INTERNET-DRAFT Geoffrey Clemm, Rational Software
draft-ietf-webdav-versioning-02.txt Chris Kaler, Microsoft

Expires December 25, 1999 June 25, 1999

Versioning Extensions to WebDAV

Status of this Memo.

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

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

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

Abstract

This document specifies a set of methods, headers, and resource-types that define the WebDAV Versioning extensions to the HTTP/1.1 protocol. WebDAV Versioning will minimize the complexity of clients so as to facilitate widespread deployment of applications capable of utilizing the WebDAV Versioning services. WebDAV Versioning includes:

- Automatic versioning support for versioning-unaware clients,

- Linear versioning , and

- Support for parallel development and configuration management.

Table of Contents

1 INTRODUCTION

This document defines DAV Versioning extensions, an application of HTTP/1.1 for handling resource versioning in a DAV environment. [VerGoal] describes the motivation and requirements for DAV Versioning.

DAV Versioning will minimize the complexity of clients so as to facilitate widespread deployment of applications capable of utilizing the DAV services. As well, DAV Versioning supports a rich level of versioning options for versioning-aware clients.

DAV Versioning consists of:

- Automatic versioning support for versioning-unaware clients,

- Linear versioning , and

- Support for parallel development and configuration management.

1.1 Relationship to DAV

To maximize interoperability and use of existing protocol functionality, versioning support is designed as extensions to the WebDAV [RFC2518] and advanced-collection protocols [AdvCol]. In particular, DAV Versioning relies on the resource and property model defined by [RFC2518] and the binding model defined by [AdvCol]. The versioning protocol is designed so that WebDAV

locking (class 2) support is optional. The effect of a lock on versioning methods and content-types will be defined to provide interoperability of servers that provide locking support.

Versioning support is defined in the form of Core versioning support, supplemented by a set of orthogonal extensions to the Core: Activity, Merging, Configuration, Versioned Collection, and Baseline versioning support. Core support provides versioning of largely independent resources. It allows authors to concurrently create and access distinct revisions of a resource. Activity support extends Core support with logical change tracking and management through activities. Merging support extends Core support with conflict detection and resolution through merging. Configuration support extends Core support with the creation of sets of consistent revisions. Versioned Collection support extends Core support with the ability to version the URL namespace. Baseline support extends Configuration and Versioned Collection support with the ability to create and compare configurations of all revisions in a URL subtree.

Throughout this specification the [xyz] notation is used to indicate that a property is introduced at the "xyz" level of versioning support. The levels of versioning support provided by a server can be discovered via an OPTIONS request.

1.2 Terms

This draft uses the terms defined in [RFC2068] and [RFC2518].

1.3 Notational Conventions

The augmented BNF used by this document to describe protocol elements is exactly the same as the one described in Section 2.1 of [RFC2068]. Because this augmented BNF uses the basic production rules provided in Section 2.2 of [RFC2068], 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].

2 CONCEPTS AND DEFINITIONS

The section presents the versioning concepts and terms/definitions used in this protocol.

Versionable Resource

A versionable resource is a resource that can be placed under version control. A null resource is a versionable resource.

Versioned Resource

A versioned resource is a resource that is under version control.

To update a resource under version control, it must be checked out and then subsequently checked in. The checked in states of a versioned resource are saved by the server to capture the history of that resource.

Revision

A revision contains a particular state of a versioned resource. An immutable revision is a revision whose body and immutable properties cannot be modified. A mutable revision is a revision whose state can be overwritten by a subsequent check in request.

Revision Name

A revision name is a name that can be used to identify a single revision of a versioned resource. There are two types of revision names: revision identifiers and revision labels.

Revision Identifier

A revision identifier (or revision ID) is a revision name that is assigned by the server when the revision is created and cannot be changed to refer to a different revision.

Revision Label

A revision label is a revision name that is assigned by a client and may later be changed to refer to a different revision. The same label may be assigned to many different versioned resources.

Initial Revision

An initial revision is the first revision of a versioned resource.

Predecessor, Successor

A predecessor of a revision is the previous revision that was checked out to create the revision. A successor of a revision is a revision whose predecessor is that revision. Each revision has a single predecessor (except for the initial revision which has no predecessor) and zero or more successors.

Merge Predecessor, Merge Successor

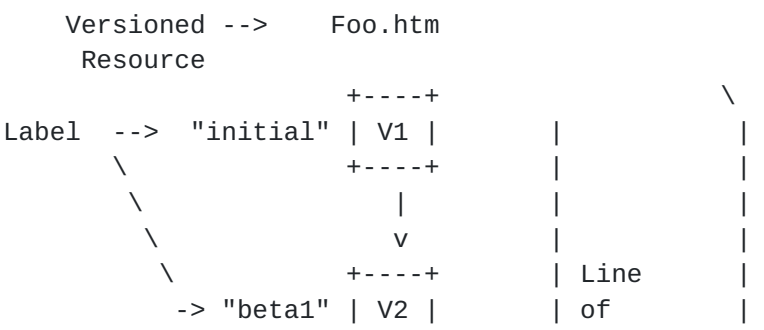
A merge predecessor of a revision is a revision that has been merged into this revision. A merge successor of a revision is a revision into which that revision has been merged. A revision can have zero or more merge predecessors and zero or more merge successors.

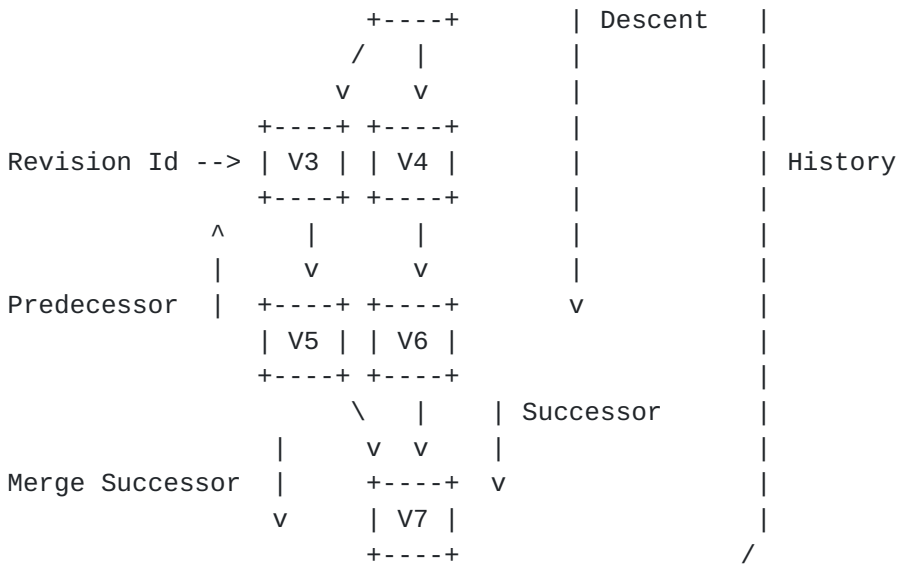
Line Of Descent

A line of descent to a specified revision is a sequence of revisions connected by successor relationships from the initial revision to that revision.

The following diagram illustrates several of the previous

definitions.





Immutable/Mutable Property

An immutable property is a property of an immutable revision that cannot be changed. A mutable property is a property of an immutable revision that can be changed. Only properties of revisions can be immutable or mutable.

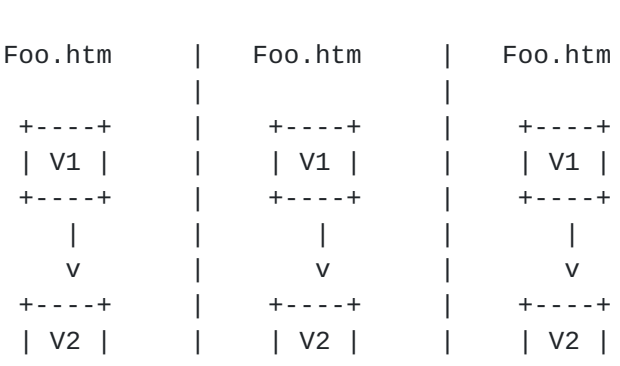
Working Resource

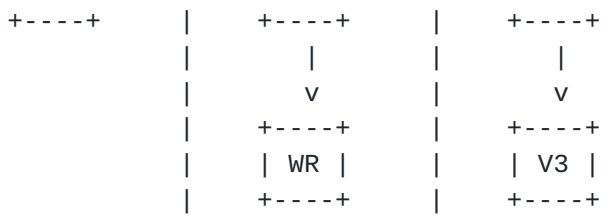
A working resource is an editable resource created by checking out a revision of a versioned resource.

Checkout/Checkin Model

The checkout/checkin model is the process by which updates are made to a versioned resource. A versioned resource is checked out to create an editable working resource. The working resource is updated or augmented as desired, and then checked in to make it part of the revision history of the versioned resource.

The following diagram illustrates the checkout/checkin process.

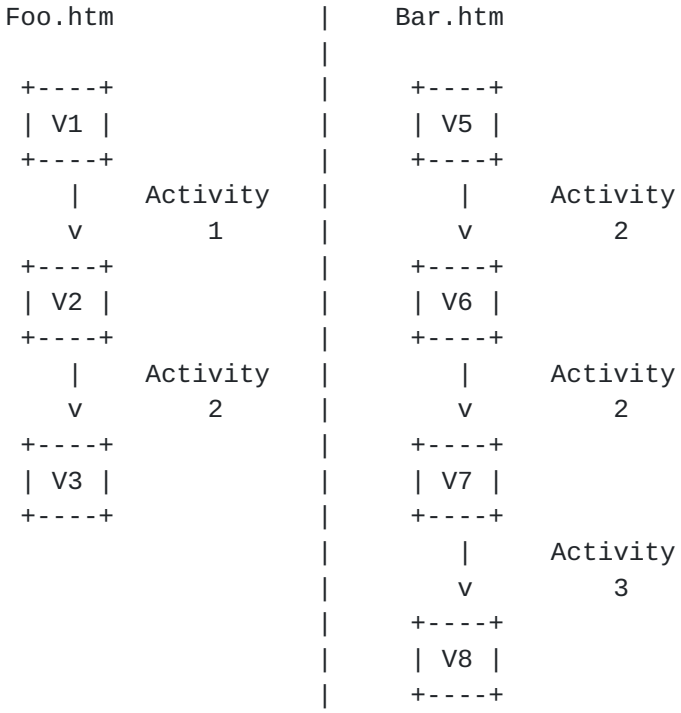



+----+
|
+----+
|
+----+
|
|
|
v
|
v
|
+----+
|
+----+
|
| WR |
|
| V3 |
|
+----+
|
+----+

Activity

An activity is a resource that selects a set of revisions that correspond to some unit of work or conceptual change. An activity can contain revisions of multiple versioned resources, and multiple revisions of the same versioned resource. If an activity contains multiple revisions of the same versioned resource, all of those revisions must be on a single line of descent to one of those revisions, and this revision is called the latest revision selected by that activity for that versioned resource.

The following diagram illustrates activities. Revision V3 is the latest revision of Foo.htm selected by activity 2, and revision V7 is the latest revision of Bar.htm selected by activity 2.



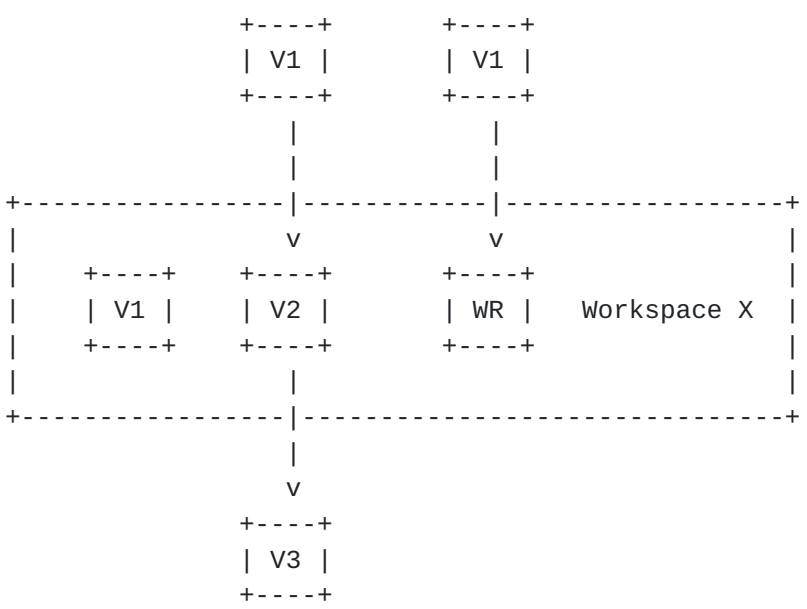
Workspace

A workspace is a resource that is used to identify working resources. A workspace can also contain a revision selection rule that specifies what revision of an arbitrary versioned resource should be accessed when the resource is referenced in the context of that workspace. A revision selection rule provides revision selection based on criteria such as revision name, latest in an activity, and membership in a configuration.

A workspace that does not contain a revision selection rule (and therefore can only be used to identify working resources) is called a basic workspace. A workspace that contains a revision selection

rule (and therefore can be used to specify revision selection) is called an extended workspace.

The following diagram illustrates an extended workspace.



Target

The target of a versioned resource is the working resource or revision of that versioned resource that is selected by the current workspace.

Conflict Report

A conflict report lists all versioned resources for which the revision selection rule of a workspace selects multiple revisions on different lines of descent. A conflict is resolved by checking out one of the selected revisions, modifying the resulting working resource so that it represents the logical merge of all selected revisions, and then indicating these merges by adding these revisions as merge predecessors of the working resource. Checking in this working resource creates a new revision that resolves the conflict.

Default Workspace

A server MUST provide a default workspace that is used to perform version selection for versioning-unaware clients. The revision selection rule of the default workspace MAY be a modifiable by a client.

Default Target

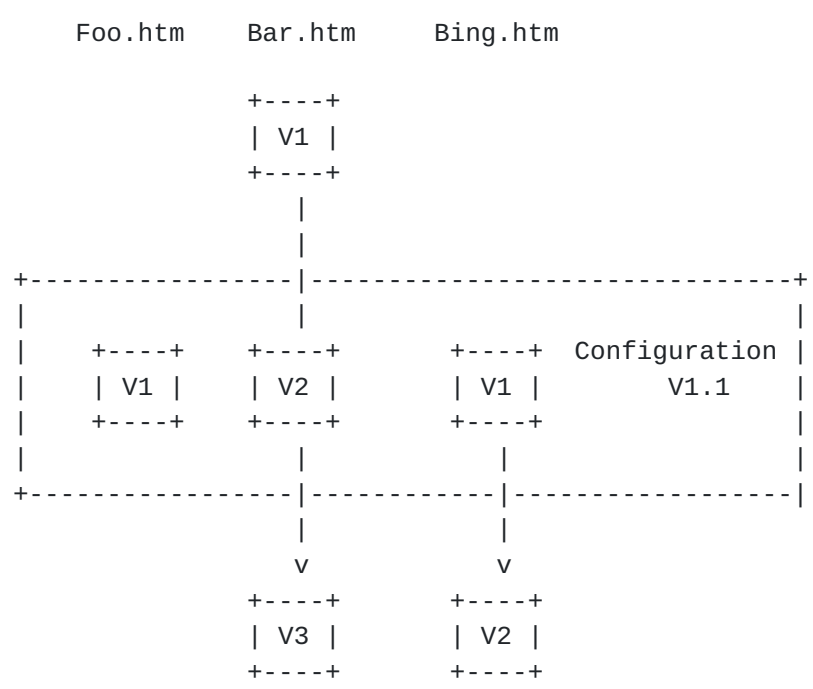
The default target of a versioned resource is the target selected by the default workspace.

Configuration

A configuration selects a particular revision from each of a set of versioned resources. Unlike a workspace, which can select both working resources and revisions, a configuration can only select revisions. Also, while the revision selected by a workspace for a versioned resource can change as a result of a change to the versioned resource (such as adding a label), the revision selected by a configuration can change only by explicitly modifying the

configuration itself. Unlike an activity, a configuration can select at most one revisions of a given versioned resource. Unlike both a workspace and an activity, a configuration can be versioned.

The following diagram illustrates a configuration.



Versioned Collection

A versioned collection is a type of versioned resource that results from placing a collection under version control. The members of a versioned collection revision are all versioned resources.

Baselined Collection

A baselined collection is a special type of versioned collection that is associated with a versioned configuration. A revision of the associated versioned configuration is called a baseline of the baselined collection. A baseline contains a revision of the versioned collection and a revision of every member of every versioned collection revision in that baseline.

History Resource

A history resource for a versioned resource contains all revisions of that versioned resource.

3 WEBDAV VERSIONING SEMANTICS

3.1 Creating Versioned Resources

A resource may or may not be versioned. When a resource is created by a PUT or MKRESOURCE request, it is commonly created as an unversioned resource. Some unversioned resources can be put under version control; these are called versionable resources. After a

resource is put under version control, it becomes a versioned resource, and an initial revision is created that is a copy of the versionable resource. This initial revision becomes the target of the versioned resource.

3.2 Modifying a Versioned Resource

A versioned resource must be checked out before it can be modified. Checking out a versioned resource produces a new resource, called a working resource. A working resource is a modifiable copy of the revision, and is identical to an unversioned resource in all respects other than that it is associated with a particular revision of a particular versioned resource. It may be edited by setting its properties or contents any number of times. When the author is satisfied that the working resource is in a state that should be retained in the versioned resource history, the author checks in the working resource to create a new revision of the versioned resource. The revision that was checked out is the predecessor of the new revision.

The use of checkout/checkin and working resources to update a versioned resource addresses the lost update problem by ensuring that each author is updating his or her own working resource rather than a shared resource, and by ensuring that the predecessor of the updated resource is reliably tracked. Authors can use checkout/checkin to register intent to modify a versioned resource similar to the way lock /unlock are used in WebDAV level 2, but the underlying semantics are very different. With lock/unlock, work is serialized and avoiding lost updates depends on clients using the lock/unlock protocol. With checkout/checkin, work can be performed in parallel, and the server prevents lost updates by retaining all saved states (revisions) of the resource.

A revision may be created as either immutable or mutable. An immutable revision cannot be changed and provides a stable environment for history management, change recovery, merging, and configuration management. A mutable revision is more suitable for situations where versioning is treated more informally, and it is not necessary or desirable to maintain strict version histories, or to guarantee the possibility of backtracking to a previous saved state. If the revision is mutable, the author may request that a subsequent checkin should overwrite the revision that was checked out, instead of creating a new revision. In this case, the previous state captured by that revision is lost. Servers may choose to support only immutable revisions, only mutable revisions, or both.

3.3 Naming Revisions: Revision Ids and Labels

Revision names are used to distinguish a revision of a versioned resource from other revisions of that versioned resource. A revision name is either a revision id or any number of revision labels. A revision of a versioned resource is given a server assigned revision id when it is created. This revision id acts as a

persistent, immutable identifier distinguishing this revision from all others of the same versioned resource. The revision id cannot be changed, assigned to another revision, or reused.

A user may assign revision labels to a revision in order to provide a more meaningful name for the revision. A given revision label can be assigned to at most one revision of a given versioned resource, but may be reassigned to any revision of the versioned resource at any time. Revisions of different versioned resources may have the same label.

3.4 Parallel Development and Activities

In a locking model, when a resource is locked for modifications by one author, all other authors are supposed to respect that lock and not work on a copy of that resource until the lock has been released. To avoid the work serialization inherent in the locking model, a versioning model allows multiple authors in different workspaces to check out the same revision at the same time, work on their respective working resources in parallel, check in their respective working resources as soon as their changes are complete, and then merge the resulting revisions at some later time.

Although a simple versioning model works well for isolated changes to independent resources, the required merge process becomes unmanageable when sequences of inter-related changes are performed on sets of related resource. To address this merge problem, resources can be checked out in the context of an activity. An activity captures the set of revisions that form a set of related changes an author is making to one or more versioned resources. Each activity represents a thread of development, where any thread can be isolated from other threads to provide a stable environment for parallel work. In case parallel work on isolated activities results in branches in the underlying versioned resource histories, the activities can be unified through a merge operation.

3.5 Revision Selection and Workspaces

Resources, working resources, and revisions of versioned resources are all accessed using a URL. When a client accesses a versioned resource, it is necessary to provide additional information to specify which working resource or which revision of the versioned resource should be accessed. Specifying the resource URL and a revision name can access a specific revision of the versioned resource. However, this requires the user to add and remember labels for each revision; it does not provide a way of accessing a consistent set of revisions captured by an activity or contained in a configuration; it does not enable non-versioning aware clients to access revisions; and it does not provide a client with access to a

working resource of a versioned resource.

To address the restrictions inherent in revision-name based revision selection, the revision selected when a specific revision name is not specified is resolved through a workspace. A workspace

is a resource whose primary purpose is to identify working resources. If the workspace contains no working resource for a given versioned resource, it can also be used to select an appropriate revision of the versioned resource. This allows versioned resources and unversioned resources to be accessed consistently by both versioning-aware and versioning-unaware clients.

In order to specify revision selection, a workspace contains a revision selection rule. A revision selection rule can specify revision names, activities, configurations, or use operators that combine several of these rule elements. A revision name selects a revision with that name. An activity selects the latest revision that was created in that activity. Configurations select the revision contained in the configuration. The "or" operator contains a sequence of rule elements, and applies them in order until the first match is found. If there is no matching revision, then a resource-not-found status is returned.

If a request is made and no workspace is specified, a server determined default workspace is used. This is the workspace used by all versioning-unaware clients. A server MAY allow modifications to the revision selection rule of the default workspace.

3.6 Configurations

A workspace selects a volatile set of revisions. Changes to selected activities or changes to labels may result in the selection of different revisions. A configuration is a resource that selects a set of immutable revisions. A workspace whose version selection rule contains a configuration will always select the same revisions as long as the configuration is not modified and no checkouts are performed in that workspace.

Configurations are convenient for defining a persistent set of revisions that relate to each other in some specific way at some point in time. This can be useful for a variety of purposes such as publishing consistent versions of resources to deploy an application, or for recovering a specific state for legal or maintenance reasons.

3.7 Versioned Collections

A collection contains a set of named bindings to other resources that are the members of that collection. For a versioned collection, the bindings are to versioned resources, not to particular revisions. To modify the state of a versioned collection (i.e. add or remove a binding), the versioned collection must be checked out, just like any other versioned resource. Requests that

modify the state of a collection member, such as checking it out or checking in a new revision, have no effect on the state of the collection. Conversely, requests that modify the state of a versioned collection, such as deleting or adding a binding to resource, have no effect on the state of that resource. In

particular, the resource will remain bound in any other revisions of the collection of which it was a member.

If a URL identifies a sequence of nested versioned collections, revision selection is performed for each versioned collection in the sequence, to select the versioned collection revision that will be used to map the next segment of the URL to the appropriate versioned resource.

4 VERSIONING RESOURCE TYPES AND PROPERTIES

This section defines the new resource types and properties introduced by WebDAV versioning.

4.1 Property Attributes

There are several important attributes of properties that will be defined for every property introduced by this document.

4.1.1 Writeable/Readonly Properties

A writeable property can be modified by a client, while a readonly property can only be modified by the server.

All properties defined in this document are writeable unless explicitly marked as "readonly".

4.1.2 Immutable/Mutable Properties

An immutable resource is a resource whose value cannot change. An immutable property is a property whose value cannot change when it appears on an immutable resource. A mutable property is a property whose value can change, even when it appears on an immutable resource.

All properties defined in this document are immutable unless explicitly marked as "mutable".

4.1.3 Property Resources

There are various properties whose contents can be represented as an HTTP resource. Doing so allows a client to use the full set of HTTP methods to manipulate the contents of that property, rather than being limited to the functionality provided by PROPFIND and PROPPATCH. This is particularly valuable for a property value that is naturally represented as a collection resource. By default, PROPFIND returns an XML document as the value of a property resource; however, when a DAV:property-resource-URL element is specified in the PROPFIND request body, PROPFIND will return the URL of the property resource. Servers MAY support DAV:property-

resource-URL for a property, but MUST support the default XML value.

All properties that are property resources are explicitly marked as "resource". If the property resource is a collection, the property is marked as "collection".

4.2 Resource Properties

WebDAV versioning introduces the following additional properties for a resource:

4.2.1 DAV:author (immutable) [Core]

This property is used to track the author of a resource.

4.2.2 DAV:comment (immutable) [Core]

This property is used to track a brief comment about a resource.

4.3 Revision Properties

WebDAV versioning introduces the following additional properties for a revision:

4.3.1 DAV:revision-id (readonly) [Core]

The DAV:revision-id is an identifier assigned to a revision by the server. Whenever a revision is created or modified by a CHECKIN, it must be assigned a new DAV:revision-id. A revision cannot be given a DAV:revision-id that has been given to any other revision of that versioned resource, even a revision that has been deleted.

4.3.2 DAV:predecessor (readonly, resource) [Core]

The DAV:predecessor of a revision is the revision that was checked out to produce a working resource that was checked in to produce this revision. The XML document for DAV:predecessor contains the revision-id of the DAV:predecessor.

4.3.3 DAV:successors (readonly, mutable, collection) [Core]

The DAV:successors collection of a revision contains the revisions whose DAV:predecessor is that revision. The XML document for DAV:successors contains a list of the revision-id's of the DAV:successors.

4.3.4 DAV:single-checkout (mutable) [Core]

When the DAV:single-checkout property of a revision is set, only one working resource can be checked out from that revision at any time.

4.3.5 DAV:auto-version (mutable) [Core]

When the DAV:auto-version property of a revision is set, a PUT request (or any request that modifies an immutable aspect of the revision) to this revision is automatically preceded by a CHECKOUT into the default workspace and followed by a CHECKIN. This allows a versioning-unaware client to modify a version-controlled resource. The DAV:auto-version value can take the same values as the DAV:checkin-policy of a working resource, and the DAV:checkin-policy of the automatically created working resource is set to be the DAV:auto-version policy of the revision.

4.3.6 DAV:revision-labels (mutable) [Core]

This property is used to identify labels that are associated with a specific revision.

4.3.7 DAV:checkin-date (readonly) [Core]

This property contains the date when the revision was checked in. This property is automatically assigned and is formatted using ISO8061.

4.3.8 DAV:working-resources (readonly, collection) [Core]

This property is a collection of the workspaces that contain working resources whose DAV:predecessor is this revision. The XML document for DAV:working-resources contains a description of these working resources.

4.3.9 DAV:history-uuid (readonly) [Core]

The DAV:history-uuid of a revision is the DAV:uuid of the history resource whose DAV:revisions collection contains this revision.

4.3.10 DAV:history (readonly, resource) [Core]

The DAV:history of a revision is the history resource whose DAV:revisions collection contains this revision. The XML document for DAV:history contains a description of revisions that form the line-of-descent to this revision.

4.3.11 DAV:merge-predecessors (mutable, collection) [Merging]

The DAV:merge-predecessors collection of a revision contains the revisions whose contents were explicitly merged by the client into that revision. The client is free to add or delete members to this collection to more accurately reflect the contents of that revision. The XML document for DAV:merge-predecessors contains the revision id's of the DAV:merge-predecessors.

4.3.12 DAV:merge-successors (mutable, collection, readonly) [Merging]

The DAV:merge-successors collection of a revision contains a binding to each revision whose DAV:merge-predecessors collection contains a binding to that revision. The XML document for DAV:merge-successors contains the revision id's of the DAV:merge-successors.

4.4 Working Resource Properties

WebDAV versioning introduces the following additional properties for a working resource:

4.4.1 DAV:workspace (readonly, resource) [Core]

The DAV:workspace of a working resource is the workspace that contains this working resource. The XML document for DAV:workspace contains the URL of this workspace.

4.4.2 DAV:predecessor (read-only, resource) [Core]

This property contains the revision that was checked out to create this working resource. The XML document for DAV:predecessor contains the revision id of DAV:predecessor.

4.4.3 DAV:checkin-policy [Core]

The DAV:checkin-policy property of a working resource indicates how this working resource should be checked in. The following are defined values for DAV:checkin-policy. The default value is DAV:immutable.

DAV:identical-abort - the CHECKIN should fail if the working resource is identical to its DAV:predecessor.

DAV:identical-uncheckout - an UNCHECKOUT should be applied instead of CHECKIN if the working resource is identical to its DAV:predecessor.

DAV:overwrite - the working resource should be copied into its DAV:predecessor instead of creating a new revision.

DAV:mutable - a new revision is created that can be overwritten by a subsequent DAV:overwrite CHECKIN.

DAV:immutable - a new revision is created that cannot be overwritten by a subsequent DAV:overwrite CHECKIN.

DAV:keep-checked-out - create a new revision but does not delete the working resource. The DAV:predecessor of the working resource

is changed to be the new revision.

DAV:baseline - instead of creating a new revision of the versioned collection, create a new baseline for it (the CHECKIN fails unless

it is applied to a versioned collection with a DAV:baselines property).

4.4.4 DAV:merge-predecessors (collection) [Merging]

The DAV:merge-predecessors collection of a working resource contains the revisions whose contents were explicitly merged by the client into that working resource. The client adds and deletes members of this collection to reflect the set of revisions that were merged by the client into the working resource. The name of a DAV:merge-predecessors binding is the DAV:revision-id of that revision. The XML document for DAV:merge-predecessors contains the revision id's of the DAV:merge-predecessors.

4.4.5 DAV:activity (readonly, resource) [Activity]

The DAV:activity property of a working resource is the DAV:activity of the workspace when the working resource was checked out. The XML document for DAV:activity is the URL for the activity.

4.5 Workspace Properties

WebDAV versioning introduces the following additional properties for a workspace:

4.5.1 DAV:working-resources (readonly, collection) [Core]

The DAV:working-resources collection contains the versioned resources that are checked out into this workspace. The XML document for DAV:working-resources contains a description of these working resources.

4.5.2 DAV:revision-selection-rule [Core]

The DAV:revision-selection-rule of a workspace can contain an XML document that describes how revision selection will be performed in that workspace. The working resources checked out into a workspace take priority over revisions selected by the revision selection rule, thus target of a versioned resource in a workspace is the working resource in that workspace for that versioned resource, else the revision selected by the workspace revision selection rule. To ensure that working resources continue to be visible in a workspace after they are checked in, the DAV:label or DAV:activity of a workspace is usually the first element of the DAV:revision-selection-rule. If the DAV:revision-selection-rule is not set or is empty, the revision selection rule will select no revision for any versioned resource.

Standard revision selection rule elements are defined in this document, but additional revision selection rule elements may be

supported by a WebDAV server.

4.5.3 DAV:label [Core]

The DAV:label property of a workspace can contain a revision label. When a working resource in a workspace is checked in, it will be given this label.

4.5.4 DAV:activity [Activity]

The DAV:activity property of a workspace is the activity that is currently being performed in that workspace. A new working resource in a workspace will have its DAV:activity property set to this activity. The XML document for DAV:activity contains the URL of DAV:activity.

4.6 Activity Properties

WebDAV versioning introduces the following additional properties for an activity:

4.6.1 DAV:revisions (readonly, collection) [Activity]

The DAV:revisions collection of an activity contains all revisions whose DAV:activity property contains this activity. The XML document for DAV:revisions contains the URL's of these revisions.

4.6.2 DAV:required-activities (collection) [Activity]

The DAV:required-activities collection of an activity contains the other activities that form a part of the logical change being captured by the activity. The DAV:needed-activities of an activity contribute to the revision selection behavior of that activity when it is used in a revision selection rule. The purpose of this property is to identify other activities that are a prerequisite to this activity. The XML document for DAV:required-activities contains the URL's of these activities.

4.6.3 DAV:workspace (resource) [Activty]

The DAV:workspace property of an activity contains the workspace that currently has that activity as its DAV:current activity. This implies that at most one workspace can be working in a given activity at a time. If any working resource of a workspace is checked out into a given activity, the DAV:workspace of that activity can only be that workspace. The XML document for DAV:workspace contains the URL of the workspace.

4.7 Configuration Properties

WebDAV versioning introduces the following properties for a configuration:

4.7.1 DAV:roots (immutable, collection) [Configuration]

The DAV:roots collection of a configuration contains the versioned resources that are not named by versioned collection revisions in that configuration. The XML document for DAV:roots contains the URL's of these versioned resources.

4.8 Versioned Collection Properties

WebDAV versioning introduces the following additional properties for a versioned collection:

4.8.1 DAV:baselines (resource) [Baseline]

The DAV:baselines of a versioned collection is a versioned configuration whose DAV:roots contains only that versioned collection. A revision of the DAV:baselines of a versioned collection effectively provides a "deep-revision" of that versioned collection. The XML document for DAV:baselines contains the URL of the versioned configuration.

4.9 History Resource Properties

WebDAV versioning introduces the following additional properties for a history resource:

4.9.1 DAV:uuid (readonly) [Core]

The DAV:uuid is an identifier assigned to a history resource by the server. A history resource cannot be given a DAV:uuid that has been given to any other history resource, even a history resource that has been deleted.

4.9.2 DAV:revisions (readonly, collection) [Core]

The DAV:revisions collection of a history resource contains all revisions of that history resource, where the name of a revision in the DAV:revisions collection is its DAV:revision-id. If a revision id contains a URL reserved character, that character is escaped in the DAV:revisions name. The XML document for DAV:revisions contains the URL's of the revisions.

4.9.3 DAV:revision-labels (collection) [Core]

The DAV:revision-labels collection of a history resource contains a binding for each label assigned to any revision of that history resource, where the binding name is that label (reserved URL characters are escaped) and the binding is to the revision selected by that label. The client can label and unlabel revisions by adding and deleting members of the DAV:revision-labels collection.

The XML document for DAV:revision-labels contains the URL's of the members of the DAV:revision-labels collection.

5 VERSIONING METHODS

5.1 Existing Methods

This section describes the extensions to the existing WebDAV methods. Under versioning, the methods inherit all of the WebDAV functionality with the following extensions.

5.1.1 GET

When GET is applied to a versioned resource, it returns the body of the target of that versioned resource. The result of GET on a workspace, activity, or history resource is undefined.

5.1.2 BIND

When BIND creates a binding in a working resource for a versioned collection, it MUST fail if the request URL of the BIND is not a versioned resource.

5.1.3 PUT

When PUT is applied to a versioned resource whose target is a working resource, the PUT is applied to that working resource. When PUT is applied to a versioned resource whose target is a revision, the PUT MUST fail except in the following two cases. If the revision has a DAV:auto-version property and no Target-Selector header has been specified, the revision is checked out into the default workspace, the PUT is applied to the resulting working resource, and the working resource is checked in. If the revision is a revision of a baselined collection and the value of DAV:checkin-policy is DAV:baseline, a new revision of the DAV:baselines configuration is created by the CHECKIN.

When PUT is applied directly to a revision (i.e. not indirectly via a versioned resource), it MUST fail.

When PUT is applied to a null resource that is an internal member of a versioned collection whose target is a working resource, a new versioned resource is created at the request URL of the PUT. When the target is a versioned collection revision, the PUT request fails unless the revision has a DAV:auto-version property and no Target-Selector header has been specified. If DAV:auto-version is set, the versioned collection revision is checked out into the default workspace, a new versioned resource is created as a member of the working collection, and the working collection is checked in.

When a PUT is applied to a workspace, activity or history resource, it fails.

5.1.4 PROPFIND

If a DAV:property-resource-URL is specified under a DAV:prop element in a PROPFIND request body, the property resource URL of that property will be returned in the PROPFIND response instead of the default XML document for that property resource. If a DAV:property-resource-URL is specified directly under the DAV:propfind element, the property resource URL will be returned for all of the property resources in the PROPFIND response.

5.1.5 PROPPATCH

When PROPPATCH is applied to a versioned resource, its behavior is similar to that of PUT. In particular, when PROPPATCH is applied to an immutable property of a revision, it MUST fail unless the revision has a DAV:auto-version property.

5.1.6 DELETE

When DELETE is applied to a versioned resource whose target is a revision, the versioned resource is deleted from the collection that contains it, but the revision is unaffected. When DELETE is applied to a workspace, the workspace and all working resources of that workspace are deleted.

When DELETE is applied to a member of the DAV:revisions collection of a history resource, it fails unless the All-Bindings header is specified. When DELETE is applied to a history resource, the history resource and all members of the DAV:revisions collection of that history resource are deleted.

5.1.7 COPY

When COPY is applied to a versioned resource, it is applied to the target of the versioned resource. That is, only the selected revision is copied, not the full version history.

When a COPY request is applied to a workspace, activity, or history resource, it fails.

5.1.8 MOVE

When MOVE is applied to a versioned resource, the MOVE request MUST fail unless a binding to that versioned resource can be created at the Destination of the MOVE.

When a MOVE request is applied to an activity or a history

resource, it fails.

Any request to MOVE a specific revision, and not the versioned resource, MUST fail.

5.1.9 LOCK

A write lock on a versioned resource is applied to the target of that versioned resource.

A write lock on a revision prevents unauthorized modifications to the properties of that revision.

A write lock on a working resource prevents unauthorized changes to the body or properties of that working resource.

A write lock on an activity prevents unauthorized modifications to the properties of that activity.

A write lock on a history resource places a write lock on all revisions of that history resource.

A write lock on a workspace prevents unauthorized modifications to the properties of that workspace.

5.1.10 OPTIONS

The OPTIONS method allows the client to discover the level of versioning support provided by a server.

The following defines the BNF for the Versioning header:

```
Versioning    := "Versioning" ":" Ver-type-list
Ver-type-list := Ver-type | (Ver-type-list "," Ver-type)
Ver-type      := "Core"
              | "Activity"
                 | "Merging"
              | "Configuration"
                 | "Versioned-Collection"
                 | "Baseline"
```

5.2 New Methods

WebDAV versioning introduces two new methods, MKRESOURCE and REPORT, that are not specific to versioning but are needed to support the versioning extensions.

5.2.1 MKRESOURCE

The MKRESOURCE method requests the simultaneous creation of a resource, identified by the Request URI, and initialization of its properties. Creation of the resource and initialization of its

properties MUST both occur, or neither occurs. The request message body of the MKRESOURCE method is the same as for the PROPPATCH method, i.e. it MUST contain the DAV:propertyupdate XML element, defined in section 12.13 of [RFC2518]. The property update directives in the request message body provide the initial values of the properties of the new resource. The type of the created

resource is specified by the DAV:resourcetype property, if present. If the DAV:resourcetype property is not specified, the created resource is an ordinary (i.e. untyped) resource. Like PROPPATCH, instruction processing MUST occur in the order instructions are received (i.e. from top to bottom). Instructions MUST all be executed, or none executed. If MKRESOURCE is applied to an existing resource, that resource is deleted prior to MKRESOURCE processing. The behavior of MKRESOURCE on an existing resource can be explicitly controlled through use of the Overwrite header.

MKRESOURCE can be used to create a new activity in a repository DAV:activities collection. When MKRESOURCE is used to create a repository from an existing versionable collection, every member of that versionable collection is also placed under version control as a history resource in that repository.

Status Codes:

201 (Created) - The new resource has successfully been created.

207 (Multi-Status) - If any error was encountered in the creation of the resource, this response is generated. Status codes defined for use with PROPPATCH (defined in section 8.2.1 of [RFC2518]) SHOULD be used to represent error cases in setting the value of properties.

TBD - Explain effect on existing resource types

TBD - Give request/response example

5.2.2 REPORT

The REPORT request is an extensible mechanism for obtaining information about a resource. This differs from OPTIONS because the information is not static. That is, it is typically a report that requires server processing in order to generate.

The REPORT method takes an XML body document that specifies the type of report. If no report is requested, the method returns a list of available reports.

TBD - More details on error codes

TBD - Give request/response example

5.3 New Versioning Methods

WebDAV versioning introduces three new methods to support the checkout/checkin versioning model.

5.3.1 CHECKOUT

A CHECKOUT request can only be applied to a versioned resource. When a CHECKOUT request is applied to a versioned resource whose target is a revision, a new working resource is created that is a copy of the revision, and the DAV:predecessor of the working resource is set to be that revision. If the DAV:predecessor has a

DAV:single-checkout property and is already checked out into a workspace, the CHECKOUT request fails. The DAV:workspace of the working resource is set to be the workspace specified in the Target-Selector header. If the Target-Selector is not a workspace or if there is no Target-Selector header, the DAV:workspace for that working resource is allocated by the server. The body of the CHECKOUT request can be used to initialize the DAV:checkin-policy of the working resource.

When a CHECKOUT request is applied to a versioned resource whose target is a working resource, the CHECKOUT request MUST fail.

On CHECKOUT, the DAV:activity of the new working resource is set to be the DAV:activity of the current workspace. If DAV:activity is not set, a server with activity support automatically assigns an activity to the new working resource. The CHECKOUT request fails if neither DAV:activity nor DAV:label is set.

TBD - Failures must include "policy not supported"

TBD - More details on error codes

TBD - Give request/response example

5.3.2 CHECKIN

When a CHECKIN request is applied to a versioned resource whose target is a working resource, a copy of the working resource is made a new revision of that versioned resource and the working resource is deleted. The new revision is added to the DAV:successors collection of the DAV:predecessor of the new revision, and the workspace of the working resource is deleted from the DAV:working-resources collection of the DAV:predecessor.

The body of a CHECKIN request can be used to override the current DAV:checkin-policy values of the working resource. The effect of DAV:checkin-policy on a CHECKIN request is described in the definition of the DAV:checkin-policy property.

When the CHECKIN method is applied to a resource that is versionable, but not currently versioned, the resource is put under version control. If a versionable collection is put under version control, all members of that collection are put under version control as well.

TBD - More details on error codes

TBD - Explain effect on existing resource types

TBD - Give request/response example

5.3.3 UNCHECKOUT

When an UNCHECKOUT request is applied to a versioned resource whose target is a working resource, the DAV:workspace of that working resource is deleted from the DAV:working-resources collection of the DAV:revision of the working resource, and the working resource

is deleted. This effectively cancels the CHECKOUT request that produced the working resource.

TBD - More details on error codes

TBD - Explain effect on existing resource types

TBD - Give request/response example

5.4 New Versioning Headers

5.4.1 Target-Selector

The Target-Selector header contains a workspace URL or a revision name. The Target-Selector header is used to specify which working resource or revision of a versioned resource should be its target. If a Target-Selector header is omitted in a request on a versioned resource, the default workspace is implicitly used as the Target-Selector.

6 THE DAV VERSIONING XML ELEMENTS

6.1 Revision Selection Rule Elements

A revision selection rule document is the value of the DAV:revision-selection-rule property of a workspace. Semantically, a revision selection rule (or RSR) element can be applied to an arbitrary versioned resource, and will either select a particular revision of that versioned resource or select no revision of that versioned resource. If it selects a particular revision, it may also detect a conflict (e.g. when there were multiple candidates for selection). A document describing the conflicts can be obtained through a conflict REPORT request.

6.1.1 DAV:rsr-configuration

A DAV:rsr-configuration element contains the URL of a configuration. If the configuration contains a revision of the versioned resource, that revision is selected by the DAV:rsr-configuration element; otherwise, no revision is selected. A DAV:rsr-configuration element never generates a conflict.

6.1.2 DAV:rsr-activity-latest

A DAV:rsr-activity-latest element contains the URL of an activity. If the DAV:revisions collection of the activity contains one or more revisions of the versioned resource, then the latest revision in that activity is selected. The semantics of activities ensures that there always is a unique latest revision for an activity. If the activity contains no revisions of a versioned resource, the

DAV:rsr-activity-latest element selects no revisions of that versioned resource.

If the DAV:needed-activities collection of an activity is non-empty, then the DAV:rsr-activity element acts like a DAV:rsr-merge element that contains that activity and each of the DAV:needed-activities. A DAV:rsr-activity-latest element can generate conflicts only if the DAV:needed-activities collection is non-empty.

6.1.3 DAV:rsr-label

A DAV:rsr-label element contains a label. If a revision of the versioned resource has that label, that revision is selected by the DAV:rsr-label element; otherwise, no revision is selected. A DAV:rsr-label element never generates a conflict.

6.1.4 DAV:rsr-revision-id

A DAV:rsr-revision-id element contains a revision id. If a revision of the versioned resource has that revision id, that revision is selected by the DAV:rsr-revision-id element; otherwise, no revision is selected. A DAV:rsr-revision-id element never generates a conflict.

6.1.5 DAV:rsr-latest

A DAV:rsr-latest element selects the revision of that versioned resource with the latest DAV:creationdate. A DAV:rsr-latest element never generates a conflict.

6.1.6 DAV:rsr-or

A DAV:rsr-or element contains a sequence of RSR elements. The behavior of a DAV:rsr-or element is identical to the behavior of the first element in this sequence that selects a revision of the versioned resource. If no element selects a revision, then the DAV:rsr-or element selects no revision of that versioned resource.

6.1.7 DAV:rsr-merge

A DAV:rsr-merge element contains a sequence of RSR elements. If one of the elements in this sequence selects a revision that is the descendent of all of the other revisions selected by elements in this sequence, then that revision is selected by the DAV:rsr-merge element. If no selected revision is a descendent of all the other selected revisions, then the result of the DAV:rsr-merge is a "conflict". A conflicts REPORT request can be used to identify the conflicts.

6.2 Report Elements

6.2.1 Conflicts Report

A conflicts report describes the conflicts in a specified workspace for a specified resource or for all the members of a specified versioned collection.

6.2.1.1 DAV:conflicts-report-request

A DAV:conflicts-report-request contains the URL of a workspace and the URL of a versioned resource.

6.2.1.2 DAV:conflicts-report-response

A DAV:conflicts-report-response contains a DAV:conflict element for each versioned resource for which the revision selection rule specified in the DAV:conflicts-report-request produced a conflict. The DAV:conflict element identifies the versioned resource which generated a conflict, as well as information about how to resolve the conflict (such as the revisions that would need to be merged).

6.2.1.3 DAV:conflict

The DAV:conflict element contains the URL of a versioned resource for which the revision selection rule generated a conflict, a DAV:common-ancestor for the conflict, and two or more DAV:contributor elements for the conflict.

6.2.1.4 DAV:common-ancestor

The DAV:common-ancestor element identifies a revision that is a common ancestor of all the DAV:contributor elements for a particular conflict.

6.2.1.5 DAV:contributor

The DAV:contributor element identifies a revision that is selected by an element of the revision selection rule but that is not an ancestor of any of the other revisions selected by the revision selection rule.

6.2.2 Compare Report

6.2.2.1 DAV:compare-report-request

A DAV:compare-report-request contains the URL's of two resources of the same type.

6.2.2.2 DAV:compare-report-response

A DAV:compare-report-response identifies the differences between two resources of the same type. These differences are indicated by appropriate DAV:added, DAV:deleted, and DAV:changed elements.

In particular, if a DAV:compare-report-request is applied to two configuration revisions. The DAV:compare-report-response contains the revisions, needed-configurations, and activities that are selected by one configuration revision but not the other.

6.2.2.3 DAV:added

A DAV:added element identifies something that appears in the second resource but not in the first.

6.2.2.4 DAV:deleted

A DAV:deleted element identifies something that appears in the first resource but not in the second.

6.2.2.5 DAV:changed

A DAV:changed element identifies something that is in both resources but that is in some way different in the two resources. For example, when two configurations are being compared, a DAV:changed element can identify a versioned resource, a versioned collection, or an activity. A versioned resource has changed if the configurations select different revisions of that versioned resource. A versioned collection has changed if the configurations select different revisions or different baselines of that versioned collection. An activity has changed if both configurations contain that activity but the DAV:revisions or DAV:needed-activities of that activity were different when those configurations were created.

7 INTERNATIONALIZATION CONSIDERATIONS

To be supplied.

NOTE: If a revision label contains a URL reserved character, that character is escaped in the DAV:revision-labels name.

8 SECURITY CONSIDERATIONS

To be supplied.

9 SCALABILITY

To be supplied.

10 AUTHENTICATION

Authentication mechanisms defined in WebDAV will also apply to DAV Versioning.

11 IANA CONSIDERATIONS

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

12 INTELLECTUAL PROPERTY

The following notice is copied from RFC 2026 [RFC2026], section 10.4, and describes the position of the IETF concerning intellectual property claims made against this document.

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

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

13 ACKNOWLEDGEMENTS

This protocol is the collaborative product of the Delta-V design team: Jim Amsden (IBM, DeltaV Chair), Geoffrey Clemm (Rational), Bruce Cragun (Novell), David Durand (INSO), Chris Kaler (Microsoft), Jeff McAffer (OTI), Bradley Sergeant (Merant), and Jim Whitehead (UCI). We would like to acknowledge the foundation laid for us by the authors of the WebDAV and HTTP protocols upon which this protocol is layered, and the invaluable feedback from the WebDAV and DeltaV working groups.

14 INDEX

To be supplied.

15 REFERENCES

[RFC2068] 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.

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

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

[AdvCol] http://www.ietf.org/internet-drafts/draft-ietf-webdav-collection-protocol-03.txt

[VerGoal] http://www.ietf.org/internet-drafts/draft-ietf-webdav-version-goals-02.txt

16 AUTHORSÆ ADDRESSES

Geoffrey Clemm
Rational Software
20 Maguire Road
Lexington MA 02421-3104
Email: geoffrey.clemm@rational.com

Christopher Kaler
Microsoft
One Microsoft Way
Redmond WA 9085-6933
Email: ckaler@microsoft.com

17 OPEN ISSUES

The following list identifies key open issues against this document:

@ TODO: Add the appropriate DAV:resourcetype properties to workspace, history resource, activity, and configuration.

@ TODO: Flush out details on methods: e.g., request/response examples needed.

@ TODO: DTDs and semantics of properties

@ TODO: Lots of details

@ ISSUE: How are policies discovered?

@ ISSUE: Does Versioning have to be a header, or can it be the body of the OPTIONS response?

@ ISSUE: Couldn't MKRESOURCE be handled just by allowing PROPPATCH to be applied to a null resource?