Network Working Group
Request for Comments: 3744
Category: Standards Track

G. Clemm
IBM
J. Reschke
greenbytes
E. Sedlar
Oracle Corporation
J. Whitehead
U.C. Santa Cruz
May 2004

Web Distributed Authoring and Versioning (WebDAV)

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright © The Internet Society (2004). All Rights Reserved.

Abstract

This document specifies a set of methods, headers, message bodies, properties, and reports that define Access Control extensions to the WebDAV Distributed Authoring Protocol. This protocol permits a client to read and modify access control lists that instruct a server whether to allow or deny operations upon a resource (such as HyperText Transfer Protocol (HTTP) method invocations) by a given principal. A lightweight representation of principals as Web resources supports integration of a wide range of user management repositories. Search operations allow discovery and manipulation of principals using human names.

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 and metadata managed by 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 what operations you can perform on 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 "operations you can perform" are 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.

Since every ACE contains the identifier of a principal, client software operated by a human must provide a mechanism for selecting this principal. This specification uses http(s) scheme URLs to identify principals, which are represented as WebDAV-capable resources. There is no guarantee that the URLs identifying principals will be meaningful to a human. For example, http://www.example.com/u/256432 and http://www.example.com/people/Greg.Stein are both valid URLs that could be used to identify the same principal. To remedy this, every principal resource has the DAV:displayname property containing a human-readable name for the principal.

Since a principal can be identified by multiple URLs, it raises the problem of determining exactly which principal is being referenced in a given ACE. It is impossible for a client to determine that an ACE granting the read privilege to http://www.example.com/people/Greg.Stein also affects the principal at http://www.example.com/u/256432. That is, a client has no mechanism for determining that two URLs identify the same principal resource. As a result, this specification requires clients to use just one of the many possible URLs for a principal when creating ACEs. A client can discover which URL to use by retrieving the DAV:principal-URL property (Section 4.2) from a principal resource. No matter which of the principal's URLs is used with PROPFIND, the property always returns the same URL.

With a system having hundreds to thousands of principals, the problem arises of how to allow a human operator of client software to select just one of these principals. One approach is to use broad collection hierarchies to spread the principals over a large number of collections, yielding few principals per collection. An example of this is a two level hierarchy with the first level containing 36 collections (a-z, 0-9), and the second level being another 36, creating collections /a/a/, /a/b/, ..., /a/z/, such that a principal with last name "Stein" would appear at /s/t/Stein. In effect, this pre-computes a common query, search on last name, and encodes it into a hierarchy. The drawback with this scheme is that it handles only a small set of predefined queries, and drilling down through the collection hierarchy adds unnecessary steps (navigate down/up) when the user already knows the principal's name. While organizing principal URLs into a hierarchy is a valid namespace organization, users should not be forced to navigate this hierarchy to select a principal.

This specification provides the capability to perform substring searches over a small set of properties on the resources representing principals. This permits searches based on last name, first name, user name, job title, etc. Two separate searches are supported, both via the REPORT method, one to search principal resources (DAV:principal-property-search, Section 9.4), the other to determine which properties may be searched at all (DAV:principal-search-property-set, Section 9.5).

Once a principal has been identified in an ACE, a server evaluating that ACE must know the identity of the principal making a protocol request, and must validate that that principal is who they claim to be, a process known as authentication. 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 group 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 a 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 ways ACLs are to be evaluated is described in Section 6. Client discovery of access control capability using OPTIONS is described in Section 7.2. Interactions between access control functionality and existing HTTP and WebDAV methods are described in the remainder of Section 7. The access control setting method, ACL, is specified in Section 8. Four reports that provide limited server-side searching capabilities are described in Section 9. Sections on XML processing (Section 10), Internationalization considerations (Section 11), security considerations (Section 12), and authentication (Section 13) round out the specification. An appendix (Appendix A) provides an XML Document Type Definition (DTD) for the XML elements defined in the specification.

1.1. Terms

This document 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.

group

A "group" is a principal that represents a set of other principals.

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 on a resource, means the privilege cannot be set in an access control element (ACE) on that resource.

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

2. Principals

A principal is a network resource that represents a distinct human or computational actor that initiates access to network resources. Users and groups are represented as principals in many implementations; 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 URIs, 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 be a group, where a group is a principal that represents a set of other principals, called the members of the group. If a person or computational agent matches a principal resource that is a member of a group, they also match the group. Membership in a group is recursive, so if a principal is a member of group GRPA, and GRPA is a member of group GRPB, then the principal is also a member of GRPB.

3. Privileges

Ability to perform a given method on a resource MUST 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 MUST be denied any HTTP access to that resource, unless the principal matches an ACE constructed using the DAV:all, DAV:authenticated, or DAV:unauthenticated pseudo-principals (see Section 5.5.1). Servers MUST report a 403 "Forbidden" error if access is denied, except in the case where the privilege restricts the ability to know the resource exists, in which case 404 "Not Found" may be returned.

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

Privileges may be declared to be "abstract" for a given resource, in which case they cannot be set in an ACE on that resource. 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; therefore, 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 resources (defined in [RFC2518], Section 3) 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 resources). On the transition from null to stateful resource, the initial access control list is set by the server's default ACL value policy (if any).

Server implementations MAY define new privileges beyond those defined in this specification. Privileges defined by individual implementations MUST NOT use the DAV: namespace, and instead should use a namespace that they control, such as an http scheme URL.

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. Any implementation-defined privilege that also controls access to GET and PROPFIND must be aggregated under DAV:read - if an ACL grants access to DAV:read, the client may expect that no other privilege needs to be granted to have access to GET and PROPFIND. Additionally, the read privilege MUST control the OPTIONS method.

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

3.2. DAV:write Privilege

The write privilege controls methods that lock a resource or 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 [RFC2518]), so effective write access requires that both write privileges and write locking requirements are satisfied. Any implementation-defined privilege that also controls access to methods modifying content, dead properties or collection membership must be aggregated under DAV:write, e.g., if an ACL grants access to DAV:write, the client may expect that no other privilege needs to be granted to have access to PUT and PROPPATCH.

```
<!ELEMENT write EMPTY>
```

3.3. DAV:write-properties Privilege

The DAV:write-properties privilege controls methods that modify the dead properties of the resource, such as PROPPATCH. Whether this privilege may be used to control access to any live properties is determined by the implementation. Any implementation-defined privilege that also controls access to methods modifying dead properties must be aggregated under DAV:write-properties - e.g., if an ACL grants access to DAV:write-properties, the client can safely expect that no other privilege needs to be granted to have access to PROPPATCH.

```
<!ELEMENT write-properties EMPTY>
```

3.4. DAV:write-content Privilege

The DAV:write-content privilege controls methods that modify the content of an existing resource, such as PUT. Any implementation-defined privilege that also controls access to content must be aggregated under DAV:write-content - e.g., if an ACL grants access to DAV:write-content, the client can safely expect that no other privilege needs to be granted to have access to PUT. Note that PUT - when applied to an unmapped URI - creates a new resource and therefore is controlled by the DAV:bind privilege on the parent collection.

```
<!ELEMENT write-content EMPTY>
```

3.5. DAV:unlock Privilege

The DAV:unlock privilege controls the use of the UNLOCK method by a principal other than the lock owner (the principal that created a lock can always perform an UNLOCK). While the set of users who may lock a resource is most commonly the same set of users who may modify a resource, servers may allow various kinds of administrators to unlock resources locked by others. Any privilege controlling access by non-lock owners to UNLOCK MUST be aggregated under DAV:unlock.

A lock owner can always remove a lock by issuing an UNLOCK with the correct lock token and authentication credentials. That is, even if a principal does not have DAV:unlock privilege, they can still remove locks they own. Principals other than the lock owner can remove a lock only if they have DAV:unlock privilege and they issue an UNLOCK with the correct lock token. Lock timeout is not affected by the DAV:unlock privilege.

```
<!ELEMENT unlock EMPTY>
```

3.6. 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>
```

3.7. 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 writable.

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>
```

3.8. 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>
```

3.9. DAV:bind Privilege

The DAV:bind privilege allows a method to add a new member URL to the specified collection (for example via PUT or MKCOL). It is ignored for resources that are not collections.

```
<!ELEMENT bind EMPTY>
```

3.10. DAV:unbind Privilege

The DAV:unbind privilege allows a method to remove a member URL from the specified collection (for example via DELETE or MOVE). It is ignored for resources that are not collections.

```
<!ELEMENT unbind EMPTY>
```

3.11. DAV:all Privilege

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

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

3.12. Aggregation of Predefined Privileges

Server implementations are free to aggregate the predefined privileges (defined above in Sections 3.1-3.10) subject to the following limitations:

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

DAV:write MUST contain DAV:bind, DAV:unbind, DAV:write-properties and DAV:write-content.

4. Principal Properties

Principals are manifested to clients as a WebDAV resource, identified by a URL. A principal MUST have a non-empty 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 XML element in the value of the DAV:resourcetype property. The element type declaration for DAV:principal is:

```
<!ELEMENT principal EMPTY>
```

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

4.1. DAV:alternate-URI-set

This protected property, if non-empty, contains the URIs of network resources with additional descriptive information about the principal. This property identifies 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. One expected use for this property is the storage of an LDAP [RFC2255] scheme URL. A user-agent encountering an LDAP URL could use LDAP [RFC2251] to retrieve additional machine-readable directory information about the principal, and display that information in its user interface. Support for this property is REQUIRED, and the value is empty if no alternate URI exists for the principal.

```
<!ELEMENT alternate-URI-set (href*)>
```

4.2. DAV:principal-URL

A principal may have many URLs, but there must be one "principal URL" that clients can use to uniquely identify a principal. This protected property contains the URL that MUST be used to identify this principal in an ACL request. Support for this property is REQUIRED.

```
<!ELEMENT principal-URL (href)>
```

4.3. DAV:group-member-set

This property of a group principal identifies the principals that are direct members of this group. Since a group may be a member of another group, a group may also have indirect members (i.e., the members of its direct members). A URL in the DAV:group-member-set for a principal MUST be the DAV:principal-URL of that principal.

```
<!ELEMENT group-member-set (href*)>
```

4.4. DAV:group-membership

This protected property identifies the groups in which the principal is directly a member. Note that a server may allow a group to be a member of another group, in which case the DAV:group-membership of those other groups would need to be queried in order to determine the groups in which the principal is indirectly a member. Support for this property is REQUIRED.

```
<!ELEMENT group-membership (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.

Access control properties (especially DAV:acl and DAV:inherited-acl-set) are defined on the resource identified by the Request-URI of a PROPFIND request. A direct consequence is that if the resource is accessible via multiple URI, the value of access control properties is the same across these URI.

HTTP resources that support the WebDAV Access Control Protocol MUST contain the following properties. Null resources (described in Section 3 of [RFC2518]) MUST NOT contain the following properties.

5.1. DAV:owner

This 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.

Servers MAY implement DAV:owner as protected property and MAY return an empty DAV:owner element as property value in case no owner information is available.

```
<!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.example.com/papers/. The principal making the request is authenticated using Digest authentication. The value of DAV:owner is the URL http://www.example.com/acl/users/gstein, wrapped in the DAV:href XML element.

>> Request <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:owner/>
  </D:prop>
</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.example.com/papers/</D:href>
    <D:propstat>
      <D:prop>
        <D:owner>
          <D:href>http://www.example.com/acl/users/gstein</D:href>
        </D:owner>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </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.example.com/papers>. Since DAV:owner is a protected property on this particular server, it responds with a 207 (Multi-Status) response that contains a 403 (Forbidden) status code for the act of setting DAV:owner. Section 8.2.1 of [RFC2518] describes PROPPATCH status code information, Section 11 of [RFC2518] describes the Multi-Status response and Sections 1.6 and 3.12 of [RFC3253] describe additional error marshaling for PROPPATCH attempts on protected properties.

>> Request <<

```
PROPPATCH /papers/ HTTP/1.1
Host: www.example.com
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="jim",
  realm="users@example.com", 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.example.com/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.example.com/papers/</D:href>
    <D:propstat>
      <D:prop><D:owner/></D:prop>
      <D:status>HTTP/1.1 403 Forbidden</D:status>
      <D:responsedescription>
        <D:error><D:cannot-modify-protected-property/></D:error>
        Failure to set protected property (DAV:owner)
      </D:responsedescription>
    </D:propstat>
  </D:response>
</D:multistatus>
```

5.2. DAV:group

This property identifies a particular principal as being the "group" of the resource. This property is commonly found on repositories that implement the Unix privileges model.

Servers MAY implement DAV:group as protected property and MAY return an empty DAV:group element as property value in case no group information is available.

```
<!ELEMENT group (href?)>
```

5.3. 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 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. Servers MUST indicate the human language of the description using the xml:lang attribute and SHOULD consider the HTTP Accept-Language request header when selecting one of multiple available languages.

```
<!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.

5.3.1. 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.example.com/papers/. The value of the DAV:supported-privilege-set property is a tree of supported privileges (using "[XML Namespace , localname]" to identify each privilege):

```
[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)
          ### [DAV:, write-properties]
          ### [DAV:, write-content]
   #
   ### [DAV:, unlock]
```

This privilege tree is not normative (except that it reflects the normative aggregation rules given in Section 3.12), and many possible privilege trees are possible.

>> Request <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:supported-privilege-set/>
  </D:prop>
</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.example.com/papers/</D:href>
    <D:propstat>
      <D:prop>
        <D:supported-privilege-set>
          <D:supported-privilege>
            <D:privilege><D:all/></D:privilege>
           <D:abstract/>
            <D:description xml:lang="en">
              Any operation
            </D:description>
            <D:supported-privilege>
              <D:privilege><D:read/></D:privilege>
              <D:description xml:lang="en">
                Read any object
              </D:description>
              <D:supported-privilege>
                <D:privilege><D:read-acl/></D:privilege>
                <D:abstract/>
                <D:description xml:lang="en">Read ACL</D:description>
              </D:supported-privilege>
              <D:supported-privilege>
                <D:privilege>
                  <D:read-current-user-privilege-set/>
                </D:privilege>
                <D:abstract/>
                <D:description xml:lang="en">
                  Read current user privilege set property
                </D:description>
              </D:supported-privilege>
            </D:supported-privilege>
            <D:supported-privilege>
              <D:privilege><D:write/></D:privilege>
              <D:description xml:lang="en">
                Write any object
              </D:description>
              <D:supported-privilege>
                <D:privilege><D:write-acl/></D:privilege>
                <D:description xml:lang="en">
                  Write ACL
                </D:description>
                <D:abstract/>
              </D:supported-privilege>
              <D:supported-privilege>
                <D:privilege><D:write-properties/></D:privilege>
                <D:description xml:lang="en">
                  Write properties
                </D:description>
              </D:supported-privilege>
              <D:supported-privilege>
                <D:privilege><D:write-content/></D:privilege>
                <D:description xml:lang="en">
```

5.4. 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 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.

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

Continuing the example from Section 5.3.1, this example shows a client requesting the DAV:current-user-privilege-set property from the resource with URL http://www.example.com/papers/. The username of the principal making the request is "khare", and Digest authentication is used in the request. The principal with username "khare" 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 5.3.1), the principal with username "khare" 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 "khare" is not listed in an ACE granting that principal write permission.

>> Request <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:current-user-privilege-set/>
  </D:prop>
</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.example.com/papers/</D:href>
  <D:propstat>
    <D:prop>
      <D:current-user-privilege-set>
        <D:privilege><D:read/></D:privilege>
      </D:current-user-privilege-set>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
  </D:response>
</D:multistatus>
```

5.5. 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 | invert), (grant|deny), protected?,
               inherited?)>
```

5.5.1. 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 and that principal matches the current user or, if the principal is a group, a member of that group matches the current user.

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

Some servers may support ACEs applying to those users NOT matching the current principal, e.g., all users not in a particular group. This can be done by wrapping the DAV:principal element with DAV:invert.

```
<!ELEMENT invert principal>
```

5.5.2. 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>
```

5.5.3. ACE Protection

A server indicates an ACE is protected by including the DAV:protected element in the ACE. If the ACL of a resource contains an ACE with a DAV:protected element, an attempt to remove that ACE from the ACL MUST fail.

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

5.5.4. 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)>
```

5.5.5. Example: Retrieving a Resource's Access Control List

Continuing the example from Sections 5.3.1 and 5.4.1, this example shows a client requesting the DAV:acl property from the resource with URL http://www.example.com/papers/. There are two ACEs defined in this ACL:

ACE #1: The group identified by URL http://www.example.com/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 5.3.1), this means the "maintainers" 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 "maintainers" group) can read the DAV:acl property and the DAV:current-user-privilege-set property.

>> Request <<

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

<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:acl/>
  </D:prop>
</D:propfind>
```

>> Response <<

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

<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.example.com/papers/</D:href>
    <D:propstat>
      <D:prop>
        <D:acl>
        <D:ace>
          <D:principal>
            <D:href
            >http://www.example.com/acl/groups/maintainers</D:href>
          </D:principal>
          <D:grant>
            <D:privilege><D:write/></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>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

5.6. DAV:acl-restrictions

This protected property defines the types of ACLs supported by this server, to avoid clients needlessly getting errors. When a client tries to set an ACL via the ACL method, the server may reject the attempt to set the ACL as specified. The following properties indicate the restrictions the client must observe before setting an ACL:

<grant-only>	Deny ACEs are not supported
<no-invert>	Inverted ACEs are not supported
<deny-before-grant>	All deny ACEs must occur before any grant ACEs
<required-principal>	Indicates which principals are required to be present

```
<!ELEMENT acl-restrictions (grant-only?, no-invert?,
                            deny-before-grant?,
                            required-principal?)>
```

5.6.1. DAV:grant-only

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

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

5.6.2. DAV:no-invert ACE Constraint

This element indicates that ACEs with the <invert> element are not allowed.

```
<!ELEMENT no-invert EMPTY>
```

5.6.3. DAV:deny-before-grant

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

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

5.6.4. Required Principals

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

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

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>
```

5.6.5. Example: Retrieving DAV:acl-restrictions

In this example, the client requests the value of the DAV:acl-restrictions property. Digest authentication provides credentials for the principal operating the client.

>> Request <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:acl-restrictions/>
  </D:prop>
</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.example.com/papers/</D:href>
    <D:propstat>
      <D:prop>
        <D:acl-restrictions>
          <D:grant-only/>
          <D:required-principal>
            <D:all/>
          </D:required-principal>
        </D:acl-restrictions>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

5.7. DAV:inherited-acl-set

This protected property contains a set of URLs that identify other resources that also control the access to this resource. To have a privilege on a resource, not only must the ACL on that resource (specified in the DAV:acl property of that resource) grant the privilege, but so must the ACL of each resource identified in the DAV:inherited-acl-set property of that resource. Effectively, the privileges granted by the current ACL are ANDed with the privileges granted by each inherited ACL.

```
<!ELEMENT inherited-acl-set (href*)>
```

5.8. DAV:principal-collection-set

This protected property of a resource contains a set of URLs that identify the root collections that contain the principals that are available on the server that implements this resource. A WebDAV Access Control Protocol user agent could use the contents of DAV:principal-collection-set to retrieve 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 principal collections, and therefore clients should not assume they have retrieved an exhaustive listing. Additionally, a server MAY elect to report none of the principal collections it knows about, in which case the property value would be empty.

The value of DAV:principal-collection-set gives the scope of the DAV:principal-property-search REPORT (defined in Section 9.4). Clients use the DAV:principal-property-search REPORT to populate their user

interface with a list of principals. Therefore, servers that limit a client's ability to obtain principal information will interfere with the client's ability to manipulate access control lists, due to the difficulty of getting the URL of a principal for use in an ACE.

5.8.1. 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.example.com/papers/. The property contains the two URLs, http://www.example.com/acl/users/ and http://www.example.com/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.example.com
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="yarong",
  realm="users@example.com", nonce="...",
  uri="/papers/", response="...", opaque="..."

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

5.9. 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.example.com
Content-type: text/xml; charset="utf-8"
Content-Length: xxx
Depth: 0
Authorization: Digest username="ejw",
  realm="users@example.com", nonce="...",
  uri="/top/container/", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop>
    <D:owner/>
    <D:supported-privilege-set/>
    <D:current-user-privilege-set/>
    <D:acl/>
  </D:prop>
</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.example.com/acl/">
  <D:response>
    <D:href>http://www.example.com/top/container/</D:href>
    <D:propstat>
      <D:prop>
        <D:owner>
          <D:href>http://www.example.com/users/gclemm</D:href>
        </D:owner>
        <D:supported-privilege-set>
          <D:supported-privilege>
            <D:privilege><D:all/></D:privilege>
            <D:abstract/>
            <D:description xml:lang="en">
              Any operation
            </D:description>
            <D:supported-privilege>
              <D:privilege><D:read/></D:privilege>
              <D:description xml:lang="en">
                Read any object
              </D:description>
            </D:supported-privilege>
            <D:supported-privilege>
              <D:privilege><D:write/></D:privilege>
              <D:abstract/>
              <D:description xml:lang="en">
                Write any object
              </D:description>
              <D:supported-privilege>
                <D:privilege><A:create/></D:privilege>
                <D:description xml:lang="en">
                  Create an object
                </D:description>
              </D:supported-privilege>
              <D:supported-privilege>
                <D:privilege><A:update/></D:privilege>
                <D:description xml:lang="en">
                  Update an object
                </D:description>
              </D:supported-privilege>
            </D:supported-privilege>
            <D:supported-privilege>
              <D:privilege><A:delete/></D:privilege>
              <D:description xml:lang="en">
                Delete an object
              </D:description>
            </D:supported-privilege>
            <D:supported-privilege>
              <D:privilege><D:read-acl/></D:privilege>
              <D:description xml:lang="en">
                Read the ACL
              </D:description>
            </D:supported-privilege>
```

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 (using "[XML Namespace , localname]" to identify each privilege):

```
[DAV:, all] (aggregate, abstract)
   #
   ### [DAV:, read]
   ### [DAV:, write] (aggregate, abstract)
          #
          ### [http://www.example.com/acl, create]
          ### [http://www.example.com/acl, update]
          ### [http://www.example.com/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.example.com/users/esedlar is granted the DAV:read, DAV:write, and DAV:read-acl privileges.

ACE #2: The principals identified by the URL http://www.example.com/groups/mrktng are denied the DAV:read privilege. In this example, the principal URL identifies a group.

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.example.com/top, the parent collection of this resource.

6. ACL Evaluation

WebDAV ACLs are evaluated in similar manner as ACLs on Windows NT and in NFSv4 [RFC3530]). An ACL is evaluated to determine whether or not access will be granted for a WebDAV request. ACEs are maintained in a particular order, and are evaluated until all of the permissions required by the current request have been granted, at which point the ACL evaluation is terminated and access is granted. If, during ACL evaluation, a <deny> ACE (matching the current user) is encountered for a privilege which has not yet been granted, the ACL evaluation is terminated and access is denied. Failure to have all required privileges granted results in access being denied.

Note that the semantics of many other existing ACL systems may be represented via this mechanism, by mixing deny and grant ACEs. For example, consider the standard "rwx" privilege scheme used by UNIX. In this scheme, if the current user is the owner of the file, access is granted if the corresponding privilege bit is set and denied if not set, regardless of the permissions set on the file's group and for the world. An ACL for UNIX permissions of "r--rw-r--" might be constructed like:

```
<D:acl>
  <D:ace>
    <D:principal>
      <D:property><D:owner/></D:property>
    </D:principal>
    <D:grant>
      <D:privilege><D:read/></D:privilege>
    </D:grant>
  </D:ace>
  <D:ace>
    <D:principal>
      <D:property><D:owner/></D:property>
    </D:principal>
    <D:deny>
      <D:privilege><D:all/></D:privilege>
    </D:deny>
  </D:ace>
  <D:ace>
    <D:principal>
      <D:property><D:group/></D:property>
    </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:group/></D:property>
    </D:principal>
    <D:deny>
      <D:privilege><D:all/></D:privilege>
    </D:deny>
  </D:ace>
  <D:ace>
    <D:principal><D:all></D:principal>
    <D:grant>
      <D:privilege><D:read/></D:privilege>
    </D:grant>
  </D:ace>
</D:acl>
```

and the <acl-restrictions> would be defined as:

```
<D:no-invert/>
<D:required-principal>
  <D:all/>
  <D:property><D:owner/></D:property>
  <D:property><D:group/><D:group/>
</D:required-principal>
```

Note that the client can still get errors from a UNIX server in spite of obeying the <acl-restrictions>, including <D:allowed-principal> (adding an ACE specifying a principal other than the ones in the ACL above) or <D:ace-conflict> (by trying to reorder the ACEs in the example above), as these particular implementation semantics are too complex to be captured with the simple (but general) declarative restrictions.

7. Access Control and existing methods

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

7.1. Any HTTP method

7.1.1. Error Handling

The WebDAV ACL mechanism requires the usage of HTTP method "preconditions" as described in section 1.6 of RFC3253 for ALL HTTP methods. All HTTP methods have an additional precondition called DAV:need-privileges. If an HTTP method fails due to insufficient privileges, the response body to the "403 Forbidden" error MUST contain the <DAV:error> element, which in turn contains the <DAV:need-privileges> element, which contains one or more <DAV:resource> elements indicating which resource had insufficient privileges, and what the lacking privileges were:

```
<!ELEMENT need-privileges (resource)* >
<!ELEMENT resource ( href , privilege ) >
```

Since some methods require multiple permissions on multiple resources, this information is needed to resolve any ambiguity. There is no requirement that all privilege violations be reported - for implementation reasons, some servers may only report the first privilege violation. For example:

>> Request <<

```
MOVE /a/b/ HTTP/1.1
Host: www.example.com
Destination: http://www.example.com/c/d
```

>> Response <<

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

<D:error xmlns:D="DAV:">
  <D:need-privileges>
    <D:resource>
      <D:href>/a</D:href>
      <D:privilege><D:unbind/></D:privilege>
    </D:resource>
    <D:resource>
      <D:href>/c</D:href>
      <D:privilege><D:bind/></D:privilege>
    </D:resource>
  </D:need-privileges>
</D:error>
```

7.2. 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. A value of "access-control" in the DAV header MUST indicate that the server supports all MUST level requirements and REQUIRED features specified in this document.

7.2.1. Example - OPTIONS

>> Request <<

```
OPTIONS /foo.html HTTP/1.1
Host: www.example.com
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.

7.3. MOVE

When a resource is moved from one location to another due to a MOVE request, the non-inherited and non-protected ACEs in the DAV:acl property of the resource MUST NOT be modified, or the MOVE request fails. Handling of inherited and protected ACEs is intentionally undefined to give server implementations flexibility in how they implement ACE inheritance and protection.

7.4. 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). Clients wishing to preserve the DAV:acl property across a copy need to read the DAV:acl property prior to the COPY, then perform an ACL operation on the new resource at the destination to restore, insofar as this is possible, the original access control list.

7.5. LOCK

A lock on a resource ensures that only the lock owner can modify ACEs that are not inherited and not protected (these are the only ACEs that a client can modify with an ACL request). A lock does not protect inherited or protected ACEs, since a client cannot modify them with an ACL request on that resource.

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 5.5) 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 MUST enforce the following constraints on an ACL request. If the constraint is violated, a 403 (Forbidden) or 409 (Conflict) response MUST be returned and the indicated XML element MUST be returned as a child of a top level DAV:error element in an XML response body.

Though these status elements are generally expressed as empty XML elements (and are defined as EMPTY in the DTD), implementations MAY return additional descriptive XML elements as children of the status element. Clients MUST be able to accept children of these status elements. Clients that do not understand the additional XML elements should ignore them.

(DAV:no-ace-conflict): The ACEs submitted in the ACL request MUST NOT conflict with each other. This is a catchall error code indicating that an implementation-specific ACL restriction has been violated.

(DAV:no-protected-ace-conflict): The ACEs submitted in the ACL request MUST NOT conflict with the protected ACEs on the resource. For example, if the resource has a protected ACE granting DAV:write to a given principal, then it would not be consistent if the ACL request submitted an ACE denying DAV:write to the same principal.

(DAV:no-inherited-ace-conflict): The ACEs submitted in the ACL request MUST NOT conflict with the inherited ACEs 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 not be consistent 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 MUST either report this error or 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:limited-number-of-aces): The number of ACEs submitted in the ACL request MUST NOT exceed the number of ACEs allowed on that resource. However, ACL-compliant servers MUST support at least one ACE granting privileges to a single principal, and one ACE granting privileges to a group.

(DAV:deny-before-grant): All non-inherited deny ACEs MUST precede all non-inherited grant ACEs.

(DAV:grant-only): The ACEs submitted in the ACL request MUST NOT include a deny ACE. This precondition applies only when the ACL restrictions of the resource include the DAV:grant-only constraint (defined in Section 5.6.1).

(DAV:no-invert): The ACL request MUST NOT include a DAV:invert element. This precondition applies only when the ACL semantics of the resource includes the DAV:no-invert constraint (defined in Section 5.6.2).

(DAV:no-abstract): The ACL request MUST NOT attempt to grant or deny an abstract privilege (see Section 5.3).

(DAV:not-supported-privilege): The ACEs submitted in the ACL request MUST be supported by the resource.

(DAV:missing-required-principal): The result of the ACL request MUST have at least one ACE for each principal identified in a DAV:required-principal XML element in the ACL semantics of that resource (see Section 5.5).

(DAV:recognized-principal): Every principal URL in the ACL request MUST identify a principal resource.

(DAV:allowed-principal): The principals specified in the ACEs submitted in the ACL request MUST be allowed as principals for the resource. For example, a server where only authenticated principals can access resources would not allow the DAV:all or DAV:unauthenticated principals to be used in an ACE, since these would allow unauthenticated access to resources.

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.example.com/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.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="fielding",
  realm="users@example.com", 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.example.com/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.example.com/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 5.5.3) granting DAV:owner the DAV:read and DAV:write privileges. The principal identified by URL http://www.example.com/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.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="fielding",
  realm="users@example.com", 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.example.com/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" ?>
<D:error xmlns:D="DAV:">
  <D:no-protected-ace-conflict/>
</D:error>
```

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.example.com/top/index.html. This resource has two inherited ACEs.

Inherited ACE #1 grants the principal identified by URL http://www.example.com/users/ejw (i.e., the user "ejw") http://www.example.com/privs/write-all and DAV:read-acl privileges. On this server, http://www.example.com/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.example.com/users/ejw (i.e., the user "ejw") 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.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="ejw",
  realm="users@example.com", nonce="...",
  uri="/top/index.html", response="...", opaque="..."

<?xml version="1.0" encoding="utf-8" ?>
<D:acl xmlns:D="DAV:" xmlns:F="http://www.example.com/privs/">
  <D:ace>
    <D:principal>
       <D:href>http://www.example.com/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" ?>
<D:error xmlns:D="DAV:">
  <D:no-inherited-ace-conflict/>
</D:error>
```

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.example.com/diamond/engagement-ring.gif. The ACL request includes a single, syntactically and semantically incorrect ACE, which attempts to grant the group identified by the URL http://www.example.com/users/friends DAV:read privilege and deny the principal identified by URL http://www.example.com/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.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Authorization: Digest username="ygoland",
  realm="users@example.com", 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.example.com/users/friends</D:href>
    </D:principal>
    <D:grant><D:read/></D:grant>
    <D:principal>
      <D:href>http://www.example.com/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

The REPORT method (defined in Section 3.6 of [RFC3253]) provides an extensible mechanism for obtaining information about a resource. Unlike the PROPFIND method, which returns the value of one or more named properties, the REPORT method can involve more complex processing. REPORT is valuable in cases where the server has access to all of the information needed to perform the complex request (such as a query), and where it would require multiple requests for the client to retrieve the information needed to perform the same request.

A server that supports the WebDAV Access Control Protocol MUST support the DAV:expand-property report (defined in Section 3.8 of [RFC3253]).

9.2. DAV:acl-principal-prop-set Report

The DAV:acl-principal-prop-set report returns, for all principals in the DAV:acl property (of the Request-URI) that are identified by http(s) URLs or by a DAV:property principal, 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-prop-set report MUST return the properties for that principal only once. Support for this report is REQUIRED.

One expected use of this report is to retrieve the human readable name (found in the DAV:displayname property) of each principal found in an ACL. This is useful for constructing user interfaces that show each ACE in a human readable form.

Marshalling

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

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

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

The response body for a successful request MUST be a DAV:multistatus XML element (i.e., the response uses the same format as the response for PROPFIND). In the case where there are no response elements, the returned multistatus XML element is empty.

```
multistatus: see RFC 2518, Section 12.9
```

The response body for a successful DAV:acl-principal-prop-set 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.

Postconditions:

(DAV:number-of-matches-within-limits): The number of matching principals must fall within server-specific, predefined limits. For example, this condition might be triggered if a search specification would cause the return of an extremely large number of responses.

9.2.1. Example: DAV:acl-principal-prop-set Report

Resource http://www.example.com/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.example.com/people/gstein (the user "gstein") is granted DAV:write, DAV:write-acl, DAV:read-acl privileges.

ACE #3: The group identified by http://www.example.com/groups/authors (the "authors" group) is granted DAV:write and DAV:read-acl privileges.

The following example shows a DAV:acl-principal-prop-set report requesting the DAV:displayname property. It returns the value of DAV:displayname for resources http://www.example.com/people/gstein and http://www.example.com/groups/authors , but not for DAV:all, since this is not an http(s) URL.

>> Request <<

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

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

>> 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.example.com/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.example.com/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 (at any depth) of the collection identified by the Request-URI that are principals and 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), 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. Support for this report is REQUIRED.

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
```

This report is only defined when the Depth header has value "0"; other values result in a 400 (Bad Request) error response. Note that [RFC3253], Section 3.6, states that if the Depth header is not present, it defaults to a value of "0". The response body for a successful request MUST be a DAV:multistatus XML element. In the case where there are no response elements, the returned multistatus XML element is empty.

```
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 matched by 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 group, it matches the group if a member 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.example.com/doc that are owned by the current user. The current user ("gclemm") is authenticated using Digest authentication.

>> Request <<

```
REPORT /doc/ HTTP/1.1
Host: www.example.com
Authorization: Digest username="gclemm",
  realm="users@example.com", nonce="...",
  uri="/papers/", response="...", opaque="..."
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Depth: 0

<?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.example.com/doc/foo.html</D:href>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:response>
  <D:response>
    <D:href>http://www.example.com/doc/img/bar.gif</D:href>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:response>
</D:multistatus>
```

9.4. DAV:principal-property-search REPORT

The DAV:principal-property-search REPORT performs a search for all principals whose properties contain character data that matches the search criteria specified in the request. One expected use of this report is to discover the URL of a principal associated with a given person or group by searching for them by name. This is done by searching over DAV:displayname, which is defined on all principals.

The actual search method (exact matching vs. substring matching vs, prefix-matching, case-sensitivity) deliberately is left to the server implementation to allow implementation on a wide set of possible user management systems. In cases where the implementation of DAV:principal-property-search is not constrained by the semantics of an underlying user management repository, preferred default semantics are caseless substring matches.

For implementation efficiency, servers do not typically support searching on all properties. A search requesting properties that are not searchable for a particular principal will not match that principal.

Support for the DAV:principal-property-search report is REQUIRED.

> **Implementation Note:** The value of a WebDAV property is a sequence of well-formed XML, and hence can include any character in the Unicode/ISO-10646 standard, that is, most known characters in

human languages. Due to the idiosyncrasies of case mapping across human languages, implementation of case-insensitive matching is non-trivial. Implementors of servers that do perform substring matching are strongly encouraged to consult "The Unicode Standard" [UNICODE4], especially Section 5.18, Subsection "Caseless Matching", for guidance when implementing their case-insensitive matching algorithms.

Implementation Note: Some implementations of this protocol will use an LDAP repository for storage of principal metadata. The schema describing each attribute (akin to a WebDAV property) in an LDAP repository specifies whether it supports case-sensitive or caseless searching. One of the benefits of leaving the search method to the discretion of the server implementation is the default LDAP attribute search behavior can be used when implementing the DAV:principal-property-search report.

Marshalling:

The request body MUST be a DAV:principal-property-search XML element containing a search specification and an optional list of properties. For every principal that matches the search specification, the response will contain the value of the requested properties on that principal.

```
<!ELEMENT principal-property-search
 ((property-search+), prop?, apply-to-principal-collection-set?) >
```

By default, the report searches all members (at any depth) of the collection identified by the Request-URI. If DAV:apply-to-principal-collection-set is specified in the request body, the request is applied instead to each collection identified by the DAV:principal-collection-set property of the resource identified by the Request-URI.

The DAV:property-search element contains a prop element enumerating the properties to be searched and a match element, containing the search string.

```
<!ELEMENT property-search (prop, match) >
prop: see RFC 2518, Section 12.11

<!ELEMENT match #PCDATA >
```

Multiple property-search elements or multiple elements within a DAV:prop element will be interpreted with a logical AND.

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

The response body for a successful request MUST be a DAV:multistatus XML element. In the case where there are no response elements, the returned multistatus XML element is empty.

```
multistatus: see RFC 2518, Section 12.9
```

The response body for a successful DAV:principal-property-search REPORT request MUST contain a DAV:response element for each principal whose property values satisfy the search specification given in DAV:principal-property-search.

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.

Preconditions:

None

Postconditions:

(DAV:number-of-matches-within-limits): The number of matching principals must fall within server-specific, predefined limits. For example, this condition might be triggered if a search specification would cause the return of an extremely large number of responses.

9.4.1. Matching

There are several cases to consider when matching strings. The easiest case is when a property value is "simple" and has only character information item content (see [REC-XML-INFOSET]). For example, the search string "julian" would match the DAV:displayname property with value "Julian Reschke". Note that the on-the-wire marshaling of DAV:displayname in this case is:

```
<D:displayname xmlns:D="DAV:">Julian Reschke</D:displayname>
```

The name of the property is encoded into the XML element information item, and the character information item content of the property is "Julian Reschke".

A more complicated case occurs when properties have mixed content (that is, compound values consisting of multiple child element items, other types of information items, and character information item content). Consider the property "aprop" in the namespace "http://www.example.com/props/", marshaled as:

```
<W:aprop xmlns:W="http://www.example.com/props/">
  {cdata 0}<W:elem1>{cdata 1}</W:elem1>
  <W:elem2>{cdata 2}</W:elem2>{cdata 3}
</W:aprop>
```

In this case, matching is performed on each individual contiguous sequence of character information items. In the example above, a search string would be compared to the four following strings:

```
{cdata 0}
{cdata 1}
{cdata 2}
{cdata 3}
```

That is, four individual matches would be performed, one each for {cdata 0}, {cdata 1}, {cdata 2}, and {cdata 3}.

9.4.2. Example: successful DAV:principal-property-search REPORT

In this example, the client requests the principal URLs of all users whose DAV:displayname property contains the substring "doE" and whose "title" property in the namespace "http://BigCorp.com/ns/" (that is, their professional title) contains "Sales". In addition, the client requests five properties to be returned with the matching principals:

In the DAV: namespace: displayname

In the http://www.example.com/ns/ namespace: department, phone, office, salary

The response shows that two principal resources meet the search specification, "John Doe" and "Zygdoebert Smith". The property "salary" in namespace "http://www.example.com/ns/" is not returned, since the principal making the request does not have sufficient access permissions to read this property.

>> Request <<

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

<?xml version="1.0" encoding="utf-8" ?>
<D:principal-property-search xmlns:D="DAV:">
  <D:property-search>
    <D:prop>
      <D:displayname/>
    </D:prop>
    <D:match>doE</D:match>
  </D:property-search>
  <D:property-search>
    <D:prop xmlns:B="http://www.example.com/ns/">
      <B:title/>
    </D:prop>
    <D:match>Sales</D:match>
  </D:property-search>
  <D:prop xmlns:B="http://www.example.com/ns/">
    <D:displayname/>
    <B:department/>
    <B:phone/>
    <B:office/>
    <B:salary/>
  </D:prop>
</D:principal-property-search>
```

>> 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:" xmlns:B="http://BigCorp.com/ns/">
  <D:response>
    <D:href>http://www.example.com/users/jdoe</D:href>
    <D:propstat>
      <D:prop>
        <D:displayname>John Doe</D:displayname>
        <B:department>Widget Sales</B:department>
        <B:phone>234-4567</B:phone>
        <B:office>209</B:office>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
    <D:propstat>
      <D:prop>
        <B:salary/>
      </D:prop>
      <D:status>HTTP/1.1 403 Forbidden</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:href>http://www.example.com/users/zsmith</D:href>
    <D:propstat>
      <D:prop>
        <D:displayname>Zygdoebert Smith</D:displayname>
        <B:department>Gadget Sales</B:department>
        <B:phone>234-7654</B:phone>
        <B:office>114</B:office>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
    <D:propstat>
      <D:prop>
        <B:salary/>
      </D:prop>
      <D:status>HTTP/1.1 403 Forbidden</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

9.5. DAV:principal-search-property-set REPORT

The DAV:principal-search-property-set REPORT identifies those properties that may be searched using the DAV:principal-property-search REPORT (defined in Section 9.4).

Servers MUST support the DAV:principal-search-property-set REPORT on all collections identified in the value of a DAV:principal-collection-set property.

An access control protocol user agent could use the results of the DAV:principal-search-property-set REPORT to present a query interface to the user for retrieving principals.

Support for this report is REQUIRED.

Implementation Note: Some clients will have only limited screen real estate for the display of lists of searchable properties. In this case, a user might appreciate having the most frequently searched properties be displayed on-screen, rather than having to scroll through a long list of searchable properties. One mechanism for signaling the most frequently searched properties is to return them towards the start of a list of properties. A client can then preferentially display the list of properties in order, increasing the likelihood that the most frequently searched properties will appear on-screen, and will not require scrolling for their selection.

Marshalling:

The request body MUST be an empty DAV:principal-search-property-set XML element.

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

The response body MUST be a DAV:principal-search-property-set XML element, containing a DAV:principal-search-property XML element for each property that may be searched with the DAV:principal-property-search REPORT. A server MAY limit its response to just a subset of the searchable properties, such as those likely to be useful to an interactive access control client.

```
<!ELEMENT principal-search-property-set
 (principal-search-property*) >
```

Each DAV:principal-search-property XML element contains exactly one searchable property, and a description of the property.

```
<!ELEMENT principal-search-property (prop, description) >
```

The DAV:prop element contains one principal property on which the server is able to perform a DAV:principal-property-search REPORT.

```
prop: see RFC 2518, Section 12.11
```

The description element is a human-readable description of what information this property represents. Servers MUST indicate the human language of the description using the xml:lang attribute and SHOULD consider the HTTP Accept-Language request header when selecting one of multiple available languages.

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

9.5.1. Example: DAV:principal-search-property-set REPORT

In this example, the client determines the set of searchable principal properties by requesting the DAV:principal-search-property-set REPORT on the root of the server's principal URL collection set, identified by http://www.example.com/users/.

>> Request <<

```
REPORT /users/ HTTP/1.1
Host: www.example.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx
Accept-Language: en, de
Authorization: BASIC d2FubmFtYWs6cGFzc3dvcmQ=
Depth: 0

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

>> Response <<

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="utf-8" ?>
<D:principal-search-property-set xmlns:D="DAV:">
  <D:principal-search-property>
    <D:prop>
      <D:displayname/>
    </D:prop>
    <D:description xml:lang="en">Full name</D:description>
  </D:principal-search-property>
  <D:principal-search-property>
    <D:prop xmlns:B="http://BigCorp.com/ns/">
      <B:title/>
    </D:prop>
    <D:description xml:lang="en">Job title</D:description>
  </D:principal-search-property>
</D:principal-search-property-set>
```

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 XML Namespace recommendation [REC-XML-NAMES].

Note that use of the DAV namespace is reserved for XML elements and property names defined in a standards-track or Experimental IETF RFC.

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 at minimum must be able to read XML elements encoded using the UTF-8 [RFC3629] 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. Futhermore, this specification requires server implementations to tag description fields with the xml:lang attribute (see Section 2.12 of [REC-XML]), which specifies the human language of the description. Additionally, server implementations should take into account the value of the Accept-Language HTTP header to determine which description string to return.

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-writable. 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, writable 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 read-current-user-privilege-set 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 for use with HTTP and WebDAV also apply to this WebDAV Access Control Protocol, in particular the Basic and Digest authentication mechanisms defined in [RFC2617]. Implementation of the ACL spec requires that Basic authentication, if used, MUST only be supported over secure transport such as TLS.

14. IANA Considerations

This document uses the namespace defined by [RFC2518] for XML elements. That is, this specification uses the "DAV:" URI namespace, previously registered in the URI schemes registry. All other IANA considerations mentioned in [RFC2518] are also applicable to this specification.

15. 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, Dylan Barrell, Gino Basso, Murthy Chintalapati, Lisa Dusseault, Stefan Eissing, Tim Ellison, Yaron Goland, Dennis Hamilton, Laurie Harper, Eckehard Hermann, Ron Jacobs, Chris Knight, Remy Maucherat, Larry Masinter, Joe Orton, Peter Raymond, and Keith Wannamaker. We thank Keith Wannamaker for the initial text of the principal property search sections. 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 DeltaV, WebDAV and HTTP protocols upon which this protocol is layered, and the invaluable feedback from the WebDAV working group.

16. References

16.1. Normative References

[REC-XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C REC-xml-20040204, February 2004, <http://www.w3.org/TR/2004/REC-xml-20040204>.

[REC-XML-INFOSET] Cowan, J. and R. Tobin, "XML Information Set (Second Edition)", W3C REC REC-xml-infoset-20040204, February 2004, <http://www.w3.org/TR/2004/REC-xml-infoset-20040204/>.

[REC-XML-NAMES] Bray, T., Hollander, D., and A. Layman, "Namespaces in XML", W3C REC REC-xml-names-19990114, January 1999, <http://www.w3.org/TR/1999/REC-xml-names-19990114>.

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

[RFC2518] Goland, Y., Whitehead, E., Faizi, A., Carter, S., and D. Jensen, "HTTP Extensions for Distributed Authoring -- WEBDAV", RFC 2518, February 1999.

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

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC3023] Makoto, M., St.Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

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

[RFC3530] Shepler, S., Ed., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, STD 63, November 2003.

16.2. Informative References

[RFC2251] Wahl, M., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

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

[UNICODE4] The Unicode Consortium, "The Unicode Standard - Version 4.0", Addison-Wesley, August 2003, <http://www.unicode.org/versions/Unicode4.0.0/>. ISBN 0321185781[1].

[1] urn:isbn:0321185781

Appendix A. WebDAV XML Document Type Definition Addendum

All XML elements defined in this Document Type Definition (DTD) belong to the DAV namespace. This DTD should be viewed as an addendum to the DTD provided in [RFC2518], Section 23.1.

<!-- Privileges -- (Section 3)>

```
<!ELEMENT read EMPTY>
<!ELEMENT write EMPTY>
<!ELEMENT write-properties EMPTY>
<!ELEMENT write-content EMPTY>
<!ELEMENT unlock EMPTY>
<!ELEMENT read-acl EMPTY>
<!ELEMENT read-current-user-privilege-set EMPTY>
<!ELEMENT write-acl EMPTY>
<!ELEMENT bind EMPTY>
<!ELEMENT unbind EMPTY>
<!ELEMENT all EMPTY>
```

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

```
<!ELEMENT principal EMPTY>

<!ELEMENT alternate-URI-set (href*)>
<!ELEMENT principal-URL (href)>
<!ELEMENT group-member-set (href*)>
<!ELEMENT group-membership (href*)>
```

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

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

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

<!-- DAV:group Property (Section 5.2) -->

```
<!ELEMENT group (href?)>
```

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

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

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

<!-- DAV:current-user-privilege-set Property (Section 5.4) -->

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

```
<!-- DAV:acl Property (Section 5.5) -->

<!ELEMENT acl (ace)* >
<!ELEMENT ace ((principal | invert), (grant|deny), protected?,
 inherited?)>

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

<!ELEMENT all EMPTY>
<!ELEMENT authenticated EMPTY>
<!ELEMENT unauthenticated EMPTY>
<!ELEMENT property ANY>
<!ELEMENT self EMPTY>

<!ELEMENT invert principal>

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

<!ELEMENT protected EMPTY>

<!ELEMENT inherited (href)>

<!-- DAV:acl-restrictions Property (Section 5.6) -->

<!ELEMENT acl-restrictions (grant-only?, no-invert?,
 deny-before-grant?, required-principal?)>

<!ELEMENT grant-only EMPTY>
<!ELEMENT no-invert EMPTY>
<!ELEMENT deny-before-grant EMPTY>

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

<!-- DAV:inherited-acl-set Property (Section 5.7) -->

<!ELEMENT inherited-acl-set (href*)>

<!-- DAV:principal-collection-set Property (Section 5.8) -->

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

<!-- Access Control and Existing Methods (Section 7) -->

<!ELEMENT need-privileges (resource)* >
<!ELEMENT resource ( href, privilege )
```

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

```
<!ELEMENT no-ace-conflict EMPTY>
<!ELEMENT no-protected-ace-conflict EMPTY>
<!ELEMENT no-inherited-ace-conflict EMPTY>
<!ELEMENT limited-number-of-aces EMPTY>
<!ELEMENT grant-only EMPTY>
<!ELEMENT no-invert EMPTY>
<!ELEMENT deny-before-grant EMPTY>
<!ELEMENT no-abstract EMPTY>
<!ELEMENT not-supported-privilege EMPTY>
<!ELEMENT missing-required-principal EMPTY>
<!ELEMENT recognized-principal EMPTY>
<!ELEMENT allowed-principal EMPTY>
```

<!-- REPORTs (Section 9) -->

```
<!ELEMENT acl-principal-prop-set 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>

<!ELEMENT principal-property-search ((property-search+), prop?) >
<!ELEMENT property-search (prop, match) >
<!ELEMENT match #PCDATA >

<!ELEMENT principal-search-property-set (
 principal-search-property*) >
<!ELEMENT principal-search-property (prop, description) >
<!ELEMENT description #PCDATA >
```

Appendix B. WebDAV Method Privilege Table (Normative)

The following table of WebDAV methods (as defined in RFC 2518, 2616, and 3253) clarifies which privileges are required for access for each method. Note that the privileges listed, if denied, MUST cause access to be denied. However, given that a specific implementation MAY define an additional custom privilege to control access to existing methods, having all of the indicated privileges does not mean that access will be granted. Note that lack of the indicated privileges does not imply that access will be denied, since a particular implementation may use a sub-privilege aggregated under the indicated privilege to control access. Privileges required refer to the current resource being processed unless otherwise specified.

METHOD	PRIVILEGES
GET	<D:read>
HEAD	<D:read>
OPTIONS	<D:read>
PUT (target exists)	<D:write-content> on target resource
PUT (no target exists)	<D:bind> on parent collection of target
PROPPATCH	<D:write-properties>
ACL	<D:write-acl>
PROPFIND	<D:read> (plus <D:read-acl> and <D:read-current-user-privilege-set> as needed)
COPY (target exists)	<D:read>, <D:write-content> and <D:write-properties> on target resource
COPY (no target exists)	<D:read>, <D:bind> on target collection
MOVE (no target exists)	<D:unbind> on source collection and <D:bind> on target collection
MOVE (target exists)	As above, plus <D:unbind> on the target collection
DELETE	<D:unbind> on parent collection
LOCK (target exists)	<D:write-content>
LOCK (no target exists)	<D:bind> on parent collection
MKCOL	<D:bind> on parent collection
UNLOCK	<D:unlock>
CHECKOUT	<D:write-properties>
CHECKIN	<D:write-properties>
REPORT	<D:read> (on all referenced resources)
VERSION-CONTROL	<D:write-properties>
MERGE	<D:write-content>
MKWORKSPACE	<D:write-content> on parent collection
BASELINE-CONTROL	<D:write-properties> and <D:write-content>
MKACTIVITY	<D:write-content> on parent collection

Index

Authors' Addresses

Geoffrey Clemm
IBM
20 Maguire Road
Lexington, MA 02421
EMail: geoffrey.clemm@us.ibm.com

Julian F. Reschke
greenbytes GmbH
Salzmannstrasse 152
Muenster, NW 48159
Germany
EMail: julian.reschke@greenbytes.de

Eric Sedlar
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
EMail: eric.sedlar@oracle.com

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

Full Copyright Statement

Copyright © The Internet Society (2004).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an “AS IS” basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat 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 on-line IPR repository at http://www.ietf.org/ipr[1].

[1] http://www.ietf.org/ipr

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 implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org[2].

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

[2] mailto:ietf-ipr@ietf.org