Internet Architecture Board (IAB)
Request for Comments: 7998
Category: Informational
ISSN: 2070-1721

P. Hoffman
ICANN
J. Hildebrand
Mozilla

December 2016

"xml2rfc" Version 3 Preparation Tool Description

Abstract

This document describes some aspects of the "prep tool" that is expected to be created when the new xml2rfc version 3 specification is deployed.

Status of This Memo

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

This document is a product of the Internet Architecture Board (IAB) and represents information that the IAB has deemed valuable to provide for permanent record. It represents the consensus of the Internet Architecture Board (IAB). Documents approved for publication by the IAB are not a candidate for any level of Internet Standard; see Section 2 of RFC 7841.

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/rfc7998.

Table of Contents

1. Introduction

As initially described in [RFC6949], the canonical format (the data that is the authorized, recognized, accepted, and archived version of the document) of the RFC Series has been plain text to date: it is now changing to XML (using the xml2rfc v3 vocabulary [RFC7991]).

However, most people will read RFCs in other formats, such as HTML, PDF, ASCII text, or other formats not yet in existence. In order to ensure as much uniformity in text output as possible across formats (and with the canonical XML itself), there is a desire that the translation from XML into the other formats will be straightforward syntactic translation. To make that happen, a good amount of data will need to be in the XML format that is not there today. That data will be added by a program called the "prep tool", which will often run as a part of the xml2rfc process.

This document specifies the steps that the prep tool will have to take. When changes to the xml2rfc v3 vocabulary [RFC7991] are made, this document is likely to be updated at the same time.

The details (particularly any vocabularies) described in this document are expected to change based on experience gained in implementing the new publication toolsets. Revised documents will be published capturing those changes as the toolsets are completed. Other implementers must not expect those changes to remain backwards-compatible with the details described in this document.

2. xml2rfc v3 Prep Tool Usage Scenarios

The prep tool will have several settings:

- Internet-Draft preparation
- Canonical RFC preparation

There are only a few differences between the two settings: for example, the boilerplate output and the date output on the front page.

Note that this document only describes what the IETF-sponsored prep tool does. Others might create their own work-alike prep tools for their own formatting needs. However, an output format developer does not need to change the prep tool in order to create their own formatter: they only need to be able to consume prepared text. The IETF-sponsored prep tool runs in two different modes: "I-D" mode when the tool is run during Internet-Draft submission and processing and "RFC production mode" when the tool is run by the RFC Production Center while producing an RFC.

This tool is described as if it is a separate tool so that we can reason about its architectural properties. In actual implementation, it might be a part of a larger suite of functionality.

3. Internet-Draft Submission

When the IETF draft submission tool accepts xml2rfc version 3 vocabulary [RFC7991] (referred to as "v3" hereafter) as an input format, the submission tool runs the submitted file through the prep tool. This is called "I-D mode" in this document. If the tool finds no errors, it keeps two XML files: the submitted file and the prepped file.

The prepped file provides a record of what a submitter was attesting to at the time of submission. It represents a self-contained record of what any external references resolved to at the time of submission.

The prepped file is used by the IETF formatters to create outputs such as HTML, PDF, and text (or the tools act in a way indistinguishable from this). The message sent out by the draft submission tool includes a link to the submitted XML as well as the other outputs, including the prepped XML.

The prepped XML can be used by tools not yet developed to output new formats that have as similar output as possible to the current IETF formatters. For example, if the IETF creates a .mobi output renderer later, it can run that renderer on all of the prepped XML that has been saved, ensuring that the content of included external references and all of the part numbers and boilerplate will be the same as what was produced by the previous IETF formatters at the time the document was first uploaded.

4. Canonical RFC Preparation

During editing, the RPC will run the prep tool in canonical RFC production mode and make the results available to the authors during AUTH48 (see [PUB-PROCESS]) so they can see what the final output would look like. When the document has passed AUTH48 review, the RPC runs the prep tool in canonical RFC production mode one last time, locks down the canonicalized XML, runs the formatters for the publication formats, and publishes all of those.

This document assumes that the prep tool will be used by the RPC in the manner described in this document; they may use something different or with different configuration.

Similar to the case for I-Ds, the prepped XML can be used later to re-render the output formats or to generate new formats.

5. What the v3 Prep Tool Does

The steps listed here are in order of processing. In all cases where the prep tool would "add" an attribute or element, if that attribute or element already exists, the prep tool will check that the attribute or element has valid values. If the value is incorrect, the prep tool will warn with the old and new values, then replace the incorrect value with the new value.

Currently, the IETF uses a tool called "idnits" [IDNITS] to check text input to the Internet-Drafts posting process. idnits indicates if it encountered anything it considers an error and provides text describing all of the warnings and errors in a human-readable form. The prep tool should probably check for as many of these errors and warnings as possible when it is processing the XML input. For the moment, tooling might run idnits on the text output from the prepared XML. The list below contains some of these errors and warnings, but the deployed version of the prep tool may contain additional steps to include more or the checks from idnits.

5.1. XML Sanitization

These steps will ensure that the input document is properly formatted and that all XML processing has been performed.

5.1.1. XInclude Processing

Process all <x:include> elements. Note: XML <x:include> elements may include more <x:include> elements (with relative references resolved against the base URI potentially modified by a previously inserted xml:base attribute). The tool may be configurable with a limit on the depth of recursion.

5.1.2. DTD Removal

Fully process any Document Type Definitions (DTDs) in the input document, then remove the DTD. At a minimum, this entails processing the entity references and includes for external files.

5.1.3. Processing Instruction Removal

Remove processing instructions.

5.1.4. Validity Check

Check the input against the RELAX NG (RNG) in [RFC7991]. If the input is not valid, give an error.

5.1.5. Check "anchor"

Check all elements for "anchor" attributes. If any "anchor" attribute begins with "s-", "f-", "t-", or "i-", give an error.

5.2. Defaults

These steps will ensure that all default values have been filled in to the XML, in case the defaults change at a later date. Steps in this section will not overwrite existing values in the input file.

5.2.1. "version" Insertion

If the <rfc> element has a "version" attribute with a value other than "3", give an error. If the <rfc> element has no "version" attribute, add one with the value "3".

5.2.2. "seriesInfo" Insertion

If the <front> element of the <rfc> element does not already have a <seriesInfo> element, add a <seriesInfo> element with the name attribute based on the mode in which the prep tool is running ("Internet-Draft" for Draft mode and "RFC" for RFC production mode) and a value that is the input filename minus any extension for Internet-Drafts, and is a number specified by the RFC Editor for RFCs.

5.2.3. <date> Insertion

If the <front> element in the <rfc> element does not contain a <date> element, add it and fill in the "day", "month", and "year" attributes from the current date. If the <front> element in the <rfc> element has a <date> element with "day", "month", and "year" attributes, but the date indicated is more than three days in the past or is in the future, give a warning. If the <front> element in the <rfc> element has a <date> element with some but not all of the "day", "month", and "year" attributes, give an error.

5.2.4. "prepTime" Insertion

If the input document includes a "prepTime" attribute of <rfc>, exit with an error.

Fill in the "prepTime" attribute of <rfc> with the current datetime.

5.2.5. Group "start" Insertion

Add a "start" attribute to every <ol> element containing a group that does not already have a start.

5.2.6. Attribute Default Value Insertion

Fill in any default values for attributes on elements, except "keepWithNext" and "keepWithPrevious" of <t>, and "toc" of <section>. Some default values can be found in the RELAX NG schema, while others can be found in the prose describing the elements in [RFC7991].

5.2.7. Section "toc" attribute

For each <section>, modify the "toc" attribute to be either "include" or "exclude":

- for sections that have an ancestor of <boilerplate>, use "exclude"

- else for sections that have a descendant that has toc="include", use "include". If the ancestor section has toc="exclude" in the input, this is an error.

- else for sections that are children of a section with toc="exclude", use "exclude".

- else for sections that are deeper than rfc/@tocDepth, use "exclude"

- else use "include"

5.2.8. "removeInRFC" Warning Paragraph

In I-D mode, if there is a <note> or <section> element with a "removeInRFC" attribute that has the value "true", add a paragraph to the top of the element with the text "This note is to be removed before publishing as an RFC." or "This section...", unless a paragraph consisting of that exact text already exists.

5.3. Normalization

These steps will ensure that ideas that can be expressed in multiple different ways in the input document are only found in one way in the prepared document.

5.3.1. "month" Attribute

Normalize the values of "month" attributes in all <date> elements in <front> elements in <rfc> elements to numeric values.

5.3.2. ASCII Attribute Processing

In every <email>, <organization>, <street>, <city>, <region>, <country>, and <code> element, if there is an "ascii" attribute and the value of that attribute is the same as the content of the element, remove the "ascii" element and issue a warning about the removal.

In every <author> element, if there is an "asciiFullname", "asciiInitials", or "asciiSurname" attribute, check the content of that element against its matching "fullname", "initials", or "surname" element (respectively). If the two are the same, remove the "ascii*" element and issue a warning about the removal.

5.3.3. "title" Conversion

For every <section>, <note>, <figure>, <references>, and <texttable> element that has a (deprecated) "title" attribute, remove the "title" attribute and insert a <name> element with the title from the attribute.

5.4. Generation

These steps will generate new content, overriding existing similar content in the input document. Some of these steps are important enough that they specify a warning to be generated when the content being overwritten does not match the new content.

5.4.1. "expiresDate" Insertion

If in I-D mode, fill in "expiresDate" attribute of <rfc> based on the <date> element of the document's <front> element.

5.4.2. <boilerplate> Insertion

Create a <boilerplate> element if it does not exist. If there are any children of the <boilerplate> element, produce a warning that says "Existing boilerplate being removed. Other tools, specifically the draft submission tool, will treat this condition as an error" and remove the existing children.

5.4.2.1. Compare <rfc> "submissionType" and <seriesInfo> "stream"

Verify that <rfc> "submissionType" and <seriesInfo> "stream" are the same if they are both present. If either is missing, add it. Note that both have a default value of "IETF".

5.4.2.2. "Status of this Memo" Insertion

Add the "Status of this Memo" section to the <boilerplate> element with current values. The application will use the "submissionType", and "consensus" attributes of the <rfc> element, the <workgroup> element, and the "status" and "stream" attributes of the <seriesInfo> element, to determine which boilerplate from [RFC7841] to include, as described in Appendix A of [RFC7991].

5.4.2.3. "Copyright Notice" Insertion

Add the "Copyright Notice" section to the <boilerplate> element. The application will use the "ipr" and "submissionType" attributes of the <rfc> element and the <date> element to determine which portions and which version of the Trust Legal Provisions (TLP) to use, as described in A.1 of [RFC7991].

5.4.3. <reference> "target" Insertion

For any <reference> element that does not already have a "target" attribute, fill the target attribute in if the element has one or more <seriesinfo> child element(s) and the "name" attribute of the <seriesinfo> element is "RFC", "Internet-Draft", or "DOI" or other value for which it is clear what the "target" should be. The particular URLs for RFCs, Internet-Drafts, and Digital Object Identifiers (DOIs) for this step will be specified later by the RFC Editor and the IESG. These URLs might also be different before and after the v3 format is adopted.

5.4.4. <name> Slugification

Add a "slugifiedName" attribute to each <name> element that does not contain one; replace the attribute if it contains a value that begins with "n-".

5.4.5. <reference> Sorting

If the "sortRefs" attribute of the <rfc> element is true, sort the <reference> and <referencegroup> elements lexically by the value of the "anchor" attribute, as modified by the "to" attribute of any <displayreference> element. The RFC Editor needs to determine what the rules for lexical sorting are. The authors of this document acknowledge that getting consensus on this will be a difficult task.

5.4.6. "pn" Numbering

Add "pn" attributes for all parts. Parts are:

- <section> in <middle>: pn=’s-1.4.2’
- <references>: pn=’s-12’ or pn=’s-12.1’
- <abstract>: pn=’s-abstract’
- <note>: pn=’s-note-2’
- <section> in <boilerplate>: pn=’s-boilerplate-1’

- <table>: pn='t-3'
- <figure>: pn='f-4'
- <artwork>, <aside>, <blockquote>, <dt>, <li>, <sourcecode>, <t>: pn='p-[section]-[counter]'

5.4.7. <iref> Numbering

In every <iref> element, create a document-unique "pn" attribute. The value of the "pn" attribute will start with 'i-', and use the item attribute, the subitem attribute (if it exists), and a counter to ensure uniqueness. For example, the first instance of "<iref item='foo' subitem='bar'>" will have the "irefid" attribute set to 'i-foo-bar-1'.

5.4.8. <xref> Processing

5.4.8.1. "derivedContent" Insertion (with Content)

For each <xref> element that has content, fill the "derivedContent" with the element content, having first trimmed the whitespace from ends of content text. Issue a warning if the "derivedContent" attribute already exists and has a different value from what was being filled in.

5.4.8.2. "derivedContent" Insertion (without Content)

For each <xref> element that does not have content, fill the "derivedContent" attribute based on the "format" attribute.

- For a value of "counter", the "derivedContent" is set to the section, figure, table, or ordered list number of the element with an anchor equal to the <xref> target.
- For format='default' and the "target" attribute points to a <reference> or <referencegroup> element, the "derivedContent" is the value of the "target" attribute (or the "to" attribute of a <displayreference> element for the targeted <reference>).
- For format='default' and the "target" attribute points to a <section>, <figure>, or <table>, the "derivedContent" is the name of the thing pointed to, such as "Section 2.3", "Figure 12", or "Table 4".
- For format='title', if the target is a <reference> element, the "derivedContent" attribute is the name of the reference, extracted from the <title> child of the <front> child of the reference.

- For format='title', if the target element has a <name> child element, the "derivedContent" attribute is the text content of that <name> element concatenated with the text content of each descendant node of <name> (that is, stripping out all of the XML markup, leaving only the text).

- For format='title', if the target element does not contain a <name> child element, the "derivedContent" attribute is the value of the "target" attribute with no other adornment. Issue a warning if the "derivedContent" attribute already exists and has a different value from what was being filled in.

5.4.9. <relref> Processing

If any <relref> element's "target" attribute refers to anything but a <reference> element, give an error.

For each <relref> element, fill in the "derivedLink" attribute.

5.5. Inclusion

These steps will include external files into the output document.

5.5.1. <artwork> Processing

1. If an <artwork> element has a "src" attribute where no scheme is specified, copy the "src" attribute value to the "originalSrc" attribute, and replace the "src" value with a URI that uses the "file:" scheme in a path relative to the file being processed. See Section 7 for warnings about this step. This will likely be one of the most common authoring approaches.

2. If an <artwork> element has a "src" attribute with a "file:" scheme, and if processing the URL would cause the processor to retrieve a file that is not in the same directory, or a subdirectory, as the file being processed, give an error. If the "src" has any shellmeta strings (such as "'", "$USER", and so on) that would be processed, give an error. Replace the "src" attribute with a URI that uses the "file:" scheme in a path relative to the file being processed. This rule attempts to prevent <artwork src='file:///etc/passwd'> and similar security issues. See Section 7 for warnings about this step.

3. If an <artwork> element has a "src" attribute, and the element has content, give an error.

4. If an <artwork> element has type='svg' and there is an "src" attribute, the data needs to be moved into the content of the <artwork> element.

   * If the "src" URI scheme is "data:", fill the content of the <artwork> element with that data and remove the "src" attribute.

   * If the "src" URI scheme is "file:", "http:", or "https:", fill the content of the <artwork> element with the resolved XML from the URI in the "src" attribute. If there is no "originalSrc" attribute, add an "originalSrc" attribute with the value of the URI and remove the "src" attribute.

   * If the <artwork> element has an "alt" attribute, and the SVG does not have a <desc> element, add the <desc> element with the contents of the "alt" attribute.

5. If an <artwork> element has type='binary-art', the data needs to be in an "src" attribute with a URI scheme of "data:". If the "src" URI scheme is "file:", "http:", or "https:", resolve the URL. Replace the "src" attribute with a "data:" URI, and add an "originalSrc" attribute with the value of the URI. For the "http:" and "https:" URI schemes, the mediatype of the "data:" URI will be the Content-Type of the HTTP response. For the "file:" URI scheme, the mediatype of the "data:" URI needs to be guessed with heuristics (this is possibly a bad idea). This also fails for content that includes binary images but uses a type other than "binary-art". Note: since this feature can't be used for RFCs at the moment, this entire feature might be

6. If an <artwork> element does not have type='svg' or type='binary-art' and there is an "src" attribute, the data needs to be moved into the content of the <artwork> element. Note that this step assumes that all of the preferred types other than "binary-art" are text, which is possibly wrong.

   * If the "src" URI scheme is "data:", fill the content of the <artwork> element with the correctly escaped form of that data and remove the "src" attribute.

   * If the "src" URI scheme is "file:", "http:", or "https:", fill the content of the <artwork> element with the correctly escaped form of the resolved text from the URI in the "src" attribute. If there is no "originalSrc" attribute, add an "originalSrc" attribute with the value of the URI and remove the "src" attribute.

5.5.2. <sourcecode> Processing

1. If a <sourcecode> element has a "src" attribute where no scheme is specified, copy the "src" attribute value to the "originalSrc" attribute and replace the "src" value with a URI that uses the "file:" scheme in a path relative to the file being processed. See Section 7 for warnings about this step. This will likely be one of the most common authoring approaches.

2. If a <sourcecode> element has a "src" attribute with a "file:" scheme, and if processing the URL would cause the processor to retrieve a file that is not in the same directory, or a subdirectory, as the file being processed, give an error. If the "src" has any shellmeta strings (such as "’", "$USER", and so on) that would be processed, give an error. Replace the "src" attribute with a URI that uses the "file:" scheme in a path relative to the file being processed. This rule attempts to prevent <sourcecode src=’file:///etc/passwd’> and similar security issues. See Section 7 for warnings about this step.

3. If a <sourcecode> element has a "src" attribute, and the element has content, give an error.

4. If a <sourcecode> element has a "src" attribute, the data needs to be moved into the content of the <sourcecode> element.

   * If the "src" URI scheme is "data:", fill the content of the <sourcecode> element with that data and remove the "src" attribute.

   * If the "src" URI scheme is "file:", "http:", or "https:", fill the content of the <sourcecode> element with the resolved XML from the URI in the "src" attribute. If there is no "originalSrc" attribute, add an "originalSrc" attribute with the value of the URI and remove the "src" attribute.

5.6. RFC Production Mode Cleanup

These steps provide extra cleanup of the output document in RFC production mode.

5.6.1. <note> Removal

In RFC production mode, if there is a <note> or <section> element with a "removeInRFC" attribute that has the value "true", remove the element.

5.6.2. <cref> Removal

If in RFC production mode, remove all <cref> elements.

5.6.3. <link> Processing

1. If in RFC production mode, remove all <link> elements whose "rel" attribute has the value "alternate".

2. If in RFC production mode, check if there is a <link> element with the current ISSN for the RFC series (2070-1721); if not, add <link rel="item" href="urn:issn:2070-1721">.

3. If in RFC production mode, check if there is a <link> element with a DOI for this RFC; if not, add one of the form <link rel="describedBy" href="https://dx.doi.org/10.17487/rfcdd"> where "dd" is the number of the RFC, such as "https://dx.doi.org/10.17487/rfc2109". The URI is described in [RFC7669]. If there was already a <link> element with a DOI for this RFC, check that the "href" value has the right format. The content of the href attribute is expected to change in the future.

4. If in RFC production mode, check if there is a <link> element with the file name of the Internet-Draft that became this RFC the form <link rel="convertedFrom" href="https://datatracker.ietf.org/doc/draft-ttttttttt/">. If one does not exist, give an error.

5.6.4. XML Comment Removal

If in RFC production mode, remove XML comments.

5.6.5. "xml:base" and "originalSrc" Removal

If in RFC production mode, remove all "xml:base" or "originalSrc" attributes from all elements.

5.6.6. Compliance Check

If in RFC production mode, ensure that the result is in full compliance to the v3 schema, without any deprecated elements or attributes and give an error if any issues are found.

5.7. Finalization

These steps provide the finishing touches on the output document.

5.7.1. "scripts" Insertion

Determine all the characters used in the document and fill in the "scripts" attribute for <rfc>.

5.7.2. Pretty-Format

Pretty-format the XML output. (Note: there are many tools that do an adequate job.)

6. Additional Uses for the Prep Tool

There will be a need for Internet-Draft authors who include files from their local disk (such as for <artwork src="mydrawing.svg"/>) to have the contents of those files inlined to their drafts before submitting them to the Internet-Draft processor. (There is a possibility that the Internet-Draft processor will allow XML files and accompanying files to be submitted at the same time, but this seems troublesome from a security, portability, and complexity standpoint.) For these users, having a local copy of the prep tool that has an option to just inline all local files would be terribly useful. That option would be a proper subset of the steps given in Section 5.

A feature that might be useful in a local prep tool would be the inverse of the "just inline" option would be "extract all". This would allow a user who has a v3 RFC or Internet-Draft to dump all of the <artwork> and <sourcecode> elements into local files instead of having to find each one in the XML. This option might even do as much validation as possible on the extracted <sourcecode> elements. This feature might also remove some of the features added by the prep tool (such as part numbers and "slugifiedName" attributes starting with "n-") in order to make the resulting file easier to edit.

7. Security Considerations

Steps in this document attempt to prevent the <artwork> and <sourcecode> entities from exposing the contents of files outside the directory in which the document being processed resides. For example, values starting with "/", "./", or "../" should generate errors.

The security considerations in [RFC3470] apply here. Specifically, processing XML-external references can expose a prep-tool implementation to various threats by causing the implementation to access external resources automatically. It is important to disallow arbitrary access to such external references within XML data from untrusted sources.

8. Informative References

[IDNITS] IETF Tools, "Idnits Tool", <https://tools.ietf.org/tools/idnits/>.

[PUB-PROCESS] RFC Editor, "Publication Process", <https://www.rfc-editor.org/pubprocess/>.

[RFC3470] Hollenbeck, S., Rose, M., and L. Masinter, "Guidelines for the Use of Extensible Markup Language (XML) within IETF Protocols", BCP 70, RFC 3470, DOI 10.17487/RFC3470, January 2003, <http://www.rfc-editor.org/info/rfc3470>.

[RFC6949] Flanagan, H. and N. Brownlee, "RFC Series Format Requirements and Future Development", RFC 6949, DOI 10.17487/RFC6949, May 2013, <http://www.rfc-editor.org/info/rfc6949>.

[RFC7669] Levine, J., "Assigning Digital Object Identifiers to RFCs", RFC 7669, DOI 10.17487/RFC7669, October 2015, <http://www.rfc-editor.org/info/rfc7669>.

[RFC7841] Halpern, J., Ed., Daigle, L., Ed., and O. Kolkman, Ed., "RFC Streams, Headers, and Boilerplates", RFC 7841, DOI 10.17487/RFC7841, May 2016, <http://www.rfc-editor.org/info/rfc7841>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <http://www.rfc-editor.org/info/rfc7991>.

IAB Members at the Time of Approval

The IAB members at the time this memo was approved were (in alphabetical order):

Jari Arkko
Ralph Droms
Ted Hardie
Joe Hildebrand
Russ Housley
Lee Howard
Erik Nordmark
Robert Sparks
Andrew Sullivan
Dave Thaler
Martin Thomson
Brian Trammell
Suzanne Woolf

Acknowledgements

Many people contributed valuable ideas to this document. Special thanks go to Robert Sparks for his in-depth review and contributions early in the development of this document and to Julian Reschke for his help getting the document structured more clearly.

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Joe Hildebrand
Mozilla

Email: joe-ietf@cursive.net