Independent Submission
Request for Comments: 7351
Category: Informational
ISSN: 2070-1721

E. Wilde
UC Berkeley

August 2014

A Media Type for XML Patch Operations

Abstract

The XML patch document format defines an XML document structure for expressing a sequence of patch operations to be applied to an XML document. The XML patch document format builds on the foundations defined in RFC 5261. This specification also provides the media type registration "application/xml-patch+xml", to allow the use of XML patch documents in, for example, HTTP conversations.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7351.

Table of Contents

1. Introduction

The Extensible Markup Language (XML) [RFC7303] is a common format for the exchange and storage of structured data. HTTP PATCH [RFC5789] extends HTTP [RFC7231] with a method to perform partial modifications to resources. HTTP PATCH requires that patch documents be sent along with the request, and it is therefore useful for there to be standardized patch document formats (identified by media types) for popular media types.

The XML patch media type "application/xml-patch+xml" is an XML document structure for expressing a sequence of operations to apply to a target XML document, suitable for use with the HTTP PATCH method. Servers can freely choose which patch formats they want to accept, and "application/xml-patch+xml" could be a simple default format that can be used unless a server decides to use a different (maybe more sophisticated) patch format for XML.

The format for patch documents is based on the XML patch framework defined in RFC 5261 [RFC5261]. While RFC 5261 does define a concrete syntax as well as the media type "application/patch-ops-error+xml" for error documents, it only defines XML Schema (XSD) [W3C.REC-xmlschema-1-20041028] types for patch operations. The concrete document format and the media type for patch operations are defined in an XSD defined in this specification.

This specification relies on RFC 5261 but also requires that errata reported to date are taken into account. The main reason for the errata is the problematic ways in which RFC 5261 relies on XML Path Language (XPath) as the expression language for selecting the location of a patch, while at the same time XPath’s data model does

not contain sufficient information to determine whether such a selector indeed can be used for a patch operation or should result in an error. Specifically, the problem occurs with namespaces, where XPath does not expose namespace declaration attributes, while the patch model needs them to determine whether or not a namespace patch is allowed. Appendix A contains more information about the general problem and errata reports.

2. Patch Documents

The following sections describe and illustrate the XML patch document format.

2.1. Patch Document Format

The XML patch document format is based on a simple schema that uses a "patch" element as the document element and allows an arbitrary sequence of "add", "remove", and "replace" elements as the children of the document element. These children follow the semantics defined in RFC 5261, which means that each element is treated as an individual patch operation, and the result of each patch operation is a patched XML document that is the target XML document for the next patch operation.

The following simple example patch document contains a single patch operation. This operation adds a new attribute called "new-attribute" to the document element of the target XML document. An XML patch document always uses a "patch" element in the "urn:ietf:rfc:7351" namespace as the document element that contains zero or more patch operation elements, which are also in the "urn:ietf:rfc:7351" namespace.

```
<p:patch xmlns:p="urn:ietf:rfc:7351">
    <p:add sel="*" type="@new-attribute">value</p:add>
</p:patch>
```

The following more complex example patch document uses the example from RFC 5261, Section A.18 (but changing the example namespaces to example.com URIs); it uses the same "patch" element and XML namespace as shown in the simpler example. It shows the general structure of an XML patch document with multiple operations, as well as an example of each operation.

```
<p:patch xmlns="http://example.com/ns1"
         xmlns:y="http://example.com/ns2"
         xmlns:p="urn:ietf:rfc:7351">
    <p:add sel="doc/elem[@a='foo']">
        <!-- This is a new child -->
        <child id="ert4773">
            <y:node/>
        </child>
    </p:add>
    <p:replace sel="doc/note/text()">Patched doc</p:replace>
    <p:remove sel="*/elem[@a='bar']/y:child" ws="both"/>
    <p:add sel="*/elem[@a='bar']" type="@b">new attr</p:add>
</p:patch>
```

As this example demonstrates, both the document element "patch" and the patch operation elements are in the same XML namespace. This is the result of RFC 5261 only defining types for the patch operation elements, which then can be reused in schemas to define concrete patch elements.

RFC 5261 defines XSD [W3C.REC-xmlschema-1-20041028] for the patch operation types. The following schema for the XML patch media type is based on the types defined in RFC 5261, which are imported as "rfc5261.xsd" in the following schema. The schema defines a "patch" document element, and then allows an unlimited (and possibly empty) sequence of the "add", "remove", and "replace" operation elements, which are directly based on the respective types from the schema defined in RFC 5261.

```
<xs:schema targetNamespace="urn:ietf:rfc:7351"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:import schemaLocation="rfc5261.xsd"/>
    <xs:element name="patch">
        <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="add" type="add"/>
                <xs:element name="remove" type="remove"/>
                <xs:element name="replace" type="replace"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

2.2. Patch Examples

Since the semantics of the XML patch operations are defined by RFC 5261, please refer to the numerous examples in that specification for more XML patch document examples. All the examples in RFC 5261 can be taken as examples for the XML patch media type, when looking at them with two minor changes in mind.

The two differences are that XML patch documents always use the "patch" element as the document element and that both the "patch" element and the individual operation elements in XML patch documents have to be in the XML namespace with the URI "urn:ietf:rfc:7351".

For example, consider the patch example in RFC 5261, Appendix A.1, "Adding an Element". In this example, the patch is applied to the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<doc>
    <note>This is a sample document</note>
</doc>
```

The patch example is based on the following patch document (with the element and namespace changes described above):

```
<?xml version="1.0" encoding="UTF-8"?>
<p:patch xmlns:p="urn:ietf:rfc:7351">
    <p:add sel="doc"><foo id="ert4773">This is a new child</foo></p:add>
</p:patch>
```

Applying the patch results in the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<doc>
    <note>This is a sample document</note>
<foo id="ert4773">This is a new child</foo></doc>
```

3. IANA Considerations

The Internet media type [RFC6838] for an XML patch document is application/xml-patch+xml.

Type name: application

Subtype name: xml-patch+xml

Required parameters: none

Optional parameters:

charset: Same as charset parameter for the media type "application/xml" as specified in RFC 7303 [RFC7303].

Encoding considerations: Same as encoding considerations of media type "application/xml" as specified in RFC 7303 [RFC7303].

Security considerations: This media type has all of the security considerations described in RFC 7303 [RFC7303], RFC 5261 [RFC5261], and RFC 3470 [RFC3470], plus those listed in Section 4.

Interoperability considerations: N/A

Published specification: RFC 7351

Applications that use this media type: Applications that manipulate XML documents.

Additional information:

Magic number(s): N/A

File extension(s): XML documents often use ".xml" as the file extension, and this media type does not propose a specific extension other than this generic one.

Macintosh file type code(s): TEXT

Person & email address to contact for further information: Erik Wilde <dret@berkeley.edu>

Intended usage: COMMON

Restrictions on usage: none

Author: Erik Wilde <dret@berkeley.edu>

Change controller: IETF

4. Security Considerations

The security considerations from RFC 5261 [RFC5261] apply to the application/xml-patch+xml media type.

In addition, parsing XML may entail including information from external sources through XML's mechanism of external entities. Implementations, therefore, should be aware of the fact that standard parsers may resolve external entities and thus include external information as a result of applying patch operations to an XML document.

5. Acknowledgements

Thanks for comments and suggestions provided by Bas de Bakker, Tony Hansen, Bjoern Hoehrmann, and Julian Reschke.

6. References

6.1. Normative References

[RFC3470] Hollenbeck, S., Rose, M., and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols", BCP 70, RFC 3470, January 2003.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5261] Urpalainen, J., "An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors", RFC 5261, September 2008.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[RFC7303] Thompson, H. and C. Lilley, "XML Media Types", RFC 7303, July 2014.

6.2. Informative References

[Err3477] RFC Errata, "Errata ID 3477", RFC 5261.

[Err3478] RFC Errata, "Errata ID 3478", RFC 5261.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[W3C.REC-DOM-Level-3-Core-20040407]
Robie, J., Wood, L., Champion, M., Hegaret, P., Nicol, G., Le Hors, A., and S. Byrne, "Document Object Model (DOM) Level 3 Core Specification", World Wide Web Consortium Recommendation REC-DOM-Level-3-Core-20040407, April 2004, <http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407>.

[W3C.REC-xml-20081126]
Sperberg-McQueen, C., Yergeau, F., Paoli, J., Maler, E., and T. Bray, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

[W3C.REC-xml-names-20091208]
Hollander, D., Layman, A., Bray, T., Tobin, R., and H. Thompson, "Namespaces in XML 1.0 (Third Edition)", World Wide Web Consortium Recommendation REC-xml-names-20091208, December 2009, <http://www.w3.org/TR/2009/REC-xml-names-20091208>.

[W3C.REC-xmlschema-1-20041028]
Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>.

[W3C.REC-xpath-19991116]
DeRose, S. and J. Clark, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

[W3C.REC-xpath20-20101214]
Boag, S., Berglund, A., Kay, M., Simeon, J., Robie, J., Chamberlin, D., and M. Fernandez, "XML Path Language (XPath) 2.0 (Second Edition)", World Wide Web Consortium Recommendation REC-xpath20-20101214, December 2010, <http://www.w3.org/TR/2010/REC-xpath20-20101214>.

Appendix A. Implementation Hints

This section is informative. It describes some issues that might be interesting for implementers, but it might also be interesting for users of XML patch that want to understand some of the differences between standard XPath 1.0 processing and the processing model of selectors in RFC 5261.

Specifically, the issues described in the following two sections have been identified as technical issues with RFC 5261 and have been filed as errata. Implementers interested in using XML patch are encouraged to take those errata into account when implementing XML patch documents. The issue about "Matching Namespaces" described in Appendix A.1 has been filed as RFC Errata ID 3477 [Err3477]. The issue about "Patching Namespaces" described in Appendix A.2 has been filed as RFC Errata ID 3478 [Err3478].

A.1. Matching Namespaces

RFC 5261 defines standard rules for matching prefixed names in expressions: any prefixes are interpreted according to the namespace bindings of the diff document (the document that the expression is applied against). This means that each prefixed name can be interpreted in the context of the diff document.

For unprefixed names in expressions, the rules depart from XPath 1.0 [W3C.REC-xpath-19991116]. XPath 1.0 defines that unprefixed names in expressions match namespace-less names (i.e., there is no "default namespace" for names used in XPath 1.0 expressions). RFC 5261 requires, however, that unprefixed names in expressions must use the default namespace of the diff document (if there is one). This means that it is not possible to simply take a selector from a patch document and evaluate it in the context of the diff document according to the rules of XPath 1.0 because this would interpret unprefixed names incorrectly. As a consequence, it is not possible to simply take an XPath 1.0 processor and evaluate XML patch selectors in the context of the diff document.

As an extension of XPath 1.0’s simple model, XPath 2.0 [W3C.REC-xpath20-20101214] specifies different processing rules for unprefixed names: they are matched against the URI of the "default element/type namespace", which is defined as part of an expression’s static context. In some XPath 2.0 applications, this can be set; XSL Transformations (XSLT) 2.0, for example, has the ability to define an "xpath-default-namespace", which then will be used to match unprefixed names in expressions. Thus, by using an XPath 2.0 implementation that allows one to set this URI, and setting it to the default namespace of the diff document (or leaving it undefined if

there is no such default namespace), it is possible to use an out-of-the-box XPath 2.0 implementation for evaluating XML patch selectors.

Please keep in mind, however, that evaluating selectors is only one part of applying patches. When it comes to applying the actual patch operation, neither XPath 1.0 nor XPath 2.0 are sufficient because they do not preserve some of the information from the XML syntax (specifically namespace declarations) that is required to correctly apply patch operations. The following section describes this issue in more detail.

Please note that [RFC5261], Section 4.2.2 on namespace matching explains XPath 2.0's rules incorrectly. For this reason, RFC Errata ID 3477 is available for Section 4.2.2 of RFC 5261.

A.2. Patching Namespaces

One of the issues when patching namespaces based on XPath is that XPath exposes namespaces differently than the XML 1.0 [W3C.REC-xml-20081126] syntax for XML namespaces [W3C.REC-xml-names-20091208]. In the XML syntax, a namespace is declared with an attribute using the reserved name or prefix "xmlns", and this results in this namespace being available recursively through the document tree. In XPath, the namespace declaration is not exposed as an attribute (i.e., the attribute, although syntactically an XML attribute, is not accessible in XPath), but the resulting namespace nodes are exposed recursively through the tree.

RFC 5261 uses the terms "namespace declaration" and "namespace" almost interchangeably, but it is important to keep in mind that the namespace declaration is an XML syntax construct that is unavailable in XPath, while the namespace itself is a logical construct that is not visible in the XML syntax, but a result of a namespace declaration. The intent of RFC 5261 is to patch namespaces as if namespace declarations were patched; thus, it only allows patching namespace nodes on the element nodes where the namespace has been declared.

Patching namespaces in XML patch is supposed to "emulate" the effect of actually changing the namespace declaration (which is why a namespace can only be patched at the element where it has been declared). Therefore, when patching a namespace, even though XPath's "namespace" axis is used, implementations have to make sure that not only the single selected namespace node is being patched but that all namespaces nodes resulting from the namespace declaration of this namespace are also patched accordingly.

This means that an implementation might have to descend into the tree, matching all namespace nodes with the selected prefix/URI pair recursively, until it encounters leaf elements or namespace declarations with the same prefix it is patching. Determining this requires access to the diff document beyond XPath, because, in XPath itself, namespace declarations are not represented; thus, such a recursive algorithm wouldn't know when to stop. Consider the following document:

```
<x xmlns:a="tag:42">
    <y xmlns:a="tag:42"/>
</x>
```

If this document is patched with a selector of /x/namespace::a, then only the namespace node on element x should be patched, even though the namespace node on element y has the same prefix/URI combination as the one on element x. However, determining that the repeated namespace declaration was present at all on element y is impossible when using XPath alone, which means that implementations must have an alternative way to determine the difference between the document above, and this one:

```
<x xmlns:a="tag:42">
    <y/>
</x>
```

In this second example, patching with a selector of /x/namespace::a should indeed change the namespace nodes on elements x and y, because they both have been derived from the same namespace declaration.

The conclusion of these considerations is that for implementing XML patch, access closer to the XML syntax (specifically access to namespace declarations) is necessary. As a result, implementations attempting to exclusively use the XPath model for implementing XML patch will fail to correctly address certain edge cases (such as the one shown above).

Note that XPath's specific limitations do not mean that it is impossible to use XML technologies other than XPath. The Document Object Model (DOM) [W3C.REC-DOM-Level-3-Core-20040407], for example, does expose namespace declaration attributes as regular attributes in the document tree; thus, they could be used to differentiate between the two variants shown above.

Please note that RFC 5261, Section 4.4.3 (on replacing namespaces) mixes the terms "namespace declaration" and "namespace". For this reason, RFC Errata ID 3478 is available for Section 4.4.3 of RFC 5261.

Appendix B. ABNF for RFC 5261

RFC 5261 [RFC5261] does not contain an ABNF grammar for the allowed subset of XPath expressions but includes an XSD-based grammar in its type definition for operation types. In order to make implementation easier, this appendix contains an ABNF grammar that has been derived from the XSD expressions in RFC 5261. In the following grammar, "xpath" is the definition for the allowed XPath expressions for remove and replace operations, and "xpath-add" is the definition for the allowed XPath expressions for add operations. The names of all grammar productions are the ones used in the XSD-based grammar of RFC 5261.

```
anychar    =  %x00-ffffffff
ncname     =  1*%x00-ffffffff
qname      =  [ ncname ":" ] ncname
aname      =  "@" qname
pos        =  "[" 1*DIGIT "]"
attr       =  ( "[" aname "=’" 0*anychar "’]" ) /
              ( "[" aname "=" DQUOTE 0*anychar DQUOTE "]" )
valueq     =  "[" ( qname / "." ) "=" DQUOTE 0*anychar DQUOTE "]"
value      =  ( "[" ( qname / "." ) "=’" 0*anychar "’]" ) / valueq
cond       =  attr / value / pos
step       =  ( qname / "*" ) 0*cond
piq        =  %x70.72.6f.63.65.73.73.69.6e.67.2d
              %x69.6e.73.74.72.75.63.74.69.6f.6e
              ; "processing-instruction", case-sensitive
              "(" [ DQUOTE ncname DQUOTE ] ")"
pi         =  ( %x70.72.6f.63.65.73.73.69.6e.67.2d
              %x69.6e.73.74.72.75.63.74.69.6f.6e
              ; "processing-instruction", case-sensitive
              "(" [ "’" ncname "’" ] ")" ) / piq
id         =  ( %x69.64 ; "id", case-sensitive
              "(" [ "’" ncname "’" ] ")" ) /
              ( %x69.64 ; "id", case-sensitive
              "(" [ DQUOTE ncname DQUOTE ] ")" )
com        =  %x63.6f.6d.6d.65.6e.74 ; "comment", case-sensitive
              "()"
text       =  %x74.65.78.74 ; "text", case-sensitive
              "()"
nspa       =  %x6e.61.6d.65.73.70.61.63.65 ; "namespace", case-sensitive
              "::" ncname
cnodes     =  ( text / com / pi ) [ pos ]
child      =  cnodes / step
last       =  child / aname / nspa
xpath      =  [ "/" ] ( ( id [ 0*( "/" step ) "/" last ] ) /
              ( 0*( step "/" ) last ) )
xpath-add  =  [ "/" ] ( ( id [ 0*( "/" step ) "/" child ] ) /
              ( 0*( step "/" ) child ) )
```

Please note that the "ncname" production listed above does not fully capture the constraints of the original XSD-based definition, where it is defined as "\i\c*". DIGIT and DQUOTE are defined by the ABNF specification [RFC5234].

Author's Address

Erik Wilde
UC Berkeley

EMail: dret@berkeley.edu
URI: http://dret.net/netdret/