Datatypes for WebDAV properties
draft-reschke-webdav-property-datatypes-04

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.


This Internet-Draft will expire in September 2003.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

This specification extends the Web Distributed Authoring Protocol (WebDAV) to support both datatyping and some amount of meta information on property values. Protocol elements are defined to let clients and servers specify the datatype and meta information of a property, and to instruct the WebDAV method PROPFIND to return datatype and meta information.

Distribution of this document is unlimited. Please send comments to the Distributed Authoring and Versioning (WebDAV) working group at
w3c-dist-auth@w3.org, which may be joined by sending a message with subject "subscribe" to w3c-dist-auth-request@w3.org.

Discussions of the WEBDAV working group are archived at URL: http://lists.w3.org/Archives/Public/w3c-dist-auth/.
# Table of Contents

Abstract ................................................. 1  
Table of Contents ........................................ 3  
1 Notational Conventions ................................. 4  
2 Introduction ........................................... 5  
3 Overview ............................................... 6  
3.1 Data types ........................................... 6  
3.2 Property flags ....................................... 6  
3.2.1 pf:hidden .......................................... 6  
3.2.2 pf:protected ....................................... 7  
3.3 Property display names ............................... 7  
3.3.1 pf:property-displayname-set property (computed) .... 7  
4 Changes for PROPPATCH method ......................... 9  
4.1 Marshalling type information ......................... 9  
4.1.1 Example for successful PROPPATCH .................. 9  
4.1.2 Example for failed PROPPATCH ..................... 10  
4.1.3 Example for successful PROPPATCH where type information was not preserved ...................... 11  
4.2 Marshalling property flag information ............... 12  
4.2.1 Example for successful PROPPATCH ................. 13  
4.2.2 Example for failed PROPPATCH .................... 14  
5 Changes for PROPFIND method ........................... 16  
5.1 Marshalling of datatype information .................. 16  
5.1.1 Example for PROPFIND/prop ....................... 16  
5.2 Marshalling of property flags ........................ 17  
5.2.1 Example for PROPFIND/prop ....................... 17  
6 Compatibility Considerations ........................... 20  
6.1 Datatype marshalling .................................. 20  
6.2 Property flags ....................................... 20  
6.3 New live properties .................................. 20  
7 Internationalization Considerations .................... 21  
8 IANA Considerations ..................................... 22  
9 Copyright .............................................. 23  
10 Intellectual Property ................................... 24  
11 Acknowledgements ..................................... 25  
Normative References ..................................... 26  
Informative References ................................... 26  
Author’s Address ......................................... 26  
A Example: marshalling of array-typed properties ...... 27  
A.1 Setting of array-typed property ..................... 27  
A.2 Getting an array-typed property ..................... 28  
B Change Log ............................................ 30  
B.1 Since ‘draft-reschke-webdav-property-datatypes-00’ .... 30  
B.2 Since ‘draft-reschke-webdav-property-datatypes-01’ .... 30  
B.3 Since ‘draft-reschke-webdav-property-datatypes-02’ .... 30  
B.4 Since ‘draft-reschke-webdav-property-datatypes-03’ .... 30
1 Notational Conventions

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].

The term "property element" refers to the XML element that identifies a particular property, for instance

```xml
<getcontentlength xmlns="DAV:" />
```

The term "prop element" is used for the WebDAV "prop" element as defined in section 12.11 of [RFC2518].

The XML representation of schema components uses a vocabulary identified by the namespace name "http://www.w3.org/2001/XMLSchema". For brevity, the text and examples in this specification use the prefix "xs:" to stand for this namespace; in practice, any prefix can be used. "XML Schema: Structures" ([XS1]) also defines several attributes for direct use in any XML documents. These attributes are in a different namespace named "http://www.w3.org/2001/XMLSchema-instance". For brevity, the text and examples in this specification use the prefix "xsi:" to stand for this latter namespace; in practice, any prefix can be used.

This document defines extension elements and attributes that could ultimately become part of the core WebDAV protocol. Being just an individual submission, it currently defines them in the proprietary namespace

```
http://sapportals.com/xmlns/cm/webdav
```

instead of the "DAV:" namespace. It uses a prefix of "pf:" for referring to elements or attributes in this namespace. However, WebDAV server and clients are free to use any prefix, provided that there is a namespace declaration that binds the prefix to the URI of the same namespace.
2 Introduction

This specification builds on the infrastructure provided by the WebDAV Distributed Authoring Protocol, adding support for data-typed properties, some property flags and display names for properties.

Although servers must support XML content in property values, it may be desirable to persist values as scalar values when possible, and to expose the data’s type when the property value is returned to the client. The client is free to ignore this information, but it may be able to take advantage of it when modifying a property.

On the other hand, when setting new properties, it can be desirable to pass data type information along with the value. A server can take advantage of this information to optimize storage and to perform additional parsing (for instance of dates). Servers that support searching can also take advantage of known data types when doing comparisons and sorting.

Furthermore, it may be desirable to add some amount of additional meta information to properties in order to enable generic WebDAV clients to provide a meaningful user interface for editing these properties. In particular, clients can take advantage of knowing that particular properties are not generally suitable for edits through humans ("hidden"), or that they can not be changed ("protected").

Finally, generic clients that allow editing of arbitrary properties need to display a "display name" for each property. This document defines a new protected live property, "pf:property-displayname-set", that provides this information.
3 Overview

3.1 Data types

Although WebDAV property types can be anything that can be marshalled as content of an XML element, in many cases they actually are simple types like integers, booleans or dates. "XML Schema Part 2: Datatypes" [XS2] defines a set of simple types which can be used as a basis for supplying type information to attributes.

Data type information is represented using the attribute "type" from the XML Schema namespace "http://www.w3.org/2001/XMLSchema-instance". In XML Schema, data types are qualified names, and the XML Schema recommendation defines a set of built-in datatypes (section 3 of [XS2]), defined in the namespace "http://www.w3.org/2001/XMLSchema".

To avoid unnecessary verbosity, data type information should only be supplied if it adds usable information to the protocol. In particular, type information is not required for live properties defined in WebDAV [RFC2518] and for properties of type "xs:string".

A server may implement any combination of datatypes, both from the XML Schema recommendation and possibly from other namespaces.

Note that a particular property can be typed for a number of reasons:

- The property is a live property with server-defined semantics and value space.
- The property may have been set using a non-WebDAV protocol that the server understands in addition to WebDAV.
- The type may have been specified in an extended PROPPATCH method as defined in section 4.

3.2 Property flags

This specification defines semantics for two specific property flags.

3.2.1 pf:hidden

pf:hidden = "false" | "true"
This is a (boolean) display hint for generic user agents. When set to "false", it indicates that it is generally not useful to allow users to modify this property.

3.2.2 pf:protected

pf:protected = "false" | "true"

When set to "true", this flag indicates that this property is protected (as defined in [RFC3253], section 1.4.2). A user agent may display this property, but should not allow edits on it.

3.3 Property display names

For the purpose of marshalling property displayname information, this specification introduces a new computed resource property. In accordance to [RFC3253], this property SHOULD not returned upon an PROPFIND/allprop request.

3.3.1 pf:property-displayname-set property (computed)

<!ELEMENT pf:property-displayname-set (pf:property-displayname*)>
<!ELEMENT pf:property-displayname (prop, pf:displayname)>

The pf:property-displayname-set contains display information about properties defined on the resource. The set may not be complete (if the server doesn’t have display information for particular properties).

prop: ANY, see RFC2518

prop: ANY, see RFC2518
The DAV:prop element contains exactly one property element identifying the resource property to which the pf:property-displayname element applies.

<!ELEMENT pf:displayname (#PCDATA) >

The pf:displayname element contains the display name for the property. Servers MUST indicate the human language of the description using the xml:lang attribute and SHOULD consider the HTTP Accept-Language request header when selecting one of multiple available languages.
4 Changes for PROPPATCH method

4.1 Marshalling type information

If the property element has an XML attribute named "xsi:type", the server may use this information to select an optimized representation for storing the property value. For instance, by specifying a type as "xs:boolean", the client declares the property value to be of type boolean (as defined in [XS2]). The server may choose any suitable internal format for persisting this property, and in particular is allowed to fail the request if the format given does not fit the format defined for this type.

The server should indicate successful detection and parsing of the typed value by setting the xsi:type attribute on the property element in the response body (this implies that it should return a MULTISTATUS status code and a <multistatus> response body).

4.1.1 Example for successful PROPPATCH

>>Request

PROPPATCH /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:Z="http://ns.example.org/standards/z39.50">
<D:set>
  <D:prop>
    <Z:released xsi:type="xs:boolean">false</Z:released>
  </D:prop>
</D:set>
</D:propertyupdate>

>>Response

HTTP/1.1 207 Multi-Status
In this cases, the xsi:type attribute on the element "Z:released" indicates that the server indeed has understood the submitted data type information.

4.1.2 Example for failed PROPPATCH

>>Request

PROPPATCH /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset=utf-8
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:Z="http://ns.example.org/standards/z39.50">
 <D:set>
   <D:prop>
     <Z:released xsi:type="xs:boolean">t</Z:released>
   </D:prop>
 </D:set>
</D:propertyupdate>
In this case the request failed because the supplied value "t" is not a valid representation for a boolean value.

Note that similar error conditions can occur in the standard WebDAV protocol even though no data type was specified: for instance, when a client tries to set a live property for which only a certain value space is allowed.

4.1.3 Example for successful PROPPATCH where type information was not preserved

>>Request

PROPPATCH /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:Z="http://ns.example.org/standards/z39.50">
 <D:set>
</D:multistatus>
In this case the request succeeded, but the server did not know how to handle the data type "Z:custom". Therefore no data type information was returned in the response body.

4.2 Marshalling property flag information

If the property element has an XML attribute named "pf:hidden", the server should persist this as part of the property value.

On the other hand, an XML attribute named "pf:protected" SHOULD be ignored, because protected properties MUST NOT be modifiable by PROPPATCH.
4.2.1 Example for successful PROPPATCH

>>Request

PROPPATCH /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:"
 xmlns:pf="http://sapportals.com/xmlns/cm/webdav"
 xmlns:Z="http://ns.example.org/standards/z39.50">
 <D:set>
   <D:prop>
     <Z:author pf:hidden="false">Joe User</Z:author>
     <Z:int-doc-id pf:hidden="true">ADJSTCR</Z:int-doc-id>
   </D:prop>
 </D:set>
</D:propertyupdate>

>>Response

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
 xmlns:Z="http://ns.example.org/standards/z39.50">
 <D:response>
   <D:href>http://example.org/bar.html</D:href>
   <D:propstat>
     <D:prop>
       <Z:author />
     </D:prop>
     <D:prop>
       <Z:int-doc-id />
     </D:prop>
     <D:status>HTTP/1.1 200 OK</D:status>
   </D:propstat>
 </D:response>
</D:multistatus>
4.2.2 Example for failed PROPPATCH

>>Request

PROPPATCH /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate xmlns:D="DAV:"
   xmlns:pf="http://sapportals.com/xmlns/cm/webdav"
   xmlns:Z="http://ns.example.org/standards/z39.50">
   <D:set>
     <D:prop>
       <Z:author pf:hidden="flase">Joe User</Z:author>
     </D:prop>
   </D:set>
</D:propertyupdate>

>>Response

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
   xmlns:Z="http://ns.example.org/standards/z39.50">
   <D:response>
     <D:href>http://example.org/bar.html</D:href>
     <D:propstat>
       <D:prop/>
       <D:status>HTTP/1.1 422 Unprocessable Entity</D:status>
       <D:responsedescription>
         Does not parse as xs:boolean
       </D:responsedescription>
     </D:propstat>
   </D:response>
</D:multistatus>
In this case the request failed because the supplied value "flase" is not a valid value for pf:hidden.
5 Changes for PROPFIND method

5.1 Marshalling of datatype information

PROPFIND is extended to return the data type information for properties unless one of the following conditions is met:

- The data type MUST be different from "xs:string" (because this can be considered the default data type).
- The property’s data type MUST not be defined in [RFC2518] (because these types are already well-defined).

5.1.1 Example for PROPFIND/prop

>>Request

PROPFIND /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  xmlns:Z="http://ns.example.org/standards/z39.50"
  <D:prop>
    <D:getcontenttype/>
    <Z:released/>
  </D:prop>
</D:propfind>

>>Response

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
  xmlns:Z="http://ns.example.org/standards/z39.50"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
<D:response>
  <D:href>http://example.org/bar.html</D:href>
  <D:propstat>
    <D:prop>
      <D:getcontenttype>text/html</D:getcontenttype>
      <Z:released xsi:type="xs:boolean">1</Z:released>
    </D:prop>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:propstat>
</D:response>
</D:multistatus>

This example shows that the property value "true" is returned with the correct data type information, and that the server chose one of the two possible representations defined in XML Schema. It also shows that data type information is not returned for "D:getcontenttype", as this property’s data type is already defined in [RFC2518].

5.2 Marshalling of property flags

Marshalling of property flags is triggered by adding extension elements to the PROPFIND request body accordingly.

```xml
<!ELEMENT propfind ((allprop | propprop | prop),
  pf:include-hidden-flag+,
  pf:include-protected-flag+)>
<!ELEMENT pf:include-hidden-flag EMPTY>
<!ELEMENT pf:include-protected-flag EMPTY>
```

Presence of a pf:include-hidden-flag element in the request body indicated that the server SHOULD include the flag pf:hidden on all properties. Presence of a pf:include-protected-flag element in the request body indicated that the server SHOULD include the flag pf:protected on all properties.

5.2.1 Example for PROPFIND/prop
>>Request

PROPFIND /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:"
xmlns:pf="http://sapportals.com/xmlns/cm/webdav"
xmlns:Z="http://ns.example.org/standards/z39.50">
  <D:prop>
    <D:getcontenttype/>
    <D:getetag/>
    <Z:author/>
  </D:prop>
  <pf:include-hidden-flag/>
  <pf:include-protected-flag/>
</D:propfind>

>>Response

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

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
xmlns:pf="http://sapportals.com/xmlns/cm/webdav"
xmlns:Z="http://ns.example.org/standards/z39.50">
  <D:response>
    <D:href>http://example.org/bar.html</D:href>
    <D:propstat>
      <D:prop>
        <D:getcontenttype pf:hidden="false" pf:protected="true">text/html</D:getcontenttype>
        <D:getetag pf:hidden="true" pf:protected="true">"abc"</D:getetag>
        <Z:author pf:hidden="false" pf:protected="false">Joe User</Z:author>
      </D:prop>
    </D:propstat>
    <D:status>HTTP/1.1 200 OK</D:status>
  </D:response>
</D:multistatus>
</D:propstat>
</D:response>
</D:multistatus>
6 Compatibility Considerations

6.1 Datatype marshalling

This part of this specification does not introduce any new protocol elements, nor does it change the informal WebDAV DTD. It merely specifies additional server semantics for the case where clients submit additional data type information in an attribute on the property element (previously undefined), and adds an additional attribute on property elements upon PROPFIND.

Clients not aware of datatype handling should not supply the "xsi:type" attribute on property elements (after all, this attribute belongs to the XML Schema-Instance namespace which has been defined for exactly this purpose). Old clients should also ignore additional attributes on property elements returned by PROPFIND (and similar methods), although the WebDAV specification only defines this behaviour for unknown elements (and is silent about unknown attributes).

Servers not aware of datatype handling either drop the "xsi:type" attribute, or persist it along with the property value. However, they will never indicate successful parsing of the data type by returning back the type in the response to PROPPATCH.

6.2 Property flags

Property flags are only reported upon special request and thus are never seen by old clients. The PROPFIND request body has been extended according to the WebDAV XML extensibility rules defined in [RFC2518], section 14.

Clients not aware of property flags should not supply the attributes on property elements. This follows from the property flag namespace being controlled by the authors of this specification.

Servers not aware of property flags either drop them or persist them along with the property value. No harm is done, unless the client supplied erroneous values.

6.3 New live properties

The introduction of new WebDAV properties does not affect compatibility with existing implementations at all.
7 Internationalization Considerations

This proposal builds on [RFC2518], and inherits its internationalizability.
8 IANA Considerations

This proposal does not introduce any new IANA considerations, since it does not specify any new namespaces (in the general sense), but merely uses existing ones.
9 Copyright

To be supplied by the RFC Editor.
10 Intellectual Property

To be supplied by the RFC Editor.
11 Acknowledgements

This draft has benefited from thoughtful discussion by Stefan Eissing and Eric Sedlar.
Normative References


Informative References


Author’s Address

Julian F. Reschke
greenbytes GmbH
Salzmannstrasse 152
Muenster, NW 48159
Germany

Phone: +49 251 2807760
Fax: +49 251 2807761
EMail: julian.reschke@greenbytes.de
URI: http://greenbytes.de/tech/webdav/
A Example: marshalling of array-typed properties

As an example for more complex data types, this section shows marshalling of array-typed properties as implemented in the WebDAV protocol adapters of SAP Portal’s Enterprise Portal System (release 5.0).

As [XML Schema][XS2] does not define simple types for arrays, it builds on the predefined array types used in [SOAP11]. These in turn can be based on the simple types defined in XML Schema.

Note the following special properties of SOAP-encoded arrays:

1. They require an additional "arrayType" attribute to specify the array length and the base type.

2. The names of the individual children of the property element aren’t relevant as the type information is already encoded on the property element itself. It is however recommended to use identical element names for all array members.

A.1 Setting of array-typed property

>>Request

PROPPATCH /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8"?>
<propertyupdate xmlns="DAV:"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:Z="http://ns.example.org/standards/z39.50"
 xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/">
<set>
  <prop>
    <Z:refs xsi:type="soap-enc:Array"
         soap-enc:arrayType="xs:string[2]">
      <xs:string>http://www.w3.org/TR/SOAP</xs:string>
      <xs:string>http://www.w3.org/TR/xmlschema-2</xs:string>
    </Z:refs>
  </prop>
</set>
</propertyupdate>
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8"版本?>
<multistatus xmlns="DAV:"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:Z="http://ns.example.org/standards/z39.50"
xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/">
    <response>
        <href>http://example.org/bar.html</href>
        <propstat>
            <prop>
                <Z:refs xsi:type="soap-enc:Array"
                    soap-enc:arrayType="xs:string[2]"/>
            </prop>
            <status>HTTP/1.1 200 OK</status>
        </propstat>
    </response>
</multistatus>

A.2 Getting an array-typed property

>>Request

PROPFIND /bar.html HTTP/1.1
Host: example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8"版本?>
<D:propfind xmlns:D="DAV:"
xmlns:Z="http://ns.example.org/standards/z39.50">
    <D:prop>
        <Z:refs/>
    </D:prop>
</D:propfind>
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
  xmlns:Z="http://ns.example.org/standards/z39.50"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/">
  <D:response>
    <D:href>http://example.org/bar.html</D:href>
    <D:propstat>
      <D:prop>
        <Z:refs xsi:type="soap-enc:Array"
          soap-enc:arrayType="xs:string[2]">
          <xs:string>http://www.w3.org/TR/SOAP</xs:string>
          <xs:string>http://www.w3.org/TR/xmlschema-2</xs:string>
        </Z:refs>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
B Change Log

B.1 Since 'draft-reschke-webdav-property-datatypes-00'

Editorial fixes.
Changed examples to explicitly use utf-8 encoding for HTTP content type and XML encoding.
Added example for marshalling array-typed properties.

B.2 Since 'draft-reschke-webdav-property-datatypes-01'

Fix width of artwork for IETF compliance.
"Non-normative references" -> "Informative references".

B.3 Since 'draft-reschke-webdav-property-datatypes-02'

Added marshalling for property flags such as "hidden" and "protected".
Moved array marshalling example into back section.
Added rational and description for pf:property-displayname-set.
Added acknowledgements section.

B.4 Since 'draft-reschke-webdav-property-datatypes-03'

Replaced domain names in examples according to RFC2606: "www.foo.com" by "example.org",
"www.example.com" by "ns.example.org/standards/z39.50/standards/z39.50" and
"www.w3.com/standards/z39.50" by "ns.example.org/standards/z39.50".

Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for
copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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