Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 11, 2016

P. Hoffman
ICANN
J. Hildebrand
Cisco

September 08, 2015

RFC v3 Prep Tool Description
draft-hoffman-rfcv3-preptool-06

Abstract

This document describes some aspects of the "prep tool" that is expected to be created when the new RFC v3 specification is deployed. This draft is just a way to keep track of the ideas; it is not (currently) expected to be published as an RFC.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on March 11, 2016.

Copyright Notice

Table of Contents

1. Introduction

For the future of the RFC format, the RFC Editor has decided that XML (using the XML2RFCv3 vocabulary [I-D.hoffman-xml2rfc]) is the canonical format, in the sense that it is the data that is blessed by the process as the actual RFC. See [RFC6949] for more detail on this.

Most people will read other formats, such as HTML, PDF, ASCII text, or other formats of the future, however. In order to ensure each of these format is as similar as possible to one another as well as the canonical XML, there is a desire for 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 draft specifies the steps that the prep tool will have to take. As changes to [I-D.hoffman-xml2rfc] are made, this document will be updated.

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

2. v3 Prep Tool Usage Scenarios

The prep tool will have several settings:

- Internet-Draft preparation
- Canonical RFC preparation

There are only a few difference between the two settings. For example, the boilerplate output will be different, as will the date output on the front page.

Note that this 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 not need to change the prep tool in order to create their own formatter: they only need to be able to consume prepared text.

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 v3 XML as an input format, the submission tool runs the submitted file through the prep tool. 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 original 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 AUTH48, the RPC will run the prep tool in canonical RFC preparation mode and make the results available to the authors so they can see what the final output might look like. When the document is done with AUTH48 review, the RPC runs the prep tool in canonical RFC preparation mode one last time, locks down the canonicalized XML, runs the formatters for the publication formats, and publishes all of those. It is probably a good idea for the RPC to keep a copy of the input XML file from the various steps of the RFC production process.

This document assumes that the prep tool will be used in the following manner by the RFC Production Center; they may use something different, or with different configuration.

Similarly to I-D's, 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 is correct. 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.

1. Fully process any DTDs in the input document, then remove the DTD. At a minimum, this entails processing the entityrefs and includes for external files.

2. Process all <x:include> elements. Note: <x:include>d XML may include more <x:include>s (with relative URLs rooted at the xml:base). The tool may be configurable with a limit on the depth of recursion.

3. Run idnits. idnits will indicate if it encountered any errors, and will also provide text with all of the warnings and errors in a human-readable form. The prep tool displays all the warnings and errors, and stops if there was an error.

4. Remove processing instructions.

5. If in RFC production mode, remove comments.

6. Add the [RFC5741] boilerplate text with current values. However, if different boilerplate text already exists in the

input, produce a warning that says that other tools, specifically the draft submission tool, will treat that condition as an error. The application will use the "ipr", "category", "submission", and "consensus" attributes of the <rfc> element to determine which [RFC5741] boilerplate to include, as described in Appendix A of [I-D.hoffman-xml2rfc].

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

8. Fill in the "mode" attribute of <rfc> with a string (to be determined later) indicating the type of prepping that was done (RFC production or I-D mode).

9. If in I-D mode, if there is a <note> element with a "removeInRFC" attribute that has the value "true", add a paragraph to the top of the <note> element that says "This note is to be removed before publishing as an RFC.", if such a paragraph does not yet exist.

10. If in I-D mode, fill in "expiresDate" attribute of <rfc> based on the the <date> element of the document's <front> element, if there is one. Use the same expiry date (if needed) in the boilerplate.

11. Fill in any default values for attributes on elements, except "keepWithNext" and "keepWithPrevious" of <t>, and "toc" of <section>.

12. For any <reference> element that does not already have a "target" attribute, fill that attribute in if the element has one or more <seriesinfo> child element(s). The particular URLs for RFCs and Internet-Drafts for this step will be specified later by the RFC Editor and the IESG. These URLs might also be different from before and after the v3 format is adopted.

13. 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-".

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

    * <section>: pn='s-1.4.2'

    * <abstract>: pn='s-abstract'

    * <note>: pn='s-note-2'

* <boilerplate>: pn='s-boilerplate'

* <table>: pn='t-3'

* <figure>: pn='f-4'

* <artwork>, <aside>, <blockquote>, <dl>, <dt>, <li>, <ol>, <references>, <sourcecode>, <t>, <ul>: pn='p-[section]-[counter]'

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

16. If the "sortRefs" attribute of the <rfc> element is true, sort the <reference>s and <referencegroup>s lexically by the value of the "anchor" attribute, as modified by the "to" attribute of any <displayreference> element.

17. 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 than what was being filled in.

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

    * For format='counter', the "derivedContent" is the section, figure, table, or ordered list number of the element with 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 something else, the "derivedContent" is the title of the thing pointed to, such as "Section 2.3" 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 value of the "target" attribute with no other adornment. Issue a warning if the "derivedContent" attribute already exists and has a different value than what was being filled in.

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

20. For each <relref> element that does not have content, fill the"derivedRemoteContent" based on the content of the target reference.

    * If the target reference is a RFC or Internet-Draft in the v3 format, find the anchor given in the "relative" attribute or derived from the "section" attribute, and use the identifier of that element (such as "Section 2.3" or "Table 4") for the "derivedRemoteContent".

    * If the target reference is not a RFC or Internet-Draft in the v3 format, use the value of the "relative" or "section" attribute for the "derivedRemoteContent".

    * Issue a warning if the "derivedRemoteContent" attribute already exists and has a different value than what was being filled in.

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

22. If an <artwork> element has a "src" attribute with no scheme is specified, treat the scheme as "file:" in a path relative to the file being processed. This will likely be one of the most common authoring approaches.

23. 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. This rule attempts to prevent <artwork src='file:///etc/passwd'> and similar security issues.

24. If an <artwork> element has type='svg' and there is a "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. Add an "originalSrc" attribute with the value of the URI and remove the "src" attribute.

25. If an <artwork> element has type='binary-art', the data needs to be in a "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, add an "originalSrc" attribute with the value of the URI, and remove the "src" attribute. 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). Note: since this feature can't be used for RFCs at the moment, this entire feature might be de-prioritized.

26. If an <artwork> element does not have type='svg' or type='binary-art' and there is a "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.

    * 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. Add an "originalSrc" attribute with the value of the URI and remove the "src" attribute.

27. If a <sourcecode> element has a "src" attribute with no scheme is specified, treat the scheme as "file:" in a path relative to the file being processed.

28. 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. This rule attempts to prevent <sourcecode src='file:///etc/passwd'> and similar security issues.

29. 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 the appropriately-escaped 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 appropriately-escaped resolved text from the URI in the "src" attribute. Add an "originalSrc" attribute with the value of the URI and remove the "src" attribute.

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

31. Ensure that the output has the "version" attribute of <rfc>, and that it is set to "3".

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

33. If in RFC production mode, check if there is a <link> element with an ISSN for the RFC series; if not, add one.

34. If in RFC production mode, check if there is a <link> element with a DOI for this RFC; if not, add one.

35. If in RFC production mode, check if there is a <link> element with the file name of the Internet-Draft that became this RFC; if not, add one.

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

37. Pretty-format the XML output. (Note: tools like https://github.com/hildjj/dentin do an adequate job.)

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

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's starting with "n-") in order to make the resulting file easier to edit.

7. IANA Considerations

None.

8. Security Considerations

None.

9. 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.

10. Informative References

[I-D.hoffman-xml2rfc]
Hoffman, P., "The 'XML2RFC' version 3 Vocabulary", draft-hoffman-xml2rfc-23 (work in progress), September 2015.

[RFC5741] Daigle, L., Ed., Kolkman, O., Ed., and IAB, "RFC Streams, Headers, and Boilerplates", RFC 5741, DOI 10.17487/RFC5741, December 2009, <http://www.rfc-editor.org/info/rfc5741>.

[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>.

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Joe Hildebrand
Cisco

Email: jhildebr@cisco.com