Section 13.2 of [RFC2518]), and a DAV:resourcetype property (defined in Section 13.9 of [RFC2518]). Additionally, a principal MUST report the DAV:principal empty XML element in the value of the DAV:resourcetype property in addition to all other reported elements. For example, a collection principal would report DAV:collection and DAV:principal elements. The element type declaration for DAV:principal is:

<!ELEMENT principal EMPTY>

This protocol defines the following additional property for a principal. Since it is expensive, for many servers, to retrieve access control information, the name and value of this property SHOULD NOT be returned by a PROPFIND allprop request (as defined in Section 12.14.1 of [RFC2518]).

4.1 DAV:alternate-URL

This protected property, if non-empty, contains the URIs of network resources with additional descriptive information about the principal. This property identifies one or more additional network resources (i.e., it contains one or more URIs) that may be consulted by a client to gain additional knowledge concerning a principal. Two potential uses for this property are to store an ldap [RFC2255] or mailto [RFC2368] scheme URL. Support for this property is REQUIRED, and the value is empty if no alternate URL exists for the principal. .

<!ELEMENT alternate-URL (href*)>

5 ACCESS CONTROL PROPERTIES

This specification defines a number of new properties for WebDAV resources. Access control properties may be retrieved just like other WebDAV properties, using the PROPFIND method. Since it is expensive, for many servers, to retrieve access

control information, a PROPFIND allprop request (as defined in Section 12.14.1 of [RFC2518]) SHOULD NOT return the names and values of the properties defined in this section.

HTTP resources that support the WebDAV Access Control Protocol MUST contain the following properties. Null, and lock-null resources (described in Section 7.4 of [RFC2518]) MUST NOT contain the following properties:

5.1 DAV:owner

This protected property identifies a particular principal as being the "owner" of the resource. Since the owner of a resource often has special access control capabilities (e.g., the owner frequently has permanent DAV:write-acl privilege), clients might display the resource owner in their user interface.

```
<!ELEMENT owner (href)>
```

5.1.1 Example: Retrieving DAV:owner

This example shows a client request for the value of the DAV:owner property from a collection resource with URL http://www.webdav.org/papers/. The principal making the request is authenticated using Digest authentication. The value of DAV:owner is the URL http://www.webdav.org/_acl/users/gstein, wrapped in the DAV:href XML element.

```
>> Request <<

PROPFIND /papers/ HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="jim",
   realm="jim@webdav.org", nonce="...",
   uri="/papers/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:owner/>
</D:propfind>

>> Response <<

HTTP/1.1 207 Multi-Status
```

```
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
```

```
      <D:response>
         <D:href>http://www.webdav.org/papers/</D:href>
         <D:propstat>
            <D:status>HTTP/1.1 200 OK</D:status>
            <D:prop>
               <D:owner>
                <D:href>
                  http://www.webdav.org/_acl/users/gstein
                </D:href>
               </D:owner>
            </D:prop>
         </D:propstat>
      </D:response>
   </D:multistatus>
```

5.1.2 Example: An Attempt to Set DAV:owner

The following example shows a client request to modify the value of the DAV:owner property on the resource with URL http://www.webdav.org/papers/. Since DAV:owner is a protected property, the server responds with a 207 (Multi-Status) response that contains a 403 (Forbidden) status code for the act of setting DAV:owner. [RFC2518], Section 8.2.1 describes PROPPATCH status code information, and Section 11 describes the Multi-Status response.

>> Request <<

```
PROPPATCH /papers/ HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="jim",
   realm="jim@webdav.org", nonce="...",
   uri="/papers/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:">
   <D:set>
      <D:prop>
         <D:owner>
           <D:href>
             http://www.webdav.org/_acl/users/jim
           </D:href>
         </D:owner>
      </D:prop>
   </D:set>
```

```
</D:propertyupdate>


>> Response <<

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

```
<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
   <D:response>
      <D:href>http://www.webdav.org/papers/</D:href>
      <D:propstat>
         <D:status>HTTP/1.1 403 Forbidden</D:status>
         <D:prop><D:owner/></D:prop>
      </D:propstat>
      <D:responsedescription>Failure to set protected property
(DAV:owner)
      </D:responsedescription>
   </D:response>
</D:multistatus>
```

5.2 DAV:supported-privilege-set

This is a protected property that identifies the privileges defined for the resource.

```
<!ELEMENT supported-privilege-set (supported-privilege*)>
```

Each privilege appears as an XML element, where aggregate privileges list as sub-elements all of the privileges that they aggregate.

```
<!ELEMENT supported-privilege
 (privilege, abstract?, description, supported-privilege*)>
<!ELEMENT privilege ANY>
```

An abstract privilege of a resource MUST NOT be used in an ACE for that resource. Servers MUST fail an attempt to set an abstract privilege.

```
<!ELEMENT abstract EMPTY>
```

A description is a human-readable description of what this privilege controls access to.

```
<!ELEMENT description #PCDATA>
```

It is envisioned that a WebDAV ACL-aware administrative client would list the supported privileges in a dialog box, and allow the user to choose non-abstract privileges to apply in an ACE. The privileges tree is useful programmatically to map well-known privileges (defined by WebDAV or other standards groups) into privileges that are supported by any particular server implementation. The privilege tree also serves to hide

complexity in implementations allowing large number of privileges to be defined by displaying aggregates to the user.

5.2.1 Example: Retrieving a List of Privileges Supported on a Resource

This example shows a client request for the DAV:supported-privilege-set property on the resource http://www.webdav.org/papers/. The value of the DAV:supported-privilege-set property is a tree of supported privileges:

```
  DAV:all (aggregate, abstract)
      |
    +-- DAV:read (aggregate)
            |
            +-- DAV:read-acl (abstract)
            +-- DAV:read-current-user-privilege-set (abstract)
    +-- DAV:write (aggregate)
         |
            +-- DAV:write-acl (abstract)
```

This privilege tree is not normative, and many possible privilege trees are possible.

```
>> Request <<

PROPFIND /papers/ HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="gclemm",
   realm="gclemm@webdav.org", nonce="...",
   uri="/papers/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:supported-privilege-set/>
</D:propfind>

>> Response <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
```

```
          <D:href>http://www.webdav.org/papers/</D:href>
          <D:propstat>
            <D:status>HTTP/1.1 200 OK</D:status>
            <D:prop>
              <D:supported-privilege-set>
```

```
            <D:supported-privilege>
              <D:privilege> <D:all/> </D:privilege>
              <D:abstract/>
              <D:description>Any operation</D:description>
              <D:supported-privilege>
                <D:privilege> <D:read/> </D:privilege>
                <D:description>Read any object</D:description>
                <D:supported-privilege>
                  <D:privilege> <D:read-acl/> </D:privilege>
                  <D:abstract/>
                  <D:description>Read ACL</D:description>
                </D:supported-privilege>
              </D:supported-privilege>
                <D:supported-privilege>
                  <D:privilege>
                    <D:read-current-user-privilege-set/>
                  </D:privilege>
                  <D:abstract/>
                  <D:description>Read current user privilege set
    property</D:description>
              </D:supported-privilege>
              <D:supported-privilege>
                <D:privilege> <D:write/> </D:privilege>
                <D:description>Write any object</D:description>
                <D:supported-privilege>
                  <D:privilege> <D:write-acl/> </D:privilege>
                  <D:description>Write ACL</D:description>
                  <D:abstract/>
                </D:supported-privilege>
              </D:supported-privilege>
            </D:supported-privilege>
          </D:supported-privilege-set>
        </D:prop>
      </D:propstat>
    </D:response>
  </D:multistatus>
```

5.3 DAV:current-user-privilege-set

DAV:current-user-privilege-set is a protected property containing the exact set of privileges (as computed by the server) granted to the currently authenticated HTTP user. Aggregate privileges and their contained privileges are listed. A user-agent can use the value of this property to adjust its user interface to make actions inaccessible (e.g., by graying out a menu item or button) for which the current principal does not have permission. This is particularly useful for an access control user interface, which can be constructed without

knowing the ACE combining semantics of the server. This property is also useful for determining what operations the current principal can perform, without having to actually execute an operation.

```
<!ELEMENT current-user-privilege-set (privilege*)>
```

```
<!ELEMENT privilege ANY>
```

If the current user is granted a specific privilege, that privilege must belong to the set of privileges that may be set on this resource. Therefore, each element in the DAV:current-user-privilege-set property MUST identify a non-abstract privilege from the DAV:supported-privilege-set property.

5.3.1 Example: Retrieving the User's Current Set of Assigned Privileges

Continuing the example from Section 5.2.1, this example shows a client requesting the DAV:current-user-privilege-set property from the resource with URL http://www.webdav.org/papers/. The username of the principal making the request is ôkhareö, and Digest authentication is used in the request. The principal with username ôkhareö has been granted the DAV:read privilege. Since the DAV:read privilege contains the DAV:read-acl and DAV:read-current-user-privilege-set privileges (see Section 5.2.1), the principal with username ôkhareö can read the ACL property, and the DAV:current-user-privilege-set property. However, the DAV:all, DAV:read-acl, DAV:write-acl and DAV:read-current-user-privilege-set privileges are not listed in the value of DAV:current-user-privilege-set, since (for this example) they are abstract privileges. DAV:write is not listed since the principal with username ôkhareö is not listed in an ACE granting that principal write permission.

```
>> Request <<

PROPFIND /papers/ HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="khare",
   realm="khare@webdav.org", nonce="...",
   uri="/papers/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:current-user-privilege-set/>
</D:propfind>


>> Response <<

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

```
     Content-Length: xxx

     <?xml version="1.0" encoding="utf-8" ?>
     <D:multistatus xmlns:D="DAV:">
       <D:response>
```

```
    <D:href>http://www.webdav.org/papers/</D:href>
    <D:propstat>
      <D:status>HTTP/1.1 200 OK</D:status>
      <D:prop>
        <D:current-user-privilege-set>
          <D:privilege> <D:read/> </D:privilege>
        </D:current-user-privilege-set>
      </D:prop>
    </D:propstat>
  </D:response>
</D:multistatus>
```

5.4 DAV:acl

This is a protected property that specifies the list of access control entries (ACEs), which define what principals are to get what privileges for this resource.

```
<!ELEMENT acl (ace*)>
```

Each DAV:ace element specifies the set of privileges to be either granted or denied to a single principal. If the DAV:acl property is empty, no principal is granted any privilege.

```
<!ELEMENT ace (principal, (grant|deny), protected?,
inherited?)>
```

5.4.1 ACE Principal

The DAV:principal element identifies the principal to which this ACE applies.

```
<!ELEMENT principal ((href)
 | all | authenticated | unauthenticated
 | property | self)>
```

The current user matches DAV:href only if that user is authenticated as being (or being a member of) the principal identified by the URL contained by that DAV:href.

The current user always matches DAV:all.

```
<!ELEMENT all EMPTY>
```

The current user matches DAV:authenticated only if authenticated.

```
<!ELEMENT authenticated EMPTY>
```

The current user matches DAV:unauthenticated only if not authenticated.

```
<!ELEMENT unauthenticated EMPTY>
```

DAV:all is the union of DAV:authenticated, and DAV:unauthenticated. For a given request, the user matches either DAV:authenticated, or DAV:unauthenticated, but not both (that is, DAV:authenticated and DAV:unauthenticated are disjoint sets).

The current user matches a DAV:property principal in a DAV:acl property of a resource only if the value of the identified property of that resource contains at most one DAV:href XML element, the URI value of DAV:href identifies a principal, and the current user is authenticated as being (or being a member of) that principal. For example, if the DAV:property element contained <DAV:owner/>, the current user would match the DAV:property principal only if the current user is authenticated as matching the principal identified by the DAV:owner property of the resource.

```
<!ELEMENT property ANY>
```

The current user matches DAV:self in a DAV:acl property of the resource only if that resource is a principal object and the current user is authenticated as being that principal or a member of that principal collection.

```
<!ELEMENT self EMPTY>
```

5.4.2 ACE Grant and Deny

Each DAV:grant or DAV:deny element specifies the set of privileges to be either granted or denied to the specified principal. A DAV:grant or DAV:deny element of the DAV:acl of a resource MUST only contain non-abstract elements specified in the DAV:supported-privilege-set of that resource.

```
<!ELEMENT grant (privilege+)>
<!ELEMENT deny (privilege+)>
<!ELEMENT privilege ANY>
```

5.4.3 ACE Protection

If an ACE contains a DAV:protected element, an ACL request without that ACE MUST fail.

```
<!ELEMENT protected EMPTY>
```

5.4.4 ACE Inheritance

The presence of a DAV:inherited element indicates that this ACE is inherited from another resource that is identified by the URL contained in a DAV:href element. An inherited ACE cannot be modified directly, but instead the ACL on the resource from which it is inherited must be modified.

Note that ACE inheritance is not the same as ACL initialization. ACL initialization defines the ACL that a newly created resource will use (if not specified). ACE inheritance refers to an ACE that is logically shared - where an update to the resource containing an ACE will affect the ACE of each resource that inherits that ACE. The method by which ACLs are initialized or by which ACEs are inherited is not defined by this document.

```
<!ELEMENT inherited (href)>
```

5.4.5 Example: Retrieving a Resource's Access Control List

Continuing the example from Sections 5.2.1 and 5.3.1, this example shows a client requesting the DAV:acl property from the resource with URL http://www.webdav.org/papers/. There are two ACEs defined in this ACL:

ACE #1: The principal collection identified by URL http://www.webdav.org/_acl/groups/maintainers/ (the group of site maintainers) is granted DAV:write privilege. Since (for this example) DAV:write contains the DAV:write-acl privilege (see Section 5.2.1), this means the ômaintainersö group can also modify the access control list.

ACE #2: All principals (DAV:all) are granted the DAV:read privilege. Since (for this example) DAV:read contains DAV:read-acl and DAV:read-current-user-privilege-set, this means all users (including all members of the ômaintainersö group) can read the DAV:acl property and the DAV:current-user-privilege-set property.

>> Request <<

```
PROPFIND /papers/ HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="masinter",
   realm="masinter@webdav.org", nonce="...",
   uri="/papers/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:acl/>
</D:propfind>
```

```
>> Response <<

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

```
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.webdav.org/papers/</D:href>
    <D:propstat>
      <D:status>HTTP/1.1 200 OK</D:status>
      <D:prop>
        <D:acl>
          <D:ace>
            <D:principal>
              <D:href>
                http://www.webdav.org/_acl/groups/maintainers/
              </D:href>
            </D:principal>
            <D:grant>
              <D:privilege> <D:write/> </D:privilege>
            </D:grant>
          </D:ace>
          <D:ace>
            <D:principal>
              <D:href> <D:all/> </D:href>
            </D:principal>
            <D:grant>
              <D:privilege> <D:read/> </D:privilege>
            </D:grant>
          </D:ace>
        </D:acl>
      </D:prop>
    </D:propstat>
  </D:response>
</D:multistatus>
```

5.5 DAV:acl-semantics

This is a protected property that defines the ACL semantics. These semantics define how multiple ACEs that match the current user are combined, what are the constraints on how ACEs can be ordered, and which principals must have an ACE. A client user interface could use the value of this property to provide feedback to a human operator concerning the impact of proposed changes to an ACL. Alternately, a client can use this property to help it determine, before submitting an ACL method invocation, what ACL changes it needs to make to accomplish a specific goal (or whether that goal is even achievable on this server).

Since it is not practical to require all implementations to use the same ACL semantics, the DAV:acl-semantics property is used to identify the ACL semantics for a particular resource. The DAV:acl-semantics element is defined in Section 6.

5.5.1 Example: Retrieving DAV:acl-semantics

In this example, the client requests the value of the DAV:acl-semantics property. Digest authentication provides credentials for the principal operating the client. In this example, the ACE combination semantics are DAV:first-match, described in Section 6.1.1, the ACE ordering semantics are not specified (some value other than DAV:deny-before-grant, described in Section 6.2.1), the DAV:allowed-ace element states that only one ACE is permitted for each principal, and an ACE describing the privileges granted the DAV:all principal must exist in every ACL.

```
>> Request <<

PROPFIND /papers/ HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="srcarter",
   realm="srcarter@webdav.org", nonce="...",
   uri="/papers/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:acl-semantics/>
</D:propfind>


>> Response <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.webdav.org/papers/</D:href>
    <D:propstat>
      <D:status>HTTP/1.1 200 OK</D:status>
      <D:prop>
        <D:acl-semantics>
          <D:ace-combination>
            <D:first-match/>
          </D:ace-combination>
          <D:ace-ordering/>
```

```
                <D:allowed-ace>
                  <D:principal-only-one-ace/>
                </D:allowed-ace>
                <D:required-principal>
```

```
              <D:all/>
            </D:required-principal>
          </D:acl-semantics>
        </D:prop>
      </D:propstat>
    <D:response>
  </D:multistatus>
```

5.6 DAV:principal-collection-set

This protected property contains zero, one, or more URLs that identify a collection principal. It is expected that implementations of this protocol will typically use a relatively small number of locations in the URL namespace for principal, and collection principals. In cases where this assumption holds, the DAV:principal-collection-set property will contain a small set of URLs identifying the top of a collection hierarchy containing multiple principals and collection principals. An access control protocol user agent could use the contents of DAV:principal-collection-set to query the DAV:displayname property (specified in Section 13.2 of [RFC2518]) of all principals on that server, thereby yielding human-readable names for each principal that could be displayed in a user interface.

```
<!ELEMENT principal-collection-set (href*)>
```

Since different servers can control different parts of the URL namespace, different resources on the same host MAY have different DAV:principal-collection-set values. The collections specified in the DAV:principal-collection-set MAY be located on different hosts from the resource. The URLs in DAV:principal-collection-set SHOULD be http or https scheme URLs. For security and scalability reasons, a server MAY report only a subset of the entire set of known collection principals, and therefore clients should not assume they have retrieved an exhaustive listing. Additionally, a server MAY elect to report none of the collection principals it knows about, in which case the property value would be empty.

5.6.1 Example: Retrieving DAV:principal-collection-set

In this example, the client requests the value of the DAV:principal-collection-set property on the collection resource identified by URL http://www.webdav.org/papers/. The property contains the two URLs, http://www.webdav.org/_acl/users/ and

http://www.webdav.org/_acl/groups/, both wrapped in <DAV:href> XML elements. Digest authentication provides credentials for the principal operating the client.

The client might reasonably follow this request with two separate PROPFIND requests to retrieve the DAV:displayname

property of the members of the two collections (/_acl/users/ and /_acl_groups/). This information could be used when displaying a user interface for creating access control entries.

```
>> Request <<

PROPFIND /papers/ HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="yarong",
   realm="yarong@webdav.org", nonce="...",
   uri="/papers/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:principal-collection-set/>
</D:propfind>

>> Response <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.webdav.org/papers/</D:href>
    <D:propstat>
      <D:status>HTTP/1.1 200 OK</D:status>
      <D:prop>
        <D:principal-collection-set>
          <D:href>
            http://www.webdav.org/_acl/users/
          </D:href>
          <D:href>
            http://www.webdav.org/_acl/groups/
          </D:href>
        </D:principal-collection-set>
      </D:prop>
    </D:propstat>
  </D:response>
</D:multistatus>
```

5.7 Example: PROPFIND to retrieve access control properties

The following example shows how access control information can be retrieved by using the PROPFIND method to fetch the values of the DAV:owner, DAV:supported-privilege-set, DAV:current-user-privilege-set, and DAV:acl properties.

```
>> Request <<

PROPFIND /top/container/ HTTP/1.1
Host: www.foo.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="ejw",
   realm="users@foo.org", nonce="...",
   uri="/top/container/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:owner/>
  <D:supported-privilege-set/>
  <D:current-user-privilege-set/>
  <D:acl/>
</D:propfind>

>> Response <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus
   xmlns:D="DAV:"
   xmlns:A="http://www.webdav.org/acl/"> <D:response>
  <D:href>http://www.foo.org/top/container/</D:href>
  <D:propstat>
  <D:status>HTTP/1.1 200 OK</D:status>
  <D:prop>
    <D:owner>
      <D:href>http://www.foo.org/users/gclemm</D:href>
    </D:owner>
    <D:supported-privilege-set>
      <D:supported-privilege>
        <D:privilege> <D:all/> </D:privilege>
        <D:abstract/>
        <D:description>Any operation</D:description>
        <D:supported-privilege>
          <D:privilege> <D:read/> </D:privilege>
          <D:description>Read any object</D:description>
        </D:supported-privilege>
        <D:supported-privilege>
          <D:privilege> <D:write/> </D:privilege>
          <D:abstract/>
```

```
                <D:description>Write any object</D:description>
                <D:supported-privilege>
                  <D:privilege> <A:create/> </D:privilege>
                  <D:description>Create an object</D:description>
                </D:supported-privilege>
                <D:supported-privilege>
```

```
            <D:privilege> <A:update/> </D:privilege>
            <D:description>Update an object</D:description>
          </D:supported-privilege>
          <D:supported-privilege>
            <D:privilege> <A:delete/> </D:privilege>
            <D:description>Delete an object</D:description>
          </D:supported-privilege>
        </D:supported-privilege>
        <D:supported-privilege>
          <D:privilege> <D:read-acl/> </D:privilege>
          <D:description>Read the ACL</D:description>
        </D:supported-privilege>
        <D:supported-privilege>
          <D:privilege> <D:write-acl/> </D:privilege>
          <D:description>Write the ACL</D:description>
        </D:supported-privilege>
      </D:supported-privilege>
    </D:supported-privilege-set>
    <D:current-user-privilege-set>
      <D:privilege> <D:read/> </D:privilege>
      <D:privilege> <D:read-acl/> </D:privilege>
    </D:current-user-privilege-set>
    <D:acl>
      <D:ace>
        <D:principal>
          <D:href>http://www.foo.org/users/esedlar</D:href>
          </D:principal>
        <D:grant>
          <D:privilege> <D:read/> </D:privilege>
          <D:privilege> <D:write/> </D:privilege>
          <D:privilege> <D:read-acl/> </D:privilege> </D:grant>
      </D:ace>
      <D:ace>
        <D:principal>
          <D:href>http://www.foo.org/groups/marketing/</D:href>
        </D:principal>
        <D:deny>
          <D:privilege> <D:read/> </D:privilege> </D:deny>
      </D:ace>
      <D:ace>
        <D:principal>
          <D:property> <D:owner/> </D:property> </D:principal>
        <D:grant>
          <D:privilege> <D:read-acl/> </D:privilege>
          <D:privilege> <D:write-acl/> </D:privilege>
        </D:grant>
      </D:ace>
      <D:ace>
```

```
            <D:principal> <D:all/> </D:principal>
            <D:grant>
              <D:privilege> <D:read/> </D:privilege></D:grant>
            <D:inherited>
              <D:href>http://www.foo.org/top/</D:href>
     </D:inherited>
```

```
      </D:ace> </D:acl>
    </D:prop>
  </D:propstat> </D:response> </D:multistatus>
```

The value of the DAV:owner property is a single DAV:href XML element containing the URL of the principal that owns this resource.

The value of the DAV:supported-privilege-set property is a tree of supported privileges:

```
  DAV:all (aggregate, abstract)
      |
    +-- DAV:read
    +-- DAV:write (aggregate, abstract)
         |
         +-- http://www.webdav.org/acl/create
         +-- http://www.webdav.org/acl/update
         +-- http://www.webdav.org/acl/delete
      +-- DAV:read-acl
      +-- DAV:write-acl
```

The DAV:current-user-privilege-set property contains two privileges, DAV:read, and DAV:read-acl. This indicates that the current authenticated user only has the ability to read the resource, and read the DAV:acl property on the resource.

The DAV:acl property contains a set of four ACEs:

ACE #1: The principal identified by the URL http://www.foo.org/users/esedlar is granted the DAV:read, DAV:write, and DAV:read-acl privileges.

ACE #2: The principals identified by the URL http://www.foo.org/groups/marketing/ are denied the DAV:read privilege. In this example, the principal URL identifies a group, which is represented by a collection principal.

ACE #3: In this ACE, the principal is a property principal, specifically the DAV:owner property. When evaluating this ACE, the value of the DAV:owner property is retrieved, and is examined to see if it contains a DAV:href XML element. If so, the URL within the DAV:href element is read, and identifies a principal. In this ACE, the owner is granted DAV:read-acl, and DAV:write-acl privileges.

ACE #4: This ACE grants the DAV:all principal (all users) the

DAV:read privilege. This ACE is inherited from the resource http://www.foo.org/top/, the parent collection of this resource.

6 ACL SEMANTICS

The ACL semantics define how multiple ACEs that match the current user are combined, what are the constraints on how ACEs can be ordered, and which principals must have an ACE.

```
<!ELEMENT acl-semantics acl-sem*>

<!ELEMENT acl-sem (ace-combination, ace-ordering, allowed-ace,
required-principal*)>
```

6.1 ACE Combination

The DAV:ace-combination element defines how privileges from multiple ACEs that match the current user will be combined to determine the access privileges for that user. Multiple ACEs may match the same user because the same principal can appear in multiple ACEs, because multiple principals can identify the same user, and because one principal can be a member of another principal.

```
<!ELEMENT ace-combination
 (first-match | all-grant-before-any-deny | specific-deny-
overrides-grant)>
```

6.1.1 DAV:first-match ACE Combination

The ACEs are evaluated in the order in which they appear in the ACL. If the first ACE that matches the current user does not grant all the privileges needed for the request, the request MUST fail.

```
<!ELEMENT first-match EMPTY>
```

6.1.2 DAV:all-grant-before-any-deny ACE Combination

The ACEs are evaluated in the order in which they appear in the ACL. If an evaluated ACE denies a privilege needed for the request, the request MUST fail. If all ACEs have been evaluated without the user being granted all privileges needed for the request, the request MUST fail.

```
<!ELEMENT all-grant-before-any-deny EMPTY>
```

6.1.3 DAV:specific-deny-overrides-grant ACE Combination

All ACEs in the ACL are evaluated. An "individual ACE" is one whose principal identifies the current user. A "group ACE" is one whose principal is a collection that contains a principal

that identifies the current user. A privilege is granted if it is granted by an individual ACE and not denied by an individual ACE, or if it is granted by a group ACE and not denied by an individual or group ACE. A request MUST fail if any of its needed privileges are not granted.

```
<!ELEMENT specific-deny-overrides-grant EMPTY>
```

6.2 ACE Ordering

The DAV:ace-ordering element defines a constraint on how the ACEs can be ordered in the ACL.

```
<!ELEMENT ace-ordering (deny-before-grant)? >
```

6.2.1 DAV:deny-before-grant ACE Ordering

This element indicates that all deny ACEs must precede all grant ACEs.

```
<!ELEMENT deny-before-grant EMPTY>
```

6.3 Allowed ACE

The DAV:allowed-ace XML element specifies constraints on what kinds of ACEs are allowed in an ACL.

```
<!ELEMENT allowed-ace (principal-only-one-ace | grant-only)*>
```

6.3.1 DAV:principal-only-one-ace ACE Constraint

This element indicates that a principal can appear in only one ACE per resource.

```
<!ELEMENT principal-only-one-ace EMPTY>
```

6.3.2 DAV:grant-only ACE Constraint

This element indicates that ACEs with deny clauses are not allowed.

```
<!ELEMENT grant-only EMPTY>
```

6.4 Required Principals

The required principal elements identify which principals must have an ACE defined in the ACL.

```
<!ELEMENT required-principal
  (href | all | authenticated | unauthenticated | property |
self)>
```

For example, the following element requires that the ACL contain a DAV:owner property ACE:

```
<D:required-principal xmlns:D="DAV:">
```

```
      <D:property> <D:owner/> </D:property>
    </D:required-principal>
```

7 ACCESS CONTROL AND EXISTING METHODS

This section defines the impact of access control functionality on existing methods.

7.1 OPTIONS

If the server supports access control, it MUST return "access-control" as a field in the DAV response header from an OPTIONS request on any resource implemented by that server.

7.1.1 Example - OPTIONS

>> Request <<

```
OPTIONS /foo.html HTTP/1.1
Host: www.webdav.org
Content-Length: 0
```

>> Response <<

```
HTTP/1.1 200 OK
DAV: 1, 2, access-control
Allow: OPTIONS, GET, PUT, PROPFIND, PROPPATCH, ACL
```

In this example, the OPTIONS response indicates that the server supports access control and that /foo.html can have its access control list modified by the ACL method.

7.2 MOVE

When a resource is moved from one location to another due to a MOVE request, the non-inherited ACEs in the DAV:acl property of the resource MUST NOT be modified, or the MOVE request fails.

7.3 COPY

The DAV:acl property on the resource at the destination of a COPY MUST be the same as if the resource was created by an individual resource creation request (e.g. MKCOL, PUT).

8 ACCESS CONTROL METHODS

8.1 ACL

The ACL method modifies the access control list (which can be read via the DAV:acl property) of a resource. Specifically, the ACL method only permits modification to ACEs that are not inherited, and are not protected. An ACL method invocation

modifies all non-inherited and non-protected ACEs in a resource's access control list to exactly match the ACEs contained within in the DAV:acl XML element (specified in Section 5.4) of the request body. An ACL request body MUST contain only one DAV:acl XML element. Unless the non-inherited

and non-protected ACEs of the DAV:acl property of the resource can be updated to be exactly the value specified in the ACL request, the ACL request MUST fail.

It is possible that the ACEs visible to the current user in the DAV:acl property may only be a portion of the complete set of ACEs on that resource. If this is the case, an ACL request only modifies the set of ACEs visible to the current user, and does not affect any non-visible ACE.

In order to avoid overwriting DAV:acl changes by another client, a client SHOULD acquire a WebDAV lock on the resource before retrieving the DAV:acl property of a resource that it intends on updating.

> Implementation Note: Two common operations are to add or remove an ACE from an existing access control list. To accomplish this, a client uses the PROPFIND method to retrieve the value of the DAV:acl property, then parses the returned access control list to remove all inherited and protected ACEs (these ACEs are tagged with the DAV:inherited and DAV:protected XML elements). In the remaining set of non-inherited, non-protected ACEs, the client can add or remove one or more ACEs before submitting the final ACE set in the request body of the ACL method.

8.1.1 ACL Preconditions

An implementation MAY enforce one or more of the following constraints on an ACL request. If the constraint is violated, a 403 (Forbidden) response MUST be returned and the indicated XML element MUST be returned as the top level element in an XML response body.

<DAV:ace-conflict/>: A conflict exists between two or more ACEs submitted in the ACL request.

<DAV:protected-ace-conflict/>: A conflict exists between an ACE in the ACL request and a protected ACE on the resource. For example, if the resource has a protected ACE granting DAV:write to a given principal, then it would be a protected ACE conflict if the ACL request submitted an ACE denying DAV:write to the same principal.

<DAV:inherited-ace-conflict/>: A conflict exists between an ACE in the ACL request and an inherited ACE on the resource. For example, if the resource inherits an ACE from its parent

collection granting DAV:write to a given principal, then it would be an inherited ACE conflict if the ACL request submitted an ACE denying DAV:write to the same principal. Note that reporting of this error will be implementation-dependent.

Implementations have the choice to either report this error, or to allow the ACE to be set, and then let normal ACE evaluation rules determine whether the new ACE has any impact on the privileges available to a specific principal.

<DAV:too-many-aces/>: An implementation MAY limit the number of ACEs in an ACL. However, ACL-compliant servers MUST support at least one ACE granting privileges to a single principal, and one ACE granting privileges to a collection principal.

<DAV:deny-before-grant/>: All non-inherited deny ACEs MUST precede all non-inherited grant ACEs.

<DAV:principal-only-one-ace/>: For implementations that have the DAV:principal-only-one-ace constraint (defined in Section 6.3.1), this XML element indicates that fulfilling the ACL request would result in multiple ACEs for one or more principals.

<DAV:grant-only/>: For implementations that have the DAV:grant-only constraint (defined in Section 6.3.2), this XML element indicates the request contained one or more deny ACEs.

<DAV:required-principal>: One or more required principals (see Section 6.4) would not be present in the access control list after processing the ACL request. The DAV:required-principal XML element MUST contain a list of the missing principal(s), following the syntax specified in Section 6.4.

8.1.2 Example: the ACL method

In the following example, user "fielding", authenticated by information in the Authorization header, grants the principal identified by the URL http://www.foo.org/users/esedlar (i.e., the user "esedlar") read and write privileges, grants the owner of the resource read-acl and write-acl privileges, and grants everyone read privileges.

>> Request <<

```
ACL /top/container/ HTTP/1.1
Host: www.foo.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="fielding",
   realm="users@foo.org", nonce="...",
   uri="/top/container/", response="...", opaque="..."
```

```
<?xml version="1.0" encoding="utf-8" ?>
<D:acl xmlns:D="DAV:">
  <D:ace>
```

```
      <D:principal>
        <D:href>http://www.foo.org/users/esedlar</D:href>
      </D:principal>
      <D:grant>
        <D:privilege> <D:read/> </D:privilege>
        <D:privilege> <D:write/> </D:privilege>
      </D:grant>
    </D:ace>
    <D:ace>
      <D:principal>
        <D:property> <D:owner/> </D:property>
      </D:principal>
      <D:grant>
        <D:privilege> <D:read-acl/> </D:privilege>
        <D:privilege> <D:write-acl/> </D:privilege>
      </D:grant>
    </D:ace>
    <D:ace>
      <D:principal> <D:all/> </D:principal>
      <D:grant>
        <D:privilege> <D:read/> </D:privilege>
      </D:grant>
    </D:ace> </D:acl>

>> Response <<

HTTP/1.1 200 OK
```

8.1.3 Example: ACL method failure due to protected ACE conflict

In the following request, user "fielding", authenticated by information in the Authorization header, attempts to deny the principal identified by the URL http://www.foo.org/users/esedlar (i.e., the user "esedlar") write privileges. Prior to the request, the DAV:acl property on the resource contained a protected ACE (see Section 5.4.3) granting DAV:owner the DAV:read and DAV:write privileges. The principal identified by URL http://www.foo.org/users/esedlar is the owner of the resource. The ACL method invocation fails because the submitted ACE conflicts with the protected ACE, thus violating the semantics of ACE protection.

```
>> Request <<

ACL /top/container/ HTTP/1.1
Host: www.foo.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
```

```
     Authorization: Digest username="fielding",
        realm="users@foo.org", nonce="...",
        uri="/top/container/", response="...", opaque="..."

     <?xml version="1.0" encoding="utf-8" ?>
```

```
<D:acl xmlns:D="DAV:">
  <D:ace>
    <D:principal>
      <D:href>http://www.foo.org/users/esedlar</D:href>
    </D:principal>
    <D:deny>
      <D:privilege> <D:write/> </D:privilege>
    </D:deny>
  </D:ace>
</D:acl>

>> Response <<

HTTP/1.1 403 Forbidden
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<DAV:protected-ace-conflict/>
```

8.1.4 Example: ACL method failure due to an inherited ACE conflict

In the following request, user "ejw", authenticated by information in the Authorization header, tries to change the access control list on the resource http://www.foo.org/top/index.html. This resource has two inherited ACEs.

Inherited ACE #1 grants the principal identified by URL http://www.foo.org/users/ejw (i.e., the user "ejw") http://www.foo.org/privs/write-all and DAV:read-acl privileges. On this server, http://www.foo.org/privs/write-all is an aggregate privilege containing DAV:write, and DAV:write-acl.

Inherited ACE #2 grants principal DAV:all the DAV:read privilege.

The request attempts to set a (non-inherited) ACE, denying the principal identified by the URL http://www.foo.org/users/ejw (i.e., the user ôejwö) DAV:write permission. This conflicts with inherited ACE #1. Note that the decision to report an inherited ACE conflict is specific to this server implementation. Another server implementation could have allowed the new ACE to be set, and then used normal ACE evaluation rules to determine whether the new ACE has any impact on the privileges available to a principal.

```
>> Request <<
```

```
ACL /top/index.html HTTP/1.1
Host: www.foo.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="ejw",
```

```
   realm="users@foo.org", nonce="...",
   uri="/top/index.html", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:acl xmlns:D="DAV:" xmlns:F="http://www.foo.org/privs/">
  <D:ace>
    <D:principal>
      <D:href>http://www.foo.org/users/ejw</D:href>
    </D:principal>
    <D:grant><D:write/></D:grant>
  </D:ace>
</D:acl>

>> Response <<

HTTP/1.1 403 Forbidden
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<DAV:inherited-ace-conflict/>
```

8.1.5 Example: ACL method failure due to an attempt to set grant and deny in a single ACE.

In this example, user "ygoland", authenticated by information in the Authorization header, tries to change the access control list on the resource http://www.foo.org/diamond/engagement-ring.gif. The ACL request includes a single, syntactically and semantically incorrect ACE, which attempts to grant the collection principal identified by the URL http://www.foo.org/users/friends/ DAV:read privilege and deny the principal identified by URL http://www.foo.org/users/ygoland-so (i.e., the user "ygoland-so") DAV:read privilege. However, it is illegal to have multiple principal elements, as well as both a grant and deny element in the same ACE, so the request fails due to poor syntax.

```
>> Request <<

ACL /diamond/engagement-ring.gif HTTP/1.1
Host: www.foo.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="ygoland",
   realm="users@foo.org", nonce="...",
   uri="/diamond/engagement-ring.gif", response="...",
opaque="..."
```

```
<?xml version="1.0" encoding="utf-8" ?>
<D:acl xmlns:D="DAV:">
  <D:ace>
    <D:principal>
```

```
          <D:href>http://www.foo.org/users/friends/</D:href>
        </D:principal>
        <D:grant><D:read/></D:grant>
        <D:principal>
          <D:href>http://www.foo.org/users/ygoland-so</D:href>
        </D:principal>
        <D:deny><D:read/></D:deny>
      </D:ace>
    </D:acl>

    >> Response <<

    HTTP/1.1 400 Bad Request
    Content-Length: 0
```

Note that if the request had been divided into two ACEs, one to grant, and one to deny, the request would have been syntactically well formed.

9 ACCESS CONTROL REPORTS

9.1 REPORT Method

A REPORT request is an extensible mechanism for obtaining information about a resource. Unlike a resource property, which has a single value, the value of a report can depend on additional information specified in the REPORT request body and in the REPORT request headers.

Marshalling:

The body of a REPORT request specifies which report is being requested, as well as any additional information that will be used to customize the report.

The request MAY include a Depth header.

The response body for a successful request MUST contain the requested report.

If a Depth request header is included, the response MUST be a 207 Multi-Status.

Postconditions:

The REPORT method MUST NOT change the content or dead properties of any resource.

If a Depth request header is included, the request MUST be

applied separately to the collection itself and to all members of the collection that satisfy the Depth value. The DAV:prop element of a DAV:response for a given resource MUST contain the requested report for that resource.

9.2 DAV:acl-principal-props Report

The DAV:acl-principle-props report returns, for all principals in the DAV:acl property that are identified by http(s) URLs, the value of the properties specified in the REPORT request body. In the case where a principal URL appears multiple times, the DAV:acl-principal-props report MUST return the properties for that principal only once.

Marshalling

The request body MUST be a DAV:acl-principal-props XML element.

<!ELEMENT acl-principal-props ANY>
ANY value: a sequence of one or more elements, with at most one DAV:prop element.
prop: see RFC 2518, Section 12.11

The response body for a successful request MUST be a DAV:multistatus XML element (i.e., the response uses the same format as the response for PROPFIND).

multistatus: see RFC 2518, Section 12.9

The response body for a successful DAV:acl-principal-props REPORT request MUST contain a DAV:response element for each principal identified by an http(s) URL listed in a DAV:principal XML element of an ACE within the DAV:acl property of the resource identified by the Request-URI.

9.2.1 Example: DAV:acl-principal-props Report

Resource http;//www.webdav.org/index.html has an ACL with three ACEs:

ACE #1: All principals (DAV:all) have DAV:read and DAV:read-current-user-privilege-set access.

ACE #2: The principal identified by http://www.webdav.org/people/gstein (the user ôgsteinö) is granted DAV:write, DAV:write-acl, DAV:read-acl privileges.

ACE #3: The collection principal identified by http://www.webdav.org/groups/authors/ (the ôauthorsö group) is granted DAV:write and DAV:read-acl privileges.

The following example shows a DAV:acl-principal-props report requesting the DAV:displayname property. It returns the value

of DAV:displayname for resources
http://www.webdav.org/people/gstein and
http://www.webdav.org/groups/authors/ , but not for DAV:all,
since this is not an http(s) URL.

```
>> Request <<

REPORT /index.html HTTP/1.1
Host: www.webdav.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:acl-principal-props xmlns:D="DAV:">
  <D:prop>
    <D:displayname/>
  </D:prop>
</D:acl-principal-props>

>> Response <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.webdav.org/people/gstein</D:href>
    <D:propstat>
      <D:prop>
        <D:displayname>Greg Stein</D:displayname>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:href>http://www.webdav.org/groups/authors/</D:href>
    <D:propstat>
      <D:prop>
        <D:displayname>Site authors</D:displayname>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

9.3 DAV:principal-match REPORT

The DAV:principal-match REPORT is used to identify all members of a collection that match the current user. In particular, if the collection contains principals, the report can be used to identify all members of the collection that match the current

user. Alternatively, if the collection contains resources that have a property that identifies a principal (e.g. DAV:owner), then the report can be used to identify all members of the collection whose property identifies a principal that matches the current user. For example, this report can return all of

the resources in a collection hierarchy that are owned by the current user.

Marshalling:

The request body MUST be a DAV:principal-match XML element.

<!ELEMENT principal-match ((principal-property | self), prop?)>
<!ELEMENT principal-property ANY>
ANY value: an element whose value identifies a property. The expectation is the value of the named property typically contains an href element that contains the URI of a principal
<!ELEMENT self EMPTY>
prop: see RFC 2518, Section 12.11

The response body for a successful request MUST be a DAV:multistatus XML element.

multistatus: see RFC 2518, Section 12.9

The response body for a successful DAV:principal-match REPORT request MUST contain a DAV:response element for each member of the collection that matches the current user. When the DAV:principal-property element is used, a match occurs if the current user is the same as the principal identified by the URI found in the DAV:href element of the property identified by the DAV:principal-property element. When the DAV:self element is used in a DAV:principal-match report issued against a collection principal, it matches a child of the collection principal if that child (a principal resource) identifies the same principal as the current user.

If DAV:prop is specified in the request body, the properties specified in the DAV:prop element MUST be reported in the DAV:response elements.

9.3.1 Example: DAV:principal-match REPORT

The following example identifies the members of the collection identified by the URL http://www.webdav.org/doc/ that are owned by the current user. The current user (ôgclemmö) is authenticated using Digest authentication.

>> Request <<

```
REPORT /doc/ HTTP/1.1
Host: www.webdav.org
```

```
Authorization: Digest username="gclemm",
   realm="gclemm@webdav.org", nonce="...",
   uri="/papers/", response="...", opaque="..."
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
```

```
<?xml version="1.0" encoding="utf-8" ?>
<D:principal-match xmlns:D="DAV:">
  <D:principal-property>
    <D:owner/>
  </D:principal-property>
</D:principal-match>

>> Response <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.webdav.org/doc/foo.html</D:href>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:response>
  <D:response>
    <D:href>http://www.webdav.org/doc/img/bar.gif</D:href>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:response>
</D:multistatus>
```

10 XML PROCESSING

Implementations of this specification MUST support the XML element ignore rule, as specified in Section 23.3.2 of [RFC2518], and the WebDAV XML Namespace interpretation convention, described in Section 23.4 of [RFC2518].

11 INTERNATIONALIZATION CONSIDERATIONS

In this specification, the only human-readable content can be found in the description XML element, found within the DAV:supported-privilege-set property. This element contains a human-readable description of the capabilities controlled by a privilege. As a result, the description element must be capable of representing descriptions in multiple character sets. Since the description element is found within a WebDAV property, it is represented on-the-wire as XML [REC-XML], and hence can leverage XML's language tagging and character set encoding capabilities. Specifically, XML processors must, at minimum, be able to read XML elements encoded using the UTF-8 [UTF-8] encoding of the ISO 10646 multilingual plane. XML examples in this specification demonstrate use of the charset parameter of the Content-Type header, as defined in [RFC3023],

as well as the XML "encoding" attribute, which together provide charset identification information for MIME and XML processors.

For XML elements other than the description element, it is expected that implementations will treat the property names, privilege names, and values as tokens, and convert these tokens

into human-readable text in the user's language and character set when displayed to a person. Only a generic WebDAV property display utility would display these values in their raw form to a human user.

For error reporting, we follow the convention of HTTP/1.1 status codes, including with each status code a short, English description of the code (e.g., 200 (OK)). While the possibility exists that a poorly crafted user agent would display this message to a user, internationalized applications will ignore this message, and display an appropriate message in the user's language and character set.

Further internationalization considerations for this protocol are described in the WebDAV Distributed Authoring protocol specification [RFC2518].

12 SECURITY CONSIDERATIONS

Applications and users of this access control protocol should be aware of several security considerations, detailed below. In addition to the discussion in this document, the security considerations detailed in the HTTP/1.1 specification [RFC2616], the WebDAV Distributed Authoring Protocol specification [RFC2518], and the XML Media Types specification [RFC3023] should be considered in a security analysis of this protocol.

12.1 Increased Risk of Compromised Users

In the absence of a mechanism for remotely manipulating access control lists, if a single user's authentication credentials are compromised, only those resources for which the user has access permission can be read, modified, moved, or deleted. With the introduction of this access control protocol, if a single compromised user has the ability to change ACLs for a broad range of other users (e.g., a super-user), the number of resources that could be altered by a single compromised user increases. This risk can be mitigated by limiting the number of people who have write-acl privileges across a broad range of resources.

12.2 Risks of the DAV:read-acl and DAV:current-user-privilege-set Privileges

The ability to read the access privileges (stored in the DAV:acl property), or the privileges permitted the currently authenticated user (stored in the DAV:current-user-privilege-set property) on a resource may seem innocuous, since reading

an ACL cannot possibly affect the resource's state. However, if all resources have world-readable ACLs, it is possible to perform an exhaustive search for those resources that have inadvertently left themselves in a vulnerable state, such as being world-writeable. In particular, the property retrieval

method PROPFIND, executed with Depth infinity on an entire hierarchy, is a very efficient way to retrieve the DAV:acl or DAV:current-user-privilege-set properties. Once found, this vulnerability can be exploited by a denial of service attack in which the open resource is repeatedly overwritten. Alternately, writeable resources can be modified in undesirable ways.

To reduce this risk, read-acl privileges should not be granted to unauthenticated principals, and restrictions on read-acl and cuprivset privileges for authenticated principals should be carefully analyzed when deploying this protocol. Access to the current-user-privilege-set property will involve a tradeoff of usability versus security. When the current-user-privilege-set is visible, user interfaces are expected to provide enhanced information concerning permitted and restricted operations, yet this information may also indicate a vulnerability that could be exploited. Deployment of this protocol will need to evaluate this tradeoff in light of the requirements of the deployment environment.

12.3 No Foreknowledge of Initial ACL

In an effort to reduce protocol complexity, this protocol specification intentionally does not address the issue of how to manage or discover the initial ACL that is placed upon a resource when it is created. The only way to discover the initial ACL is to create a new resource, then retrieve the value of the DAV:acl property. This assumes the principal creating the resource also has been granted the DAV:read-acl privilege.

As a result, it is possible that a principal could create a resource, and then discover that its ACL grants privileges that are undesirable. Furthermore, this protocol makes it possible (though unlikely) that the creating principal could be unable to modify the ACL, or even delete the resource. Even when the ACL can be modified, there will be a short period of time when the resource exists with the initial ACL before its new ACL can be set.

Several factors mitigate this risk. Human principals are often aware of the default access permissions in their editing environments and take this into account when writing information. Furthermore, default privilege policies are usually very conservative, limiting the privileges granted by the initial ACL.

13 AUTHENTICATION

Authentication mechanisms defined in WebDAV also apply to this WebDAV Access Control Protocol, in particular the Basic and Digest authentication mechanisms defined in [RFC2617].

14 IANA CONSIDERATIONS

This document uses the namespace defined by [RFC2518] for XML elements. All other IANA considerations mentioned in [RFC2518] also applicable to WebDAV ACL.

15 INTELLECTUAL PROPERTY

The following notice is copied from RFC 2026, section 10.4, and describes the position of the IETF concerning intellectual property claims made against this document.

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use other technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

16 ACKNOWLEDGEMENTS

This protocol is the collaborative product of the WebDAV ACL design team: Bernard Chester, Geoff Clemm, Anne Hopkins, Barry Lind, Sean Lyndersay, Eric Sedlar, Greg Stein, and Jim Whitehead. The authors are grateful for the detailed review and comments provided by Jim Amsden, Gino Basso, Murthy Chintalapati, Dennis Hamilton, Laurie Harper, Ron Jacobs, Chris Knight, Remy Maucherat, Larry Masinter, Yaron Goland, Lisa Dusseault, and Joe Orton. Prior work on WebDAV access control protocols has been performed by Yaron Goland, Paul Leach, Lisa Dusseault, Howard Palmer, and Jon Radoff. We would like to acknowledge the foundation laid for us by the authors of the WebDAV and HTTP protocols upon which this protocol is layered, and the invaluable feedback from the WebDAV working group.

17 REFERENCES

17.1 Normative References

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

[REC-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/REC-xml-19980210.

[RFC2616] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1." RFC 2616. U.C. Irvine, Compaq, Xerox, Microsoft, MIT/LCS, June, 1999.

[RFC2617] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication." RFC 2617. Northwestern University, Verisign, AbiSource, Agranat, Microsoft, Netscape, Open Market, June, 1999.

[RFC2518] Y. Goland, E. Whitehead, A. Faizi, S. R. Carter, D. Jensen, "HTTP Extensions for Distributed Authoring -- WEBDAV." RFC 2518. Microsoft, U.C. Irvine, Netscape, Novell, February, 1999.

[RFC2368] P. Hoffman, L. Masinter, J. Zawinski, "The mailto URL scheme." RFC 2368. Internet Mail Consortium, Xerox, Netscape, July, 1998.

[RFC2255] T. Howes, M. Smith, "The LDAP URL Format." RFC 2255. Netscape, December, 1997.

[RFC3023] M. Murata, S. St.Laurent, D. Kohn, "XML Media Types." RFC 3023. IBM Tokyo Research Laboratory, simonstl.com, Skymoon Ventures, January, 2001.

[UTF-8] F. Yergeau, "UTF-8, a transformation format of Unicode and ISO 10646." RFC 2279. Alis Technologies. January, 1998.

17.2 Informational References

[RFC2026] S.Bradner, "The Internet Standards Process û Revision 3." RFC 2026, BCP 9. Harvard, October, 1996.

18 AUTHORS' ADDRESSES

Geoffrey Clemm
Rational Software
20 Maguire Road
Lexington, MA 02421
Email: geoffrey.clemm@rational.com

Anne Hopkins
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
Email: annehop@microsoft.com

Eric Sedlar
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
Email: esedlar@us.oracle.com

Jim Whitehead
U.C. Santa Cruz
Dept. of Computer Science
Baskin Engineering
1156 High Street
Santa Cruz, CA 95064
Email: ejw@cse.ucsc.edu

19 APPENDICIES

19.1 XML Document Type Definition

```
<!-- Privileges -->

<!ELEMENT read EMPTY>
<!ELEMENT write EMPTY>
<!ELEMENT read-acl EMPTY>
<!ELEMENT read-current-user-privilege-set EMPTY>
<!ELEMENT write-acl EMPTY>
<!ELEMENT all EMPTY>


<!-- Principal Properties (Section 4) -->

<!ELEMENT is-principal (#PCDATA)>

<!ELEMENT alternate-URL (href*)>

<!-- Access Control Properties (Section 5) -->

<!-- DAV:owner Property (Section 5.1) -->

<!ELEMENT owner (href prop?)>
<!ELEMENT prop (see [RFC2518], section 12.11)>
```

```
<!-- DAV:supported-privilege-set Property (Section 5.2) -->

<!ELEMENT supported-privilege-set (supported-privilege*)>
<!ELEMENT supported-privilege
 (privilege, abstract?, description, supported-privilege*)>
```

```
<!ELEMENT privilege ANY>
<!ELEMENT abstract EMPTY>
<!ELEMENT description #PCDATA>
<!ELEMENT privilege ANY>


<!-- DAV:current-user-privilege-set Property (Section 5.3) -->

<!ELEMENT current-user-privilege-set (privilege*)>


<!-- DAV:acl Property (Section 5.4) -->

<!ELEMENT acl (ace*)>

<!ELEMENT ace (principal, (grant|deny), protected?,
inherited?)>
<!ELEMENT principal ((href, prop?)
 | all | authenticated | unauthenticated
 | property | self)>

<!ELEMENT prop (see [RFC2518], section 12.11)>
<!ELEMENT all EMPTY>
<!ELEMENT authenticated EMPTY>
<!ELEMENT unauthenticated EMPTY>
<!ELEMENT property ANY>
<!ELEMENT self EMPTY>

<!ELEMENT grant (privilege+)>
<!ELEMENT deny (privilege+)>
<!ELEMENT privilege ANY>

<!ELEMENT protected EMPTY>

<!ELEMENT inherited (href)>


<!-- DAV:principal-collection-set Property (Section 5.6) -->

<!ELEMENT principal-collection-set (href*)>


<!-- DAV:acl-semantics Property (Section 6) -->

<!ELEMENT acl-semantics acl-sem*>
<!ELEMENT acl-sem (ace-combination, ace-ordering, allowed-ace,
required-principal*)>

<!ELEMENT ace-combination
```

```
      (first-match | all-grant-before-any-deny | specific-deny-
     overrides-grant)>
     <!ELEMENT first-match EMPTY>
     <!ELEMENT all-grant-before-any-deny EMPTY>
```

```
<!ELEMENT specific-deny-overrides-grant EMPTY>

<!ELEMENT ace-ordering (deny-before-grant)? >
<!ELEMENT deny-before-grant EMPTY>

<!ELEMENT allowed-ace (principal-only-one-ace | grant-only)*>
<!ELEMENT principal-only-one-ace EMPTY>
<!ELEMENT grant-only EMPTY>

<!ELEMENT required-principal
  (href | all | authenticated | unauthenticated | property |
self)>


<!-- ACL method preconditions (Section 8.1.1) -->


<!ELEMENT ace-conflict EMPTY>
<!ELEMENT protected-ace-conflict EMPTY>
<!ELEMENT inherited-ace-conflict EMPTY>
<!ELEMENT too-many-aces EMPTY>


<!-- REPORT Method -->

<!ELEMENT acl-principal-props ANY>
ANY value: a sequence of one or more elements, with at most one
DAV:prop element.
<!ELEMENT principal-match ((principal-property | self), prop?)>
<!ELEMENT principal-property ANY>
ANY value: an element whose value identifies a property. The
expectation is the value of the named property typically
contains an href element that contains the URI of a principal
<!ELEMENT self EMPTY>
```

20 NOTE TO RFC EDITOR

*** This section (Section 20) MUST be removed before publication as an RFC ***

Section 9.1 defines the REPORT method. The REPORT method is also defined in draft-ietf-deltav-versioning-15, in Section 3.6, using identical text. This was done to avoid making this specification dependent on draft-ietf-deltav-versioning.

If draft-ietf-deltav-versioning is published as an RFC before this specification, Section 9.1 MUST be removed.