WebDAV and the Writeable Web

Infrastructure for Mobile, Collaborative XML

Jim Whitehead

Jim@4K-Associates.com

9 August 1999

"The next phase of the Web"

- With the dramatic flood of rich material of all kinds onto the Web in the 1990s, the *first part of the dream* is largely realized, although still *very few people in practice have access to intuitive hypertext creation tools*. The second part has yet to happen, but there are signs and plans which make us confident. The great need for information about information, to help us categorize, sort, pay for, own information is driving the design of languages for the web designed for processing by machines, rather than people.
  The web of human-readable documents is being merged with a web of machine-understandable data. The potential of the mixture of humans and machines working together and communicating through the web could be immense.

  - *W3C Director Tim Berners-Lee, May 1998*

Every Bit in XML

- Across Time
  - *Save Self-Description with Data State*
  - *“Future Proofing” of Documents and Data*
- Across Space
  - *For Exchange as well as Storage of Data*
- Across Organizations
  - *Building Community-Specific Ontologies*
  - *Key to Knowledge Representation*

Web Automation & Collaboration

- Automation — XML
  - *Shifting from "screen-scrapers" of presentational data to programmatic access to logical data*
- Collaboration — DAV
  - *Returning the Web to a read and write medium...*
  - *… Adding coordination for group authoring*
  - *… Making XML mobile for knowledge management & workflow*

Interdependencies

- The enabling technologies for automation and collaboration are interdependent
  - *XML knowledge is represented as Web resources*
  - *XML is the key to automation, and is the format of WebDAV properties*
  - *WebDAV locking ensures safe, remote access to XML resources*
  - *WebDAV properties provide metadata storage with Web resources, linking XML to the resources they describe*

Network Effects

- XML and WebDAV each generate network effects for each other
  - *WebDAV increases the value of XML by…*
    - … providing remote access to XML resources
    - … linking XML properties to resources
    - … giving mobility to XML resources
  - *XML increases the value of WebDAV by…*
    - … providing an extensible marshalling syntax
    - … its i18n capabilities

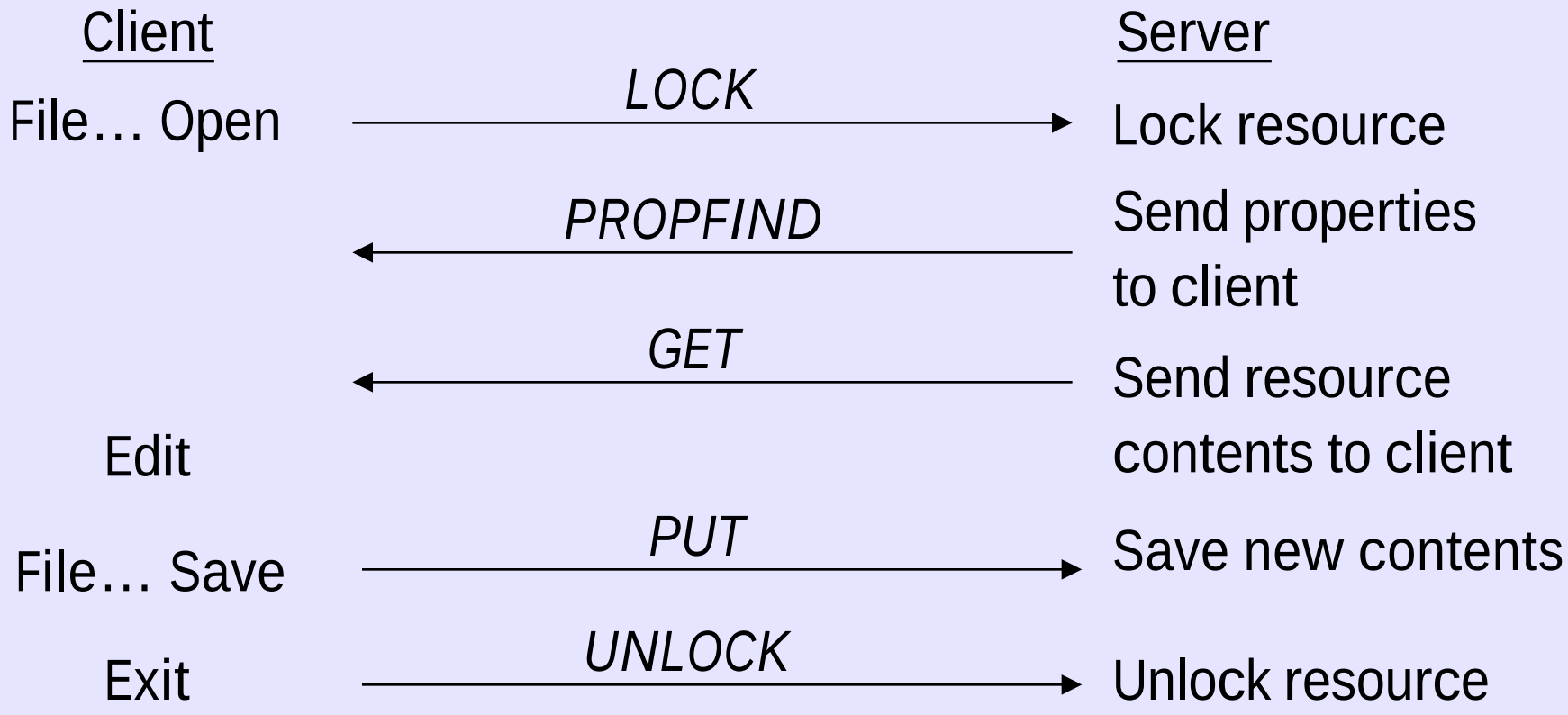
WebDAV: Extending HTTP

- WebDAV is a major extension to HTTP
  - *WebDAV adds XML properties and collections to the HTTP data model*
- WebDAV provides facilities for
  - *Overwrite prevention:* *lock, unlock*
  - *Properties:* *list, add, remove*
  - *Namespace Operations:* *move, copy*
  - *Collections:* *mkcol, hierarchy operations*

Using WebDAV

- You have to produce an XML document
  - *Fire up your favorite XML processor & start editing*
- You decide to bring on another author
  - *Using the same XML processor, save to the Web*
  - *Give your collaborator the URL of the XML document*
  - *Start collaboration on the document by editing in-place on the Web*
- A seamless transition from individual to collaborative work

Application Use of WebDAV



Visions for WebDAV

- Participants in WebDAV have many views of it:
  - *A protocol for collaborative authoring (of XML)*
  - *A protocol for renderng XML mobile by remotely manipulating XML*
  - *A large-grain Web-based network file system, with nice high-latency behavior.*
  - *A data integration technology for accessing a wide range of repositories*
    - document mgmt. systems, configuration mgt. systems, filesystems, &c
  - *A protocol for remote software engineering*
- All views are correct!

Commercial WebDAV Products 1/2

- Several companies support WebDAV in their products:
  - *Microsoft*
    - Office 2000 (Web Folders)
    - Internet Explorer 5 (Web Folders)
    - Internet Information Services 5 (IIS)
  - *Glyphica*
    - PortalWare

Commercial WebDAV Products 2/2

- DataChannel — *RIO*
- Xerox — *DocuShare 2.0*
- Cyberteams — *WebSite Director*
  - *integrated WebDAV server and workflow system*
- Intraspect — *Intraspect Knowledge Server*
- IBM — *DAV4J*
  - *server and client API (AlphaWorks – not a product)*

Open Source Projects

- Strong Open Source WebDAV support:
  - *Apache*
    - Greg Stein's mod_dav Apache module
  - *Zope – Digital Creations*
  - *Sitecopy – Joe Orton, site synchronization tool*
  - *PyDAV – Jim Davis, Python-based WebDAV server*
  - *WebDAV Explorer, DAV Posties – UC Irvine*
  - *WebRFM – Yoram Last, CGI-Perl WebDAV server*
  - *Complete list:*
    - http://www.webdav.org/projects/

Facets of WEBDAV

- There are many ways to view the DAV work:
  - *Collaboration infrastructure*
  - *Metadata recording infrastructure*
  - *Namespace management infrastructure*
  - *Ordered collection infrastructure*
  - *Versioning infrastructure – Delta-V*
  - *Access control infrastructure*
  - *Searching infrastructure – DASL*

Collaboration Infrastructure

- Whole resource locking supports:
  - *remote collaborative authoring of HTML pages and associated images*
  - *remote collaborative authoring of any media type (word processing, presentations, etc.)*
- Infrastructure for development of asynchronous, widely distributed, hypertext-aware, collaborative editing tools.

Metadata Recording Infrastructure

- Metadata support
  - Properties. *(name, value) pairs can be created, modified, deleted, and read on Web resources.*
  - *Consistency of properties can be maintained by the server or the client*
  - *Property values are well-formed Extensible Markup Language (XML)*
- *Infrastructure* for how to record information *about* Web data

Namespace Management Infrastructure

- Remote name space management:
  - *Copy and Move individual resources, and hierarchies of resource*
  - *Create and modify (ordered) collections of resources*
  - *Add/remove members by-reference*
- Infrastructure for remotely organizing and viewing collections of Web resources

Versioning Infrastructure

- Versioning and Configuration Management

  Proposed Delta-V Working Group

  - *check-out, check-in*
  - *version graph history / browse old versions*
  - *comments on check-out/check-in*
  - *automatic versioning for unaware clients*
  - *basic, high-value configuration management operations*
  - *workspaces*
  - *activities*
- Infrastructure for remotely versioning Web resources

Access Control Infrastructure

- Access Control:
  - *The ability to remotely control who can read and write a resource*
  - *Key challenge:*
    - Expose the access control capabilities of the repository…
    - …while ensuring the client-side user interface can be simple (I.e., avoid lots of feature discovery)
  - *Will be the focus of a new working group.*
- Infrastructure for remotely creating collaboration groups

Searching Infrastructure

- Searching a WebDAV repository - DASL:
  - *Search for resources with a given property, or a given property value*
  - *Search for a substring inside a resource body*
  - *Search scope can be one resource, a collection of resources, a hierarchy of resources, or a whole server*
- Infrastructure for remote searching

Document Roadmap

WebDAV Working Group:

Distributed Authoring
Locking, Properties, Copy/Move
RFC 2518 complete

Access Control
Protocol for remote access control
Finish: Mid 2000

Ordered Collections
Requirements and protocol for
ordered collections, external members
Finish: August/Sept. 1999

DASL Working Group:

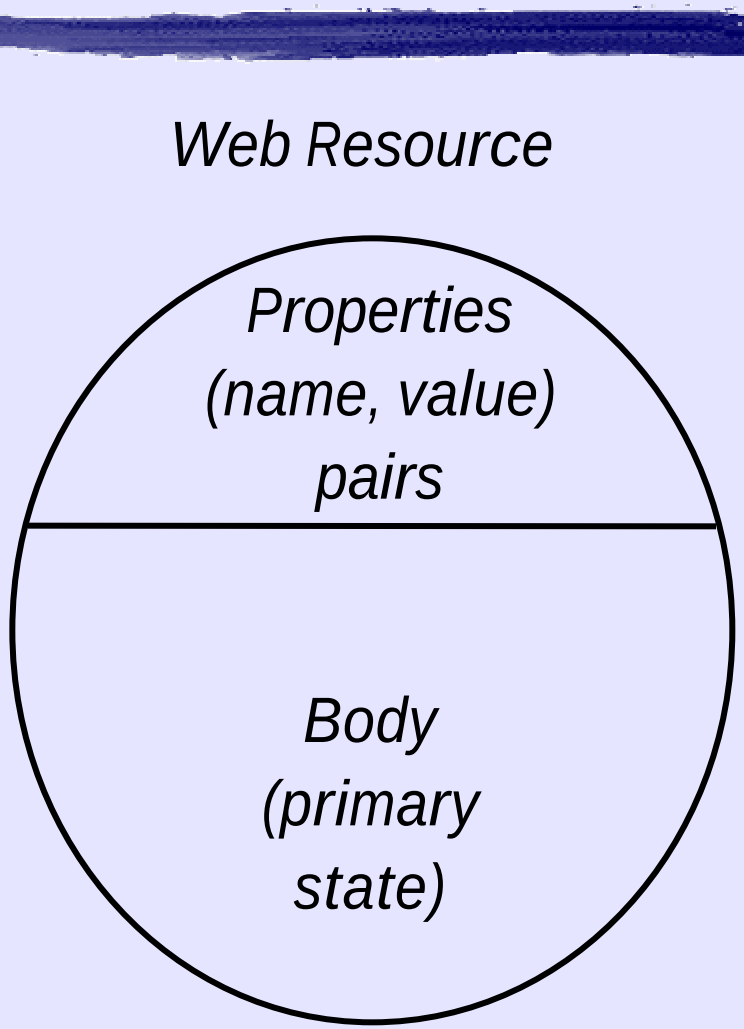
Searching
Requirements and protocol for
searching a WebDAV repository
Finish: Nov/Dec 1999

Delta-V Working Group:

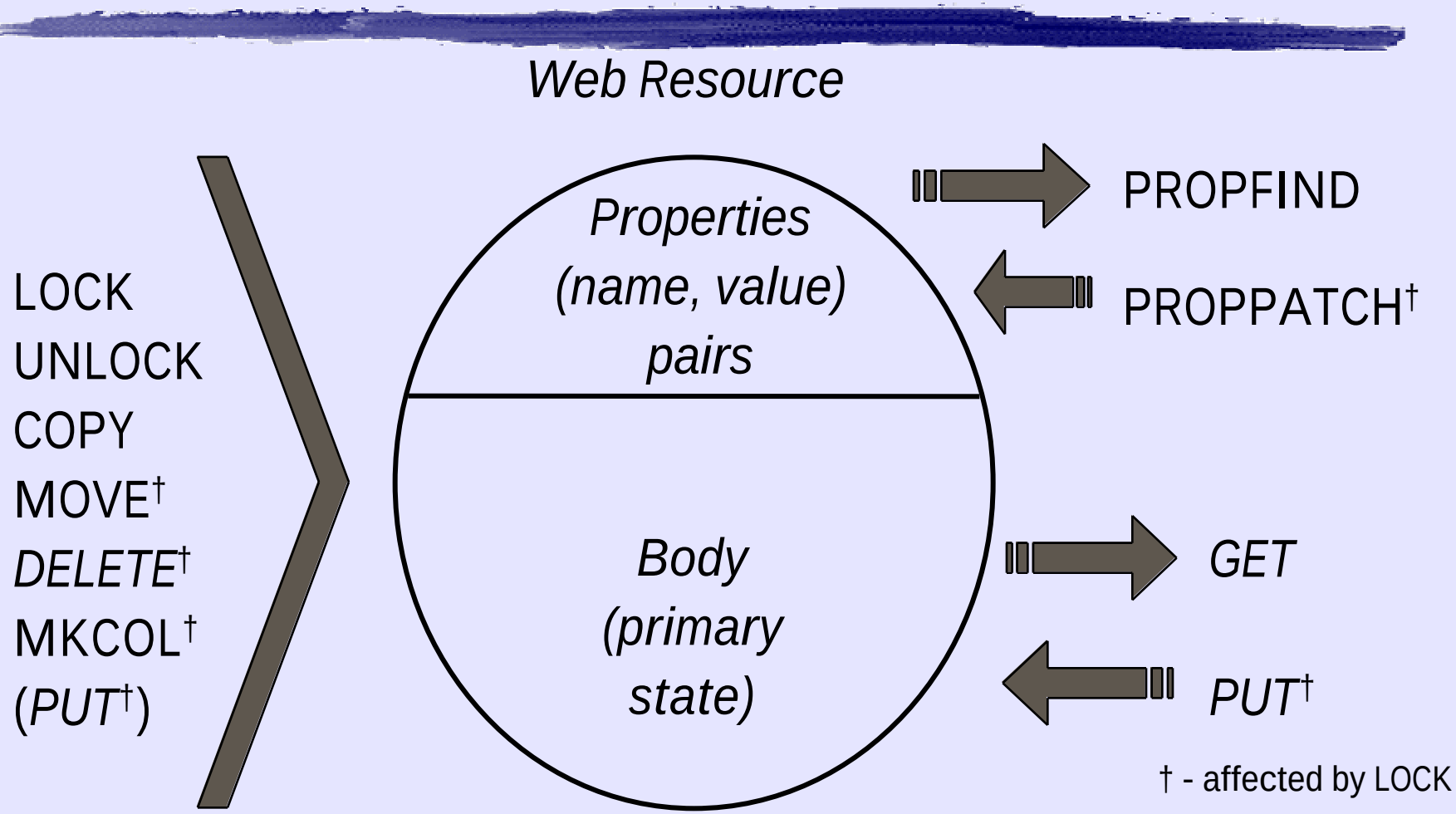
Versioning
Checkin/Checkout, Variants
Finish: Mid 2000

DAV/HTTP Object Model

WebDAV Object Model



Scope of WebDAV Methods



Properties

Properties: Naming

- Properties are (name, value) pairs
- Property names are URIs
  - *can be a URL (no registration needed)*
  - *can be a URI (register new URI scheme)*
- Benefits:
  - *Due to ownership of a domain name, URLs provide globally unique names without registration*
  - *URLs allow rapid development and deployment of new schemas*
  - *Stable, long-lifetime schemas can be named with a URI scheme, which is registered with IANA*

Properties: Name/Instance Distinction

- A property name URI names its syntax and semantics
  - *Only one instance of a property may be created on a resource (but may be multi-valued)*
  - *“live” properties: server maintains consistency by enforcing syntax and semantics*
  - *non-live properties: client enforces syntax and semantics (property may be inconsistent)*
- Benefits:
  - *server can provide properties with values it generates*
  - *client can define new properties unknown to the server*
  - *major benefit: flexibility*

Value is well-formed XML

- The value of a property is a well-formed XML (eXtensible Markup Language) fragment
  - *WebDAV also requires use of XML Namespaces*
- Benefits:
  - *extensibility: namespaces allow mix of elements in properties*
  - *i18n support: XML supports ISO 10646 encoding of characters*
  - *property contents are structured values*
  - *supports “plug-and-play” of W3C RDF work*

Properties: PROPPATCH

- PROPPATCH method is used to create and remove properties from a resource
  - *Property creation and removal directives are specified with XML "create" and "remove" elements*
  - *Directives are executed sequentially and atomically*
- Benefits:
  - *simple method handles modification to the state of a property*
  - *can modify several properties at once, with one network round trip*
  - *atomicity ensures properties will not be left in an inconsistent state*

Properties: PROPFIND

- PROPFIND retrieves properties from a resource.
  - *Retrieve all property names and values*
  - *Retrieve only specified names and values*
  - *Retrieve only a list of property names*
- Benefits:
  - *can retrieve property information with a single network round trip*

PROPFIND, implicit allprop

- PROPFIND /demo.txt HTTP/1.1
- Host: dav.ics.uci.edu
- Depth: 0
- Content-Length: 0

HTTP/DAV request

- HTTP/1.1 207 Multi-Status
- Server: DAV-demo-server/1.0
- Date: Tue, 09 Feb 1999 00:25:47 GMT
- Content-Type: text/xml; charset="utf-8"
- Content-Length: 891

HTTP/DAV response

- <?xml version="1.0" encoding="utf-8" ?>
- <a:multistatus xmlns:a="DAV:">
- <a:response>
- <a:href>http://dav.ics.uci.edu/demo.txt</a:href>
- <a:propstat>

response body

PROPFIND, implicit allprop (2)

```
<a:status>HTTP/1.1 200 OK</a:status>
    <a:prop>
        <a:getcontentlength>59</a:getcontentlength>
        <a:creationdate>
            1999-02-09T00:16:40.574Z
        </a:creationdate>
        <a:displayname>demo.txt</a:displayname>
        <a:getetag>"10598798c153be1:a82"</a:getetag>
        <a:getlastmodified>
            Tue, 09 Feb 1999 00:17:36 GMT
        </a:getlastmodified>
        <a:resourcetype/>
        <a:lockdiscovery/>
```

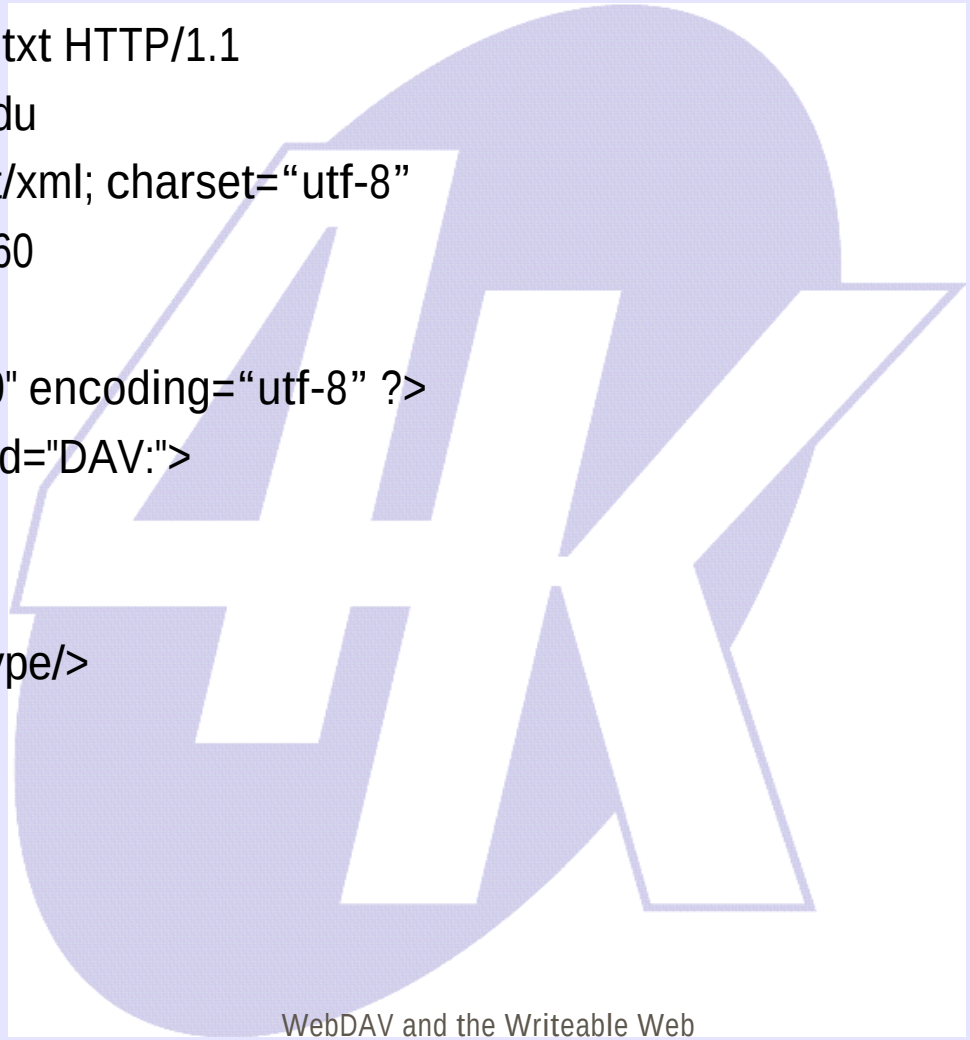
PROPFIND, implicit allprop (3)

```
                    <a:supportedlock>
                        <a:lockentry>
                            <a:locktype><a:write/></a:locktype>
                            <a:lockscope><a:shared/></a:lockscope>
                        </a:lockentry>
                        <a:lockentry>
                            <a:locktype><a:write/></a:locktype>
                            <a:lockscope><a:exclusive/></a:lockscope>
                        </a:lockentry>
                    </a:supportedlock>
                    <a:getcontenttype>text/plain</a:getcontenttype>
                </a:prop>
            </a:propstat>
    </a:response>
    </a:multistatus>
```

PROPFIND, named properties

```
PROPFIND /demo.txt HTTP/1.1
Host: dav.ics.uci.edu
Content-Type: text/xml; charset=“utf-8”
Content-Length: 160

<?xml version="1.0" encoding=“utf-8” ?>
<d:propfind xmlns:d="DAV:">
  <d:prop>
    <d:getetag/>
    <d:getcontenttype/>
  </d:prop>
</d:propfind>
```



PROPFIND, named properties (2)

- HTTP/1.1 207 Multi-Status
- Server: DAV-demo-server/1.0
- Date: Tue, 09 Feb 1999 00:52:55 GMT
- Content-Type: text/xml; charset="utf-8"
- Content-Length: 321

- <?xml version="1.0" encoding="utf-8" ?>
- <a:multistatus xmlns:a="DAV:">
- <a:response>
- <a:href>http://dav.ics.uci.edu/demo.txt</a:href>
- <a:propstat>
- <a:status>HTTP/1.1 200 OK</a:status>

PROPFIND, named properties (3)

```
            <a:prop>
                <a:getetag>"10598798c153be1:a82"</a:getetag>
                <a:getcontenttype>text/plain</a:getcontenttype>
            </a:prop>
        </a:propstat>
    </a:response>
    </a:multistatus>
```

PROPPATCH

- PROPPATCH /webdav.html HTTP/1.1
- Host: sandbox.xerox.com
- Content-Type: text/xml; charset="utf-8"
- Content-Length: 283

- <?xml version="1.0" encoding="utf-8" ?>
- <d:propertyupdate xmlns:d="DAV:">
- <d:set>
- <d:prop xmlns:j="http://www.ics.uci.edu/~ejw/">
- <j:personal>
- <j:item>My property</j:item>
- </j:personal>
- </d:prop>
- </d:set>
- </d:propertyupdate>

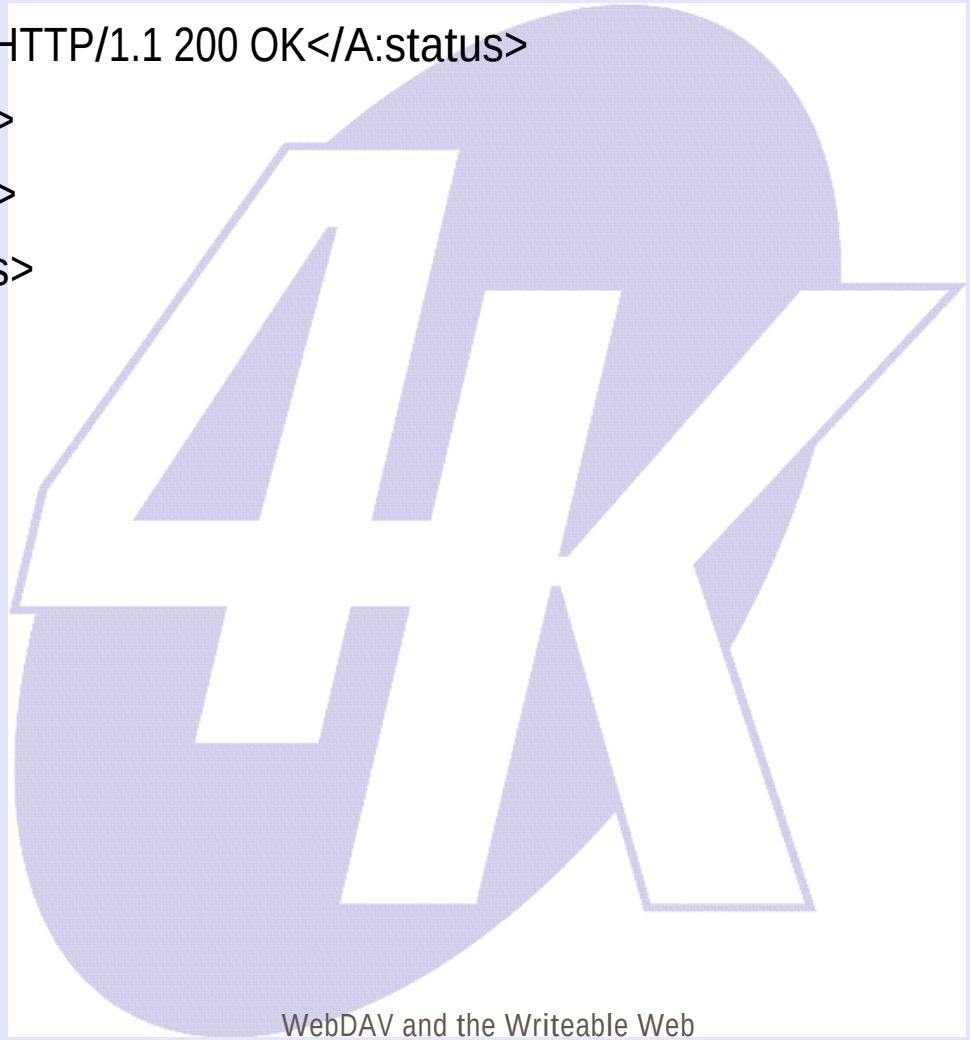
PROPPATCH (2)

- HTTP/1.1 207 Multi-Status
- Date: Tue, 09 Feb 1999 01:36:43 GMT
- Server: PyDAV 1.1 filestore 1.1
- Content-Type: text/xml; charset="utf-8"
- Content-Length: 317

- <?xml version="1.0" encoding="utf-8" ?>
- <A:multistatus xmlns:A="DAV:">
- <A:response>
- <A:href>/webdav.html</A:href>
- <A:propstat>
- <A:prop>
- <B:personal xmlns:B="http://www.ics.uci.edu/~ejw/"/>
- </A:prop>

PROPPATCH (3)

- <A:status>HTTP/1.1 200 OK</A:status>
- </A:propstat>
- </A:response>
- </A:multistatus>



Overwrite Prevention: Locking

Write Lock

- A write lock prevents a principal without the lock from successfully modifying the state of the resource
  - *Specifically, it prevents execution of PUT, POST, DELETE, MKCOL, PROPPATCH, MOVE, LOCK, UNLOCK*
  - *GET, PROPFIND are unaffected by write lock*

Write Lock Scope

- The scope of a lock is an entire resource
  - *It is not possible to specify a sub-resource lock. Why?*
    - The ability to create a sub-resource lock requires knowledge about the content type being locked.
    - WebDAV wanted to support all content types equally, not give preferential support to a few.
    - Due to the large number of content types, support for sub-resource locks would have added a lot of extra semantics to the protocol. Some content types wouldn't have been supported.

Write Lock Scope (2)

- Further arguments against sub-resource locking:
  - *Content types change rapidly (two revisions of HTML and XML occurred during development of WebDAV protocol). Supporting one revision of a content type would make protocol brittle, quickly obsolete.*
  - *New content types would not be supported, yet new content types emerge constantly.*

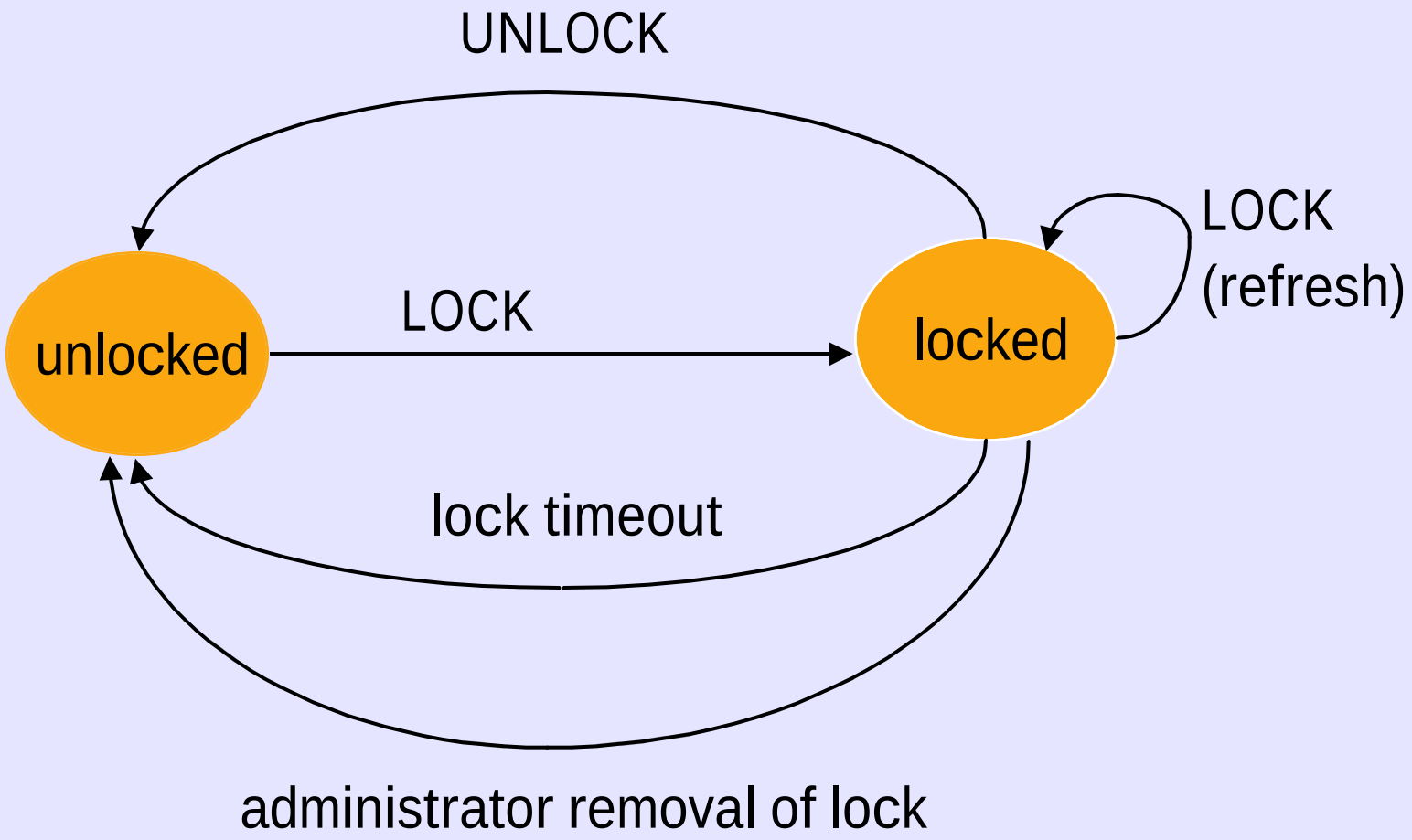
Write Lock Scope (3)

- Implications of whole-resource locking:
  - *Pro:*
    - Supports existing applications which operate on entire files, providing easy migration path to add WebDAV support
    - Handles all content types
  - *Con:*
    - Reduced availability of resources during collaboration (but, shared locks can help…)

Write Lock (cont'd)

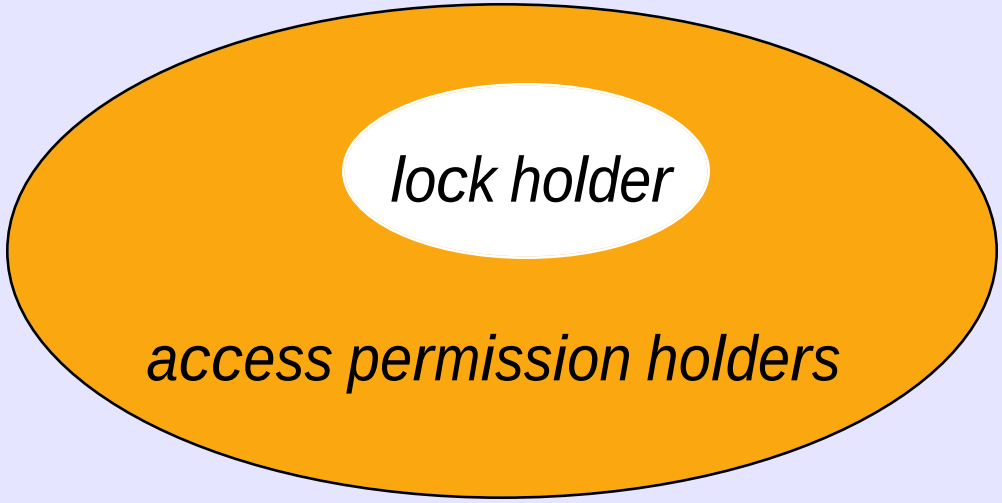
- Live properties may still change even though a lock is active
- Dead properties may only be changed by the lock owner.
- A null resource may be locked to reserve its name. This makes the resource non-null, since it now has lock related properties defined on it.

Lifecycle of a Lock



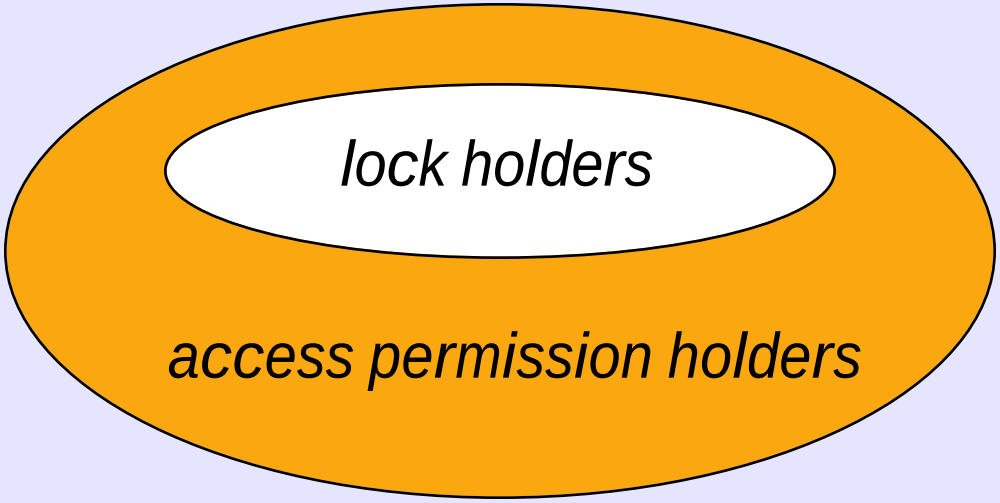
Exclusive Lock

- Zero or one possible lock holders



Shared Lock

Zero, one, or many possible lock holders



Why Exclusive and Shared?

- Exclusive locks are too rigid
  - *people often forget to release locks*
  - *requires an administrator to release the lock*
- Shared locks allow people to use out-of-band communication to negotiate access to the resource
  - *if someone forgets to release a lock, it doesn't hold up the entire group*
- Collaborators work *opportunistically*

Lock Compatibility

Current lock state \ Lock request	Shared Lock	Exclusive Lock
None	true	true
Shared Lock	true	false
Exclusive Lock	false	false*

Legend: True = lock MAY be granted
False = lock MUST NOT be granted

* = owner of lock MAY have lock regranted

LOCK Required Support

- A WEBDAV server is not required to support locking
- Why? Systems differ markedly in the type of locking support they provide, and may not be able to support locks at all (i.e., some replicated stores)
- … However, most WebDAV servers do support locking

LOCK Method

- LOCK creates the lock specified by the <lockinfo> XML element (in the request body) on the Request-URI.
- A user-agent should submit owner information with a lock request
- LOCK returns a lock token which identifies the lock to the server
- The client may request a timeout value

Lock Owner Information

- The owner XML element (inside <lockinfo>) provides a means to associate lock holder contact information with a lock.
  - *If you want the lock holder to release the lock, perhaps you can contact them and ask them to relinquish it*
  - *Authentication information often does not contain contact information (e.g., a key)*

LOCK, single resource

```
LOCK /webdav.html HTTP/1.1
Host: sandbox.xerox.com
Timeout: Second-500, Infinite
Content-Type: text/xml
Content-Length: 151

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

LOCK, single resource (2)

```
HTTP/1.1 200 OK
Date: Tue, 09 Feb 1999 02:25:21 GMT
Server: PyDAV 1.1 filestore 1.1
Content-Type: text/xml
Content-Length: 435

<?xml version="1.0"?>
<A:prop xmlns:A="DAV:">
  <A:lockdiscovery>
    <A:activelock>
      <A:lockscope>
        <A:exclusive/>
      </A:lockscope>
```

LOCK, single resource (3)

```
    <A:locktype>
     <A:write/>
    </A:locktype>
    <A:depth>infinity</A:depth>
    <A:timeout>Second-500</A:timeout>
    <A:locktoken>
     <A:href>opaquelocktoken:918527121.406</A:href>
    </A:locktoken>
   </A:activelock>
  </A:lockdiscovery>
</A:prop>
```

Hierarchy Locks

- Using the Depth header set to Infinity, can lock a collection hierarchy
  - *A single lock token is returned, identifying the lock on all resources.*
  - *An UNLOCK on this token removes the lock from all associated resources.*
- All or nothing semantics
  - *All resources in hierarchy are locked, or none are*

Hierarchy Locks (cont'd)

- Hierarchy locks act to ensure:
  - *All resources in the hierarchy are members of the lock*
  - *Resources removed from the hierarchy are removed from the lock*
- But…
  - *If a locked hierarchy is copied/moved, the destination hierarchy is not locked.*

Hierarchy Lock Cases

- Cases:
  - *COPY/MOVE IN: a resource copied/moved into a locked hierarchy is added to the lock for that hierarchy*
  - *COPY/MOVE WITHIN: a resource copied/moved within a locked hierarchy is still a member of that hierarchy*
  - *COPY OUT: when a resource is copied from a locked hierarchy, the source resource of the copy is still a member of the lock, the destination resource is not.*

Hierarchy Lock Cases (cont'd)

- *MOVE OUT: when a resource is moved from a locked hierarchy, it is removed from the source lock*
- *DELETE: removes the resource from the hierarchy lock*

- Combinations:
  - *Moving a resource from one locked hierarchy to another causes the resource to be removed from the source lock, and added to the destination lock.*
  - *Copying a resource from a locked hierarchy to another causes the source resource to remain in the source hierarchy lock, and the destination resource is added to the destination hierarchy lock.*

UNLOCK

- UNLOCK removes the lock identified by a lock token from the Request-URI, and all other resources included in the lock
  - *If a lock affects an entire collection, UNLOCK removes the lock from all resources in the collection.*