

WEBDAV Working Group  
INTERNET-DRAFT  
<[draft-ietf-webdav-requirements-02.txt](#)>

J.A. Slein  
Xerox Corporation  
F. Vitali  
University of Bologna  
E.J. Whitehead, Jr.  
U.C. Irvine  
D.G. Durand  
Boston University  
August 29, 1997

Expires February 28, 1998

## **Requirements for Distributed Authoring and Versioning on the World Wide Web**

### Status of this Memo

This document is an Internet draft. Internet drafts are working documents of the Internet Engineering Task Force (IETF), its areas and its working groups. Note that other groups may also distribute working information as Internet drafts.

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

To learn the current status of any Internet draft please check the "lid-abstracts.txt" listing contained in the Internet drafts shadow directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East coast) or ftp.isi.edu (US West coast). Further information about the IETF can be found at URL: <http://www.ietf.org/>

Distribution of this document is unlimited. Please send comments to the WWW Distributed Authoring and Versioning (WebDAV) mailing list, <w3c-dist-auth@w3.org>, which may be joined by sending a message with subject "subscribe" to <w3c-dist-auth-request@w3.org>. Discussions are archived at URL: <http://www.w3.org/pub/WWW/Archives/Public/w3c-dist-auth/>.

### Abstract

Current World Wide Web (WWW or Web) standards provide simple support for applications which allow remote editing of typed data. In practice, the existing capabilities of the WWW have proven inadequate to support efficient, scalable remote editing free of overwriting conflicts. This document presents a list of features in the form of requirements which, if implemented, would improve the efficiency of common remote editing operations, provide a locking mechanism to prevent overwrite conflicts, improve link management support between non-HTML

data types, provide a simple attribute-value metadata facility, provide for the creation and reading of container data types, and integrate versioning into the WWW.

## **1. Introduction**

Slein, Vitali, Whitehead, Durand Page 1  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

This document describes functionality which, if incorporated in an extension to the existing HTTP proposed standard [[HTTP](#)], would allow tools for remote loading, editing and saving (publishing) of various media types on the WWW to interoperate with any compliant Web server. As much as possible, this functionality is described without suggesting a proposed implementation, since there are many ways to perform the functionality within the WWW framework. It is also possible that a single mechanism could simultaneously satisfy several requirements.

This document is intended to reflect the consensus of the WWW Distributed Authoring and Versioning working group (WebDAV) as to the functionality that needs to be standardized to support distributed authoring and versioning on the Web.

## **2. Rationale**

Current Web standards contain functionality which enables the editing of Web content at a remote location, without direct access to the storage media via an operating system. This capability is exploited by several existing HTML distributed authoring tools, and by a growing number of mainstream applications (e.g., word processors) which allow users to write (publish) their work to an HTTP server. To date, experience from the HTML authoring tools has shown they are unable to meet their users' needs using the facilities of Web standards. The consequence of this is either postponed introduction of distributed authoring capability, or the addition of nonstandard extensions to the HTTP protocol or other Web standards. These extensions, developed in isolation, are not interoperable.

Other authoring applications have wanted to access document repositories or version control systems through Web gateways, and have been similarly frustrated. Where this access is available at all, it is through nonstandard extensions to HTTP or other standards that force clients to use a different interface for each vendor's service.

This document describes requirements for a set of standard extensions to HTTP that would allow distributed Web authoring tools to provide the functionality their users need by means of the same standard syntax across all compliant servers. The broad categories of functionality that need to be standardized are:

Properties

Links  
Locking  
Reservations  
Retrieval of Unprocessed Source  
Partial Write  
Name Space Manipulation  
Collections  
Versioning  
Variants  
Security  
Internationalization

Slein, Vitali, Whitehead, Durand Page 2  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

### **3. Terminology**

Where there is overlap, usage is intended to be consistent with that in the HTTP 1.1 specification [[HTTP](#)].

#### **Client**

A program which issues HTTP requests and accepts responses.

#### **Collection**

A collection is a resource that contains other resources, either directly or by reference.

#### **Distributed Authoring Tool**

A program which can retrieve a source entity via HTTP, allow editing of this entity, and then save/publish this entity to a server using HTTP.

#### **Entity**

The information transferred in a request or response.

#### **Hierarchical Collection**

A hierarchical organization of resources. A hierarchical collection is a resource that contains other resources, including collections, either directly or by reference.

#### **Link**

A typed connection between two or more resources.

#### **Lock**

A mechanism for preventing anyone other than the owner of the lock from accessing a resource.

#### **Member of Version Graph**

A resource that is a node in a version graph, and so is derived from the resources that precede it in the graph, and is the basis of those that succeed it.

#### Property

Named descriptive information about a resource.

#### Reservation

A declaration that one intends to edit a resource.

#### Resource

A network data object or service that can be identified by a URI.

#### Server

A program which receives and responds to HTTP requests.

#### User Agent

The client that initiates a request.

#### Variant

A representation of a resource. A resource may have one or more

Slein, Vitali, Whitehead, Durand

Page 3

[draft-ietf-webdav-requirements-02.txt](#)

WebDAV Requirements

Aug 1997

representations associated with it at any given time.

#### Version Graph

A directed acyclic graph with resources as its nodes, where each node is derived from its predecessor(s).

#### Write Lock

A lock that prevents anyone except its owner from modifying the resource it applies to.

## **4. General Principles**

This section describes a set of general principles that the WebDAV extensions should follow. These principles cut across categories of functionality.

### **4.1. User Agent Interoperability**

All WebDAV clients should be able to work with any WebDAV-compliant HTTP server. It is acceptable for some client/server combinations to provide special features that are not universally available, but the protocol should be sufficient that a basic level of functionality will be universal.

### **4.2. Client Simplicity**

The WebDAV extensions should be designed to allow client implementations to be simple.

### **4.3. Legacy Client Support**

It should be possible to implement a WebDAV-compliant server in such a way that it can interoperate with non-WebDAV clients. Such a server would be able to understand any valid HTTP 1.1 request from an ordinary Web client without WebDAV extensions, and to provide a valid HTTP 1.1 response that does not require the client to understand the extensions.

### **4.4. Data Format Compatibility**

WebDAV-compliant servers should be able to work with existing resources and URIs [[URL](#)]. Special additional information should not become a mandatory part of document formats.

### **4.5. Replicated, Distributed Systems**

Distribution and replication are at the heart of the Internet. All WebDAV extensions should be designed to allow for distribution and replication. Version trees should be able to be split across multiple servers. Collections may have members on different servers. Resources may have properties on different servers. Any resources may be cached or replicated for mobile computing or other reasons. Consequently, the WebDAV extensions must be able to operate in a distributed, replicated environment.

Slein, Vitali, Whitehead, Durand Page 4  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

### **4.6 Parsimony in Client-Server Interactions**

The WebDAV extensions should keep to a minimum the number of interactions between the client and the server needed to perform common functions. For example, publishing a document to the Web will often mean publishing content together with related properties. A client may often need to find out what version graph a particular resource belongs to, or to find out which resource in a version graph is the published one. The extensions should make it possible to do these things efficiently.

### **4.7. Changes to HTTP**

WebDAV adds a number of new types of objects to the Web: properties, collections, version graphs, etc. Existing HTTP methods such as DELETE and PUT will have to operate in well-defined ways in this expanded environment. WebDAV should explicitly address not only new methods, headers, and MIME types, but also any required changes to the existing HTTP methods and headers.

### **4.8. Alternate Transport Mechanisms**

It may be desirable to transport WebDAV requests and responses by other

mechanisms, particularly EMail, in addition to HTTP. The WebDAV protocol specification should not preclude a future body from developing an interoperability specification for disconnected operation via EMail.

## **5. Requirements**

In the requirement descriptions below, the requirement will be stated, followed by its rationale.

### **5.1. Properties**

#### **5.1.1. Functional Requirements**

It must be possible to create, modify, read and delete arbitrary properties on resources of any media type.

#### **5.1.2. Rationale**

Properties describe resources of any media type. They may include bibliographic information such as author, title, publisher, and subject, constraints on usage, PICS ratings, etc. These properties have many uses, such as supporting searches on property values, enforcing copyrights, and the creation of catalog entries as placeholders for objects which are not available in electronic form, or which will be available later.

### **5.2. Links**

#### **5.2.1. Functional Requirements**

It must be possible to create, modify, read and delete typed links between resources of any media type.

#### **5.2.2. Rationale**

One type of link between resources is the hypertext link, which is browsable using a hypertext style point-and-click user interface. Links, whether they are browsable hypertext links, or simply a means of capturing a connection between resources, have many purposes. Links can support pushbutton printing of a multi-resource document in a prescribed order, jumping to the access control page for a resource, and quick browsing of related information, such as a table of contents, an index, a glossary, a bibliographic record, help pages, etc. While link support is provided by the HTML "LINK" element, this is limited only to HTML resources [[HTML](#)]. Similar support is needed for bitmap image types, and other non-HTML media types.

### **5.3. Locking**

### **5.3.1. General Principles**

**5.3.1.1. Independence of locks.** It must be possible to lock a resource without re-reading the resource, and without committing to editing the resource.

**5.3.1.2. Multi-Resource Locking.** It must be possible to take out a lock on multiple resources residing on the same server in a single action, and this locking operation must be atomic across these resources.

### **5.3.2. Functional Requirements**

**5.3.2.1. Write Locks.** It must be possible to restrict modification of a resource to a specific person.

**5.3.2.2. Lock Query.** It must be possible to find out whether a given resource has any active locks, and if so, who holds those locks.

**5.3.2.3. Unlock.** It must be possible to remove a lock.

### **5.3.3. Rationale**

At present, the Web provides limited support for preventing two or more people from overwriting each other's modifications when they save to a given URI. Furthermore, there is no way to discover whether someone else is currently making modifications to a resource. This is known as the "lost update problem," or the "overwrite problem." Since there can be significant cost associated with discovering and repairing lost modifications, preventing this problem is crucial for supporting distributed authoring. A write lock ensures that only one person may modify a resource, preventing overwrites. Furthermore, locking support is a key component of many versioning schemes, a desirable capability for distributed authoring.

An author may wish to lock an entire web of resources even though he is editing just a single resource, to keep the other resources from changing. In this way, an author can ensure that if a local hypertext web is consistent in his distributed authoring tool, it will then be consistent when he writes it to the server. Because of this, it should

be possible to take out a lock without also causing transmission of the contents of a resource.

It is often necessary to guarantee that a lock or unlock operation occurs at the same time across multiple resources, a feature which is supported by the multiple-resource locking requirement. This is useful for preventing a collision between two people trying to establish locks on the same set of resources, since with multi-resource locking, one of

the two people will get a lock. If this same multiple-resource locking scenario was repeated by using atomic lock operations iterated across the resources, the result would be a splitting of the locks between the two people, based on resource ordering and race conditions.

## **5.4. Reservations**

### **5.4.1. Functional Requirements**

**5.4.1.1. Reserve.** It must be possible for a principal to register with the server an intent to edit a given resource, so that other principals can discover who intends to edit the resource.

**5.4.1.2. Reservation Query.** It must be possible to find out whether a given resource has any active reservations, and if so, who currently holds reservations.

**5.4.1.3. Release Reservation.** It must be possible to release the reservation.

### **5.4.2. Rationale**

Experience from configuration management systems has shown that people need to know when they are about to enter a parallel editing situation. Once notified, they either decide not to edit in parallel with the other authors, or they use out-of-band communication (face-to-face, telephone, etc.) to coordinate their editing to minimize the difficulty of merging their results. Reservations are separate from locking, since a write lock does not necessarily imply a resource will be edited, and a reservation does not carry with it any access restrictions. This capability supports versioning, since a check-out typically involves taking out a write lock, making a reservation, and getting the resource to be edited.

## **5.5. Retrieval of Unprocessed Source for Editing**

### **5.5.1. Functional Requirement**

The source of any given resource must be retrievable.

### **5.5.2. Rationale**

There are many cases where the source stored on a server does not correspond to the actual entity transmitted in response to an HTTP GET. Current known cases are server side include directives, and Standard Generalized Markup Language (SGML) source which is converted on the fly to HyperText Markup Language (HTML) [[HTML](#)] output

entities. There are many possible cases, such as automatic conversion



of bitmap images into several variant bitmap media types (e.g. GIF, JPEG), and automatic conversion of an application's native media type into HTML. As an example of this last case, a word processor could store its native media type on a server which automatically converts it to HTML. A GET of this resource would retrieve the HTML. Retrieving the source would retrieve the word processor native format.

## **5.6. Partial Write.**

### **5.6.1. Functional Requirement**

After editing a resource, it must be possible to write only the changes to the resource, rather than retransmitting the entire resource.

### **5.6.2. Rationale**

During distributed editing which occurs over wide geographic separations and/or over low bandwidth connections, it is extremely inefficient and frustrating to rewrite a large resource after minor changes, such as a one-character spelling correction. Support is needed for transmitting "insert" (e.g., add this sentence in the middle of a document) and "delete" (e.g. remove this paragraph from the middle of a document) style updates. Support for partial resource updates will make small edits more efficient, and allow distributed authoring tools to scale up for editing large documents.

## **5.7. Name Space Manipulation**

### **5.7.1. Copy**

#### **5.7.1.1. Functional Requirements**

It must be possible to duplicate a resource without a client loading, then resaving the resource. After the copy operation, the content of the destination resource must be octet for octet identical to the content of the source resource. A modification to either resource must not cause a modification to the other.

#### **5.7.1.2. Rationale**

There are many reasons why a resource might need to be duplicated, such as changing ownership, preparing for major modifications, or making a backup. Due to network costs associated with loading and saving a resource, it is far preferable to have a server perform a resource copy than a client. If a copied resource records which resource it is a copy of, then it would be possible for a cache to avoid loading the copied resource if it already locally stores the original.

### **5.7.2. Move/Rename**

#### **5.7.2.1. Functional Requirements**

It must be possible to change the location of a resource without a client loading, then resaving the resource under a different name.

Slein, Vitali, Whitehead, Durand Page 8  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

After the move operation, the content of the resource at its new location must be octet for octet identical to the content of the prior resource. It must no longer be possible to access the resource at its original location.

#### **5.7.2.2. Rationale**

It is often necessary to change the name of a resource, for example due to adoption of a new naming convention, or if a typing error was made entering the name originally. Due to network costs, it is undesirable to perform this operation by loading, then resaving the resource, followed by a delete of the old resource. Similarly, a single rename operation is more efficient than a copy followed by a delete operation. Note that moving a resource is considered the same function as renaming a resource.

### **5.8. Collections**

A collection is a resource that is a container for other resources, including other collections. A resource may belong to a collection either directly or by reference. If a resource belongs to a collection directly, name space operations like copy, move, and delete applied to the collection also apply to the resource. If a resource belongs to a collection by reference, name space operations applied to the collection affect only the reference, not the resource itself.

#### **5.8.1. Functional Requirements**

**5.8.1.1. List Collection.** A listing of all resources in a specific collection must be accessible.

**5.8.1.2. Make Collection.** It must be possible to create a new collection.

**5.8.1.3. Add to Collection.** It must be possible to add a resource to a collection directly or by reference.

**5.8.1.4. Remove from Collection.** It must be possible to remove a resource from a collection.

#### **5.8.2. Rationale**

In [[URL](#)] it states that, "some URL schemes (such as the ftp, http, and file schemes) contain names that can be considered hierarchical." Especially for HTTP servers which directly map all or part of their URL

name space into a filesystem, it is very useful to get a listing of all resources located at a particular hierarchy level. This functionality supports "Save As..." dialog boxes, which provide a listing of the entities at a current hierarchy level, and allow navigation through the hierarchy. It also supports the creation of graphical visualizations (typically as a network) of the hypertext structure among the entities at a hierarchy level, or set of levels. It also supports a tree visualization of the entities and their hierarchy levels.

Slein, Vitali, Whitehead, Durand Page 9  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

In addition, document management systems may want to make their documents accessible through the Web. They typically allow the organization of documents into collections, and so also want their users to be able to view the collection hierarchy through the Web.

There are many instances where there is not a strong correlation between a URL hierarchy level and the notion of a collection. One example is a server in which the URL hierarchy level maps to a computational process which performs some resolution on the name. In this case, the contents of the URL hierarchy level can vary depending on the input to the computation, and the number of resources accessible via the computation can be very large. It does not make sense to implement a directory feature for such a name space. However, the utility of listing the contents of those URL hierarchy levels which do correspond to collections, such as the large number of HTTP servers which map their name space to a filesystem, argue for the inclusion of this capability, despite not being meaningful in all cases. If listing the contents of a URL hierarchy level does not make sense for a particular URL, then a "405 Method Not Allowed" status code could be issued.

The ability to create collections to hold related resources supports management of a name space by packaging its members into small, related clusters. The utility of this capability is demonstrated by the broad implementation of directories in recent operating systems. The ability to create a collection also supports the creation of "Save As..." dialog boxes with "New Level/Folder/Directory" capability, common in many applications.

## **5.9. Versioning**

### **5.9.1. Background and General Principles**

**5.9.1.1. Stability of versions.** Most versioning systems are intended to provide an accurate record of the history of evolution of a document. This accuracy is ensured by the fact that a version eventually becomes "frozen" and immutable. Once a version is frozen, further changes will create new versions rather than modifying the original. In order for caching and persistent references to be properly maintained, a client

must be able to determine that a version has been frozen. Any successful attempt to retrieve a frozen version of a resource will always retrieve exactly the same content, or return an error if that version (or the resource itself) is no longer available.

#### **5.9.1.2. Operations for Creating New Versions**

Version management systems vary greatly in the operations they require, the order of the operations, and how they are combined into atomic functions. In the most complete cases, the logical operations involved are:

- o Reserve existing version
- o Lock existing version
- o Retrieve existing version
- o Request or suggest identifier for new version
- o Write new version
- o Release lock

Slein, Vitali, Whitehead, Durand  
[draft-ietf-webdav-requirements-02.txt](#)

WebDAV Requirements      Page 10  
Aug 1997

- o Release reservation

With the exception of requesting a new version identifier, all of these operations have applications outside of versioning and are either already part of HTTP or are discussed in earlier sections of these requirements. Typically, versioning systems combine reservation, locking, and retrieval -- or some subset of these -- into an atomic checkout function. They combine writing, releasing the lock, and releasing the reservation -- or some subset of these -- into an atomic checkin function. The new version identifier may be assigned either at checkout or at checkin.

The WebDAV extensions must find some balance between allowing versioning servers to adopt whatever policies they wish with regard to these operations and enforcing enough uniformity to keep client implementations simple.

#### **5.9.1.3. The Versioning Model**

Each version typically stands in a "derived from" relationship to its predecessor(s). It is possible to derive several different versions from a single version (branching), and to derive a single version from several versions (merging). Consequently, the collection of related versions forms a directed acyclic graph. In the following discussion, this graph will be called a "version graph". Each node of this graph is a "version" or "member of the version graph". The arcs of the graph capture the "derived from" relationships.

It is also possible for a single resource to participate in multiple version graphs.

The WebDAV extensions should support this versioning model, though particular servers may restrict it in various ways.

**5.9.1.4. Versioning Policies.** Many writers, including Feiler [CM] and Haake and Hicks [VSE], have discussed the notion of versioning styles (referred to here as versioning policies, to reflect the nature of client/server interaction) as one way to think about the different policies that versioning systems implement. Versioning policies include decisions on the shape of version histories (linear or branched), the granularity of change tracking, locking requirements made by a server, etc. The protocol should clearly identify the policies that it dictates and the policies that are left up to versioning system implementors or administrators.

**5.9.1.5. It is possible to version resources of any media type.**

## **5.9.2. Functional Requirements**

**5.9.2.1. Referring to a version graph.** There must be a way to refer to a version graph as a whole.

Some queries and operations apply, not to any one member of a version graph, but to the version graph as a whole. For example, a client may request that an entire graph be moved, or may ask for a version history. In these cases, a way to refer to the whole version

Slein, Vitali, Whitehead, Durand Page 11  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

graph is required.

**5.9.2.2. Referring to a specific member of a version graph.** There must be a way to refer to each member of a version graph. This means that each member of the graph is itself a resource.

Each member of a version graph must be a resource if it is to be possible for a hypertext link to refer to specific version of a page, or for a client to request a specific version of a document for editing.

**5.9.2.3. A client must be able to determine whether a resource is a version graph, or whether a resource is itself a member of a version graph.**

A resource may be a simple, non-versioned resource, or it may be a version graph, or it may be a member of a version graph. A client needs to be able to tell which sort of resource it is accessing.

**5.9.2.4. There must be a way to refer to a server-defined default member of a version graph.**

The server should return a default version of a resource for requests that ask for the default version, as well as for requests where no

specific version information is provided. This is one of the simplest ways to guarantee non-versioning client compatibility. This does not rule out the possibility of a server returning an error when no sensible default exists.

It may also be desirable to be able to refer to other special members of a version graph. For example, there may be a current version for editing that is different from the default version. For a graph with several branches, it may be useful to be able to request the tip version of any branch.

**5.9.2.5. It must be possible, given a reference to a member of a version graph, to find out which version graph(s) that resource belongs to.**

This makes it possible to understand the versioning context of the resource. It makes it possible to retrieve a version history for the graphs to which it belongs, and to browse the version graph. It also supports some comparison operations: It makes it possible to determine whether two references designate members of the same version graph.

**5.9.2.6. Navigation of a version graph.** Given a reference to a member of a version graph, it must be possible to discover and access the following related members of the version graph.

- o root member of the graph
- o predecessor member(s)
- o successor member(s)
- o default member of the graph

It must be possible in some way for a versioning client to access versions related to a resource currently being examined.

**5.9.2.7. Version Topology.** There must be a way to retrieve the complete

Slein, Vitali, Whitehead, Durand Page 12  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

version topology for a version graph, including information about all members of the version graph. The format for this information must be standardized so that the basic information can be used by all clients. Other specialized formats should be accommodated, for servers and clients that require information that cannot be included in the standard topology.

**5.9.2.8. A client must be able to propose a version identifier to be used for a new member of a version graph.** The server may refuse to use the client's suggested version identifier. The server should tell the client what version identifier it has assigned to the new member of the version graph.

**5.9.2.9. A version identifier must be unique across a version graph.**

**5.9.2.10. A client must be able to supply version-specific properties to**

be associated with a new member of a version graph. (See [Section 5.1](#) "Properties" above.) At a minimum, it must be possible to associate comments with the new member, explaining what changes were made.

**5.9.2.11.** A client must be able to query the server for information about a version tree, including which versions are locked, which are reserved for editing, and by whom (Session Tracking).

### **5.9.3. Rationale**

Versioning in the context of the world-wide web offers a variety of benefits:

It provides infrastructure for efficient and controlled management of large evolving web sites. Modern configuration management systems are built on some form of repository that can track the revision history of individual resources, and provide the higher-level tools to manage those saved versions. Basic versioning capabilities are required to support such systems.

It allows parallel development and update of single resources. Since versioning systems register change by creating new objects, they enable simultaneous write access by allowing the creation of variant versions. Many also provide merge support to ease the reverse operation.

It provides a framework for coordinating changes to resources. While specifics vary, most systems provide some method of controlling or tracking access to enable collaborative resource development.

It allows browsing through past and alternative versions of a resource. Frequently the modification and authorship history of a resource is critical information in itself.

It provides stable names that can support externally stored links for annotation and link-server support. Both annotation and link servers frequently need to store stable references to portions of resources that are not under their direct control. By providing stable states of resources, version control systems allow not only stable pointers into those resources, but also well-defined methods to determine the

Slein, Vitali, Whitehead, Durand Page 13  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

relationships of those states of a resource.

It allows explicit semantic representation of single resources with multiple states. A versioning system directly represents the fact that a resource has an explicit history, and a persistent identity across the various states it has had during the course of that history.

### **5.10. Variants**

### **5.10.1. Functional Requirements**

It must be possible to send variants to the server, describing the relationships between the variants and their parent resource. In addition, it must be possible to write and retrieve variants of property labels, property descriptions, and property values.

### **5.10.2. Rationale**

The HTTP working group is addressing problems of content negotiation and retrieval of variants of a resource. To extend this work to an authoring environment, WEBDAV must standardize mechanisms for authors to use when submitting variants to a server. Authors need to be able to provide variants in different file or document formats, for different uses. They need to provide variants optimized for different for different clients and for different output devices. They need to be able to provide variants in different languages in the international environment of the Web. In support of internationalization requirements (See 5.12 below), variants need to be supported not just for the content of resources, but for any information intended for human use, such as property values, labels, and descriptions.

## **5.11. Security**

**5.11.1. Authentication.** The WebDAV specification should state how the WebDAV extensions interoperate with existing authentication schemes, and should make recommendations for using those schemes.

**5.11.2. Access Control.** Access control requirements are specified in a separate access control draft [[AC](#)].

**5.11.3. Interoperability with Security Protocols.** The WebDAV specification should provide a minimal list of security protocols which any compliant server / client should support. These protocols should insure the authenticity of messages and the privacy and integrity of messages in transit.

## **5.12. Internationalization**

### **5.12.1. Character Sets and Languages**

Since Web distributed authoring occurs in a multi-lingual environment, information intended for user comprehension must conform to the IETF Character Set Policy [[CHAR](#)]. This policy addresses character sets and encodings, and language tagging.

### **5.12.2. Rationale**



In the international environment of the Internet, it is important to insure that any information intended for user comprehension can be displayed in a writing system and language agreeable to both the client and the server. The information encompassed by this requirement includes not only the content of resources, but also such things as display names and descriptions of properties, property values, and status messages.

## **6. Acknowledgements**

Our understanding of these issues has emerged as the result of much thoughtful discussion, email, and assistance by many people, who deserve recognition for their effort.

Terry Allen, tallen@sonic.net  
Alan Babich, FileNet, babich@filenet.com  
Dylan Barrell, Open Text, dbarrell@opentext.ch  
Barbara Bazemore, PC DOCS, barbarab@pcdocs.com  
Martin Cagan, Continuuus Software, Marty\_Cagan@continuuus.com  
Steve Carter, Novell, srcarter@novell.com  
Dan Connolly, World Wide Web Consortium, connolly@w3.org  
Jim Cunningham, Netscape, jfc@netscape.com  
Ron Daniel Jr., Los Alamos National Laboratory, rdaniel@lanl.gov  
Mark Day, Lotus, Mark\_Day@lotus.com  
Martin J. Duerst, mduerst@ifi.unizh.ch  
Asad Faizi, Netscape, asad@netscape.com  
Ron Fein, Microsoft, ronfe@microsoft.com  
David Fiander, Mortice Kern Systems, davidf@mks.com  
Roy Fielding, U.C. Irvine, fielding@ics.uci.edu  
Mark Fisher, Thomson Consumer Electronics, FisherM@indy.tce.com  
Yaron Y. Goland, Microsoft, yarong@microsoft.com  
Phill Hallam-Baker, MIT, hallam@ai.mit.edu  
Dennis Hamilton, Xerox PARC, hamilton@parc.xerox.com  
Andre van der Hoek, University of Colorado, Boulder,  
andre@cs.colorado.edu  
Del Jensen, Novell, dcjensen@novell.com  
Gail Kaiser, Columbia University, kaiser@cs.columbia.edu  
Rohit Khare, World Wide Web Consortium, khare@w3.org  
Ora Lassila, Nokia Research Center, ora.lassila@research.nokia.com  
Ben Laurie, A.L. Digital, ben@algroup.co.uk  
Mike Little, Bellcore, little@bellcore.com  
Dave Long, America Online, dave@sb.aol.com  
Larry Masinter, Xerox PARC, masinter@parc.xerox.com  
Murray Maloney, SoftQuad, murray@sq.com  
Jim Miller, World Wide Web Consortium, jmilller@w3.org  
Howard S. Modell, Boeing, howard.s.modell@boeing.com  
Keith Moore, University of Tennessee, Knoxville, moore@cs.utk.edu  
Henrik Frystyk Nielsen, World Wide Web Consortium, frystyk@w3.org  
Jon Radoff, NovaLink, jradoff@novalink.com  
Alan Robertson, alanr@bell-labs.com  
Henry Sanders, Microsoft,

Andrew Schulert, Microsoft, andyschu@microsoft.com  
Christopher Seiwald, Perforce Software, seiwald@perforce.com

Slein, Vitali, Whitehead, Durand Page 15  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Aug 1997

Einar Stefferud, stef@nma.com  
Richard Taylor, U.C. Irvine, taylor@ics.uci.edu  
Robert Thau, MIT, rst@ai.mit.edu  
Sankar Virdhagriswaran, sv@hunchuen.crystaliz.com  
Dan Whelan, FileNet, dan@FILENET.COM  
Gregory J. Woodhouse, gjw@wnetc.com

## 7. References

[AC] J. Radoff, "Requirements for Access Control within Distributed Authoring and Versioning Environments on the World Wide Web".

[CHAR] H.T. Alvestrand, "IETF Policy on Character Sets and Languages", June 1997, working draft, [draft-alvestrand-charset-policy-00.txt](#).

[CM] P. Feiler, "Configuration Management Models in Commercial Environments", Software Engineering Institute Technical Report CMU/SEI-91-TR-7,  
<<http://www.sei.cmu.edu/products/publications/91.reports/91.tr.007.html>>

[HTML] T. Berners-Lee, D. Connolly, "HyperText Markup Language Specification - 2.0", [RFC 1866](#), MIT/LCS, November 1995.

[HTTP] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, and **T. Berners-Lee**, "Hypertext Transfer Protocol -- HTTP/1.1", [RFC 2068](#), U.C. Irvine, DEC, MIT/LCS, January 1997.

[ISO 10646] ISO/IEC 10646-1:1993. "International Standard -- Information Technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane."

[URL] T. Berners-Lee, L. Masinter, M. McCahill. "Uniform Resource Locators (URL)", [RFC 1738](#), CERN, Xerox PARC, University of Minnesota, December 1994.

[VSE] A. Haake, D. Hicks, "VerSE: Towards Hypertext Versioning Styles", Proc. Hypertext'96, The Seventh ACM Conference on Hypertext, 1996, pages 224-234.

## 8. Authors' Addresses

Judith Slein  
Xerox Corporation  
**800 Phillips Road 128-29E**  
Webster, NY 14580

E-Mail: slein@wrc.xerox.com

Fabio Vitali  
Department of Computer Science  
University of Bologna  
ITALY

E-Mail: fabio@cs.unibo.it

Slein, Vitali, Whitehead, Durand  
[draft-ietf-webdav-requirements-02.txt](#) WebDAV Requirements Page 16  
Aug 1997

**E. James Whitehead, Jr.**  
Department of Information and Computer Science  
University of California  
Irvine, CA 92697-3425

Fax: 714-824-4056  
E-Mail: ejw@ics.uci.edu

David G. Durand  
Department of Computer Science  
Boston University  
Boston, MA

E-Mail: dgd@cs.bu.edu

Expires February 28, 1998

Slein, Vitali, Whitehead, Durand Page 17