Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 12, 2015

P. Hoffman
VPN Consortium
J. Hildebrand
Cisco

June 10, 2015

RFC v3 Prep Tool Description
draft-hoffman-rfcv3-preptool-03

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 December 12, 2015.

Copyright Notice

Table of Contents

1. Introduction

As a part of the new HTML format work, the RFC Editor has decided that the XML2RFCv3 vocabulary [I-D.hoffman-xml2rfc] will be canonical, 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.

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 non-canonical output 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.

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. If in RFC production mode:

   * Remove comments.

   * Remove processing instructions.

4. 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].

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

6. If in I-D mode, if there is a <note> element with a "removeInRFC" attribute, add a paragraph to the top of the <note> element that says "This note is to be removed before publishing as an RFC".

7. If in I-D mode, if there is a <date> element in the document's <front> element, and that date is earlier than today, it is an error.

8. 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.

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

10. If the <workgroup> content doesn't end with "Group", issue a warning.

11. Sanity-check the value of the "obsoletes" and "updates" attributes in the <rfc> element and give a waring if the sanity check fails.

12. For each <section> element that has a "numbered" attribute set to "false", verify that the section is either followed by a section at the top level, or is not followed by another section that does not contain the "numbered" attribute set to "false"; if both tests fail, it is an error.

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-[counter]'

    * <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 doesn't already have a start.

16. Look for groups of <postalLine> elements; if found, emit them in the order they are seen.

17. Sort the references, if "sortRefs" of <rfc> is true.

18. For each <xref> element, give an error if there is no "target" attribute, or if that attribute is not an anchor in another element.

19. For each <xref> element that has content, fill the "derivedContent" with the element content. Issue a warning if the "derivedContent" attribute already exists and has a different value than what was being filled in.

20. 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. For format='default' and the "target" attribute points to a <reference> or <referencegroup> element, the "derivedContent" is the value of the "target" attribute. 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 the name of the "anchor" attribute of that element with no other adornment. Issue a warning if the "derivedContent" attribute already exists and has a different value than what was being filled in.

21. For each <relref> element, give an error if there is no "target" attribute, or if that attribute is not an anchor in a <reference> or <referencegroup> element.

22. For each <relref> element, if the element has neither a "relative" nor a "section" attribute, give an error.

23. For each <relref> element, if the element has both a "relative" nor a "section" attribute, give an error.

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

25. 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, and the anchor given in the "relative" attribute or derived from the "section" attribute is not in the target refernence, give an error. 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 that attribute already exists and has a different value than what was being filled in. Issue a warning if the "derivedRemoteContent" attribute already exists and has a different value than what was being filled in.

26. For each <relref> element that has content, fill the "derivedRemoteContent" with the element content. Issue a warning if the "derivedRemoteContent" attribute already exists and has a different value than what was being filled in.

27. If an <artwork> element has both a "src" attribute and there is any existing content, give an error.

28. 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.

29. 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.

30. If an <artwork> element has a "src" attribute URI scheme that is not "data:", "file:", "http:", or "https:", give an error.

31. 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 mediatype of the data: URI is not "image/svg+xml", error.

* 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.

32. If an <artwork> element has type='svg', check the SVG schema of the resulting content against the profile in [I-D.brownlee-svg-rfc]. If it does not pass the check, give an error (?) or strip the offending content (?).

33. 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 element has content, give an error.

    * If in RFC production mode, give an error.

    * 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.

34. 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.

35. If a <sourcecode> element has both a "src" attribute and there is any existing content, give an error.

36. 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.

37. 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 <artwork src='file:///etc/passwd'> and similar security issues.

38. If a <sourcecode> element has a "src" attribute URI scheme that is not "data:", "file:", "http:", or "https:", give an error.

39. 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.

40. If an <sourcecode> element has a "type" attribute that matches one of the known types, and that type has code that can validate the code, do the validation (noting that elements that have the same "name" attribute should be collected) and give a warning if the validation fails.

41. Add a <link> child element to <rfc> for the DOI, if in RFC production mode.

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

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

44. Prepare the index, if one is called for in the document. The steps for doing that is out of scope for this document.

45. If in I-D mode, check if there is a <link> element with an ISSN for the RFC series; if so, delete it.

46. If in I-D mode, check if there is a <link> element with a DOI for this draft; if so, delete it.

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

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

49. 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.

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

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

52. 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.brownlee-svg-rfc]
Brownlee, N., "SVG Drawings for RFCs: SVG 1.2 RFC", draft-brownlee-svg-rfc-09 (work in progress), March 2015.

[I-D.hoffman-xml2rfc]
Hoffman, P., "The 'XML2RFC' version 3 Vocabulary", draft-hoffman-xml2rfc-18 (work in progress), May 2015.

[RFC5741] Daigle, L., Kolkman, O., and IAB, "RFC Streams, Headers, and Boilerplates", RFC 5741, December 2009.

[RFC6949] Flanagan, H. and N. Brownlee, "RFC Series Format Requirements and Future Development", RFC 6949, May 2013.

Authors' Addresses

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org

Joe Hildebrand
Cisco

Email: jhildebr@cisco.com