Workgroup: Network Working Group
Published: 30 January 2023
Intended Status: Informational
Expires: 3 August 2023
Authors: M. Petit-Huguenin
Impedance Mismatch LLC

Mappings Between XML2RFC v3 and AsciiDoc

Abstract

This document specifies a mapping between XML2RFC v3 and AsciiDoc. The goal of this mapping and its associated tooling is to make writing an Internet-Draft as simple as possible, by converting any AsciiDoc formatted document into a valid Internet-Draft, ready to be submitted to the IETF. This is still work in progress and for the time being this mapping only ensures that any valid XML2RFC element can be generated from AsciiDoc.

Status of This Memo

This Internet-Draft will expire on 3 August 2023.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a mapping between XML2RFC v3, as defined in the successor [I-D.irse-draft-irse-xml2rfcv3-implemented-03] of [RFC7991], and [AsciiDoc].

The goal of this mapping and its associated tooling is to make writing an Internet-Draft as simple as possible, by converting any AsciiDoc formatted document into a valid Internet-Draft, ready to be submitted to the IETF. This is still work in progress and for the time being this mapping only ensures that any valid XML2RFC element can be generated from AsciiDoc.

Installation and usage of the tool is described in Appendix A.

2. Overview

AsciiDoc is an extensible documentation format whose syntax can be split in two parts:

*The standard syntax as defined in https://docs.asciidoctor.org/asciidoc/latest/

*Extensions that extend the standard syntax. A popular example of extension is asciidoctor-diagrams, which permits the insertion of diagrams in a document.

The mapping described in this document splits these parts further. The standard syntax is split in two subparts:

*The primitive standard syntax contains all the AsciiDoc elements that have a direct translation back and forth with XML2RFC v3.

*The compound standard syntax contains all the other standard elements.

Elements in the compound standard syntax are converted into XML2RFC v3 by using multiple elements. That means that the conversion back to AsciiDoc will always be to elements in the primitive standard syntax, not the original elements.

For instance an AsciiDoc unordered list is a primitive element that is converted to an <ol> element in XML2RFC. On the other hand the AsciiDoc checklist is a compound element that is converted into a combination of <ol> and <t> elements in XML2RFC, but the conversion back will not result in an AsciiDoc checklist.

Similarly some AsciiDoc extensions have been developed to support XML2RFC elements and attributes that do not have a standard AsciiDoc equivalent. The XML2RFC elements and attributes will be converted back into these AsciiDoc extensions, but any other AsciiDoc extensions will be converted back into a combination of primitive AsciiDoc elements.

When choosing the mapping between an XML2RFC elements and an AsciiDoc element, reusing as much as possible of the standard AsciiDoc syntax was a priority. An extension is added only if a standard element with the correct semantics cannot be found.

AsciiDoc is not really meant to support highly structured XML2RFC elements like <front>, <reference>, <referencegroup>, <author>, and <contact>. Thus these element are used directly in AsciiDoc by inserting an XML2RFC fragment inside a passthrough block. Passthrough blocks serve also as an escape hatch for new elements added to the XML2RFC syntax until a new revision of this document is published.

2.1. Deprecated and Unsupported Elements

In any case AsciiDoc, both standard and extensions, never generate deprecated XML2RFC elements or attributes.

This mapping is specifically designed to write Internet-Drafts so there is no support, in any direction, for XML2RFC elements and attributes that are specific to RFCs.

The list of xml2rc v3 elements that are specific to the generation of RFCs, and so cannot be generated from an AsciiDoc document are:

- the <boilerplate> element

- the <link> element

- the "iprExtract", "number", "prepTime", and "seriesNo" attributes in the <rfc> element

- the <toc> element

- all the attributes added by the preptool.

2.2. Docinfo Processing

If a docinfo file is used, it completely replaces the generation of the <front> element. To reduce redundancy, some of the attributes declared in the document header can be referenced in the docinfo file:

subtitle: This attribute contains the subtitle to be inserted in the "abbrev" attribute of the <title> element. This is an extension to the standard AsciiDoc.

author: This attribute contains the full name to be inserted in the "fullname" attribute of the <author> element.

email: This attribute contains the email to be inserted in the <email> element.

revday: This attribute contains the day to be inserted in the "day" attribute of the <date> element. This is an extension to the standard AsciiDoc.

revmonth: This attribute contains the month to be inserted in the "month" attribute of the <date> element. This is an extension to the standard AsciiDoc.

revyear: This attribute contains the year to be inserted in the "year" attribute of the <date> element. This is an extension to the standard AsciiDoc.

abstract: This attribute contains the abstract to be inserted in the <abstract> element. This is an extension to the standard AsciiDoc.

revremark: This attribute contains the note to be inserted in a <note> element.

2.3. Include Citation

References can be automatically extracted from local Internet-Drafts written in AsciiDoc, and inserted into a reference section by including the document with the fragment identifier "#citation", as in the following example:

```
- [[[I-D.petithuguenin-mapping-xml2rfc-asciidoc-00]]]
++++
include::test.adoc#citation[]
++++
```

3. Mappings

4. Mapping between AsciiDoc and XML2RFC v3

This section describes the mapping between the AsciiDoc direct syntax and IETF extensions to non-deprecated XML2RFC v3 elements and attributes.

4.1. <abstract>

The <abstract> element inside a <front> element is either generated from an abstract section or copied from a docinfo file.

The <abstract> element can also be used inside a <reference> element in a passthrough block in the bibliography.

4.1.1. "anchor" Attribute

The <abstract> "anchor" attribute can be copied from a docinfo file. It cannot be generated from an abstract section.

4.2. <address>

The <address> element inside a <front> element is copied from a docinfo file.

The <address> element can also be used inside a <reference> element in a passthrough block in the bibliography.

The <address> element can also be used inside an <author> element in a passthrough block in a section.

4.3. <annotation>

The <annotation> element can be used inside a <reference> element in a passthrough block in the bibliography.

4.4. <area>

The <area> element inside a <front> element is copied from a docinfo file.

The <area> element can be used inside a <reference> element in a passthrough block in the bibliography.

4.5. <artset>

The <artset> element is generated from an example block using the style "alt". This block can only contain images or literal blocks.

The alt style on an example block is an extension to standard AsciiDoc.

The following example

```
[alt]
====
image::test.jpg[]

....
ASCII Art
....
====
```

is converted to

```
<artset>
<artwork src='test.jpg' />
<artwork><![CDATA[ASCII Art]]></artwork>
</artset>
```

4.5.1. "anchor" Attribute

The <artset> "anchor" attribute is generated from the "alt" block ID, if present.

The following example

```
[alt#id1]
====
image::test.jpg[]

....
ASCII Art
....
====
```

is converted to

```
<artset anchor='id1'>
<artwork src='test.jpg' />
<artwork><![CDATA[ASCII Art]]></artwork>
</artset>
```

4.6. <artwork>

The <artwork> element is generated from a literal or image block.

The following example

```
....
ASCII Art
....
```

is converted to

```
<artwork><![CDATA[ASCII Art]]></artwork>
```

The following example

```
image::test.jpg[]
```

is converted to

```
<artwork src='test.jpg' />
```

4.6.1. "align" Attribute

The <artwork> "align" attribute is generated from the align attribute, if present.

The following example

```
[align=center]
....
ASCII Art
....
```

is converted to

```
<artwork align='center'><![CDATA[ASCII Art]]></artwork>
```

The following example

```
image::test.jpg[align=center]
```

is converted to

```
<artwork align='center' src='test.jpg' />
```

4.6.2. "alt" Attribute

The <artwork> "alt" attribute is generated from the alt attribute, if present.

The alt attribute on a literal block is an extension to standard AsciiDoc.

The following example

```
[alt="Alternative"]
....
Some text
....
```

is converted to

```
<artwork alt='Alternative' name=''><![CDATA[Some text]]></artwork>
```

The following example

```
image::test.jpg[Some text]
```

is converted to

```
<artwork alt='Some text' src='test.jpg' />
```

4.6.3. "anchor" Attribute

The <artwork> "anchor" attribute is generated from the image or literal block ID, if present.

The following example

```
[[id2]]
....
ASCII Art
....
```

is converted to

```
<artwork anchor='id2'><![CDATA[ASCII Art]]></artwork>
```

The following example

```
[[id3]]
image::test.jpg[]
```

is converted to

```
<artwork anchor='id3' src='test.jpg' />
```

4.6.4. "name" Attribute

The <artwork> "name" attribute is generated from the name attribute, if present.

The name attribute on a literal block is an extension to standard AsciiDoc.

The following example

```
[name="test.txt"]
....
ASCII Art
....
```

is converted to

```
<artwork><![CDATA[ASCII Art]]></artwork>
```

4.6.5. "src" Attribute

The <artwork> "src" attribute is generated from the target of an image block.

The src attribute on an image block is an extension to standard AsciiDoc.

The following example

```
image::test.jpg[]
```

is converted to

```
<artwork src='test.jpg' />
```

4.6.6. "type" Attribute

The <artwork> "type" attribute is generated from the format attribute, if present.

The format attribute on an literal block is an extension to standard AsciiDoc.

The following example

```
[format="ascii-art"]
....
ASCII Art
....
```

is converted to

```
<artwork type='ascii-art'><![CDATA[ASCII Art]]></artwork>
```

4.7. <aside>

The <aside> element is generated from a sidebar block.

The following example

```
****
Some text
****
```

is converted to

```
<aside>
<t>
Some text
</t>
</aside>
```

4.7.1. "anchor" Attribute

The <aside> "anchor" attribute is generated from the aside block ID, if present.

The following example

```
[[id4]]
****
Some text
****
```

is converted to

```
<aside anchor='id4'>
<t>
Some text
</t>
</aside>
```

4.8. <author>

The <author> element inside a <front> element is either generated from the author information in the document header or copied from a docinfo file.

The <author> element can also be used inside a <reference> element in a passthrough block in the bibliography.

4.9. <back>

The <back> element is implicitly generated when a bibliography or a appendices are added to a document.

4.10. <bcp14>

The <bcp14> element is generated from the text span style bcp14.

The text span style bcp14 is an extension to standard AsciiDoc.

The following example

```
Implementers [bcp14]#MUST# implement all MUST.
```

is converted to

```
<t>
Implementers <bcp14>MUST</bcp14> implement all MUST.
</t>
```

4.11. <blockquote>

The <blockquote> element is generated from a quote block.

The following example

```
____
Some text
____
```

is converted to

```
<blockquote>
<t>
Some text
</t>
</blockquote>
```

4.11.1. "anchor" Attribute

The <blockquote> "anchor" attribute is generated from the quote block ID, if present.

The following example

```
[[id5]]
____
Some text
____
```

is converted to

```
<blockquote anchor='id5'>
<t>
Some text
</t>
</blockquote>
```

4.11.2. "cite" Attribute

The <blockquote> "cite" attribute is generated from the quote block cite attribute, if present.

The cite attribute on an literal block is an extension to standard AsciiDoc.

The following example

```
[quote,cite=https://link]
____
Some text
____
```

is converted to

```
<blockquote cite='http://link'>
<t>
Some text
</t>
</blockquote>
```

4.11.3. "quotedFrom" Attribute

The <blockquote> "quotedFrom" attribute is generated from the quote block positional attribute, if present.

The following example

```
[quote,attribution]
____
Some text
____
```

is converted to

```
<blockquote quotedFrom='attribution'>
<t>
Some text
</t>
</blockquote>
```

4.12. <boilerplate>

The <boilerplate> element is used only for RFCs and so is not generated.

4.13. <br>

The <br> element is generated from the " +" string at the end of a line in a paragraph.

The following example

```
This is +
two lines
```

is converted to

```
<t>
This is<br />
two lines
</t>
```

4.14. <contact>

The <contact> element and its attributes are provided in a section as a passthrough block containing that element, or in a paragraph as an inline passthrough containing that element.

4.15. <country>

The <country> element and its attribute inside a <front> element is copied from a docinfo file.

The <country> element can be used inside a <reference> element in a passthrough block in the bibliography.

4.16. <cref>

The <cref> element is generated from the comment inline macro.

The comment inline macro and its attributes are extensions to standard AsciiDoc.

The following example

```
The
comment:[Replace native]
native mode can be used.
```

is converted to

```
<t>
The
<cref>Replace native</cref>
native mode can be used.
</t>
```

4.16.1. "anchor" Attribute

The <cref> "anchor" attribute is generated from the comment inline macro ID, if present.

The following example

```
The
comment:[Replace native,id=id6]
native mode can be used.
```

is converted to

```
<t>
The
<cref anchor='id6'>Replace native</cref>
native mode can be used.
</t>
```

4.16.2. "display" Attribute

The <cref> "display" attribute is generated from the comment inline macro display attribute, if present.

The following example

```
The
comment:[Replace native,display=true]
native mode can be used.
```

is converted to

```
<t>
The
<cref display='true'>Replace native</cref>
native mode can be used.
</t>
```

4.16.3. "source" Attribute

The <cref> "source" attribute is generated from the comment inline macro source attribute, if present.

The following example

```
The
comment:[Replace native,source=MPH]
native mode can be used.
```

is converted to

```
<t>
The
<cref source='MPH'>Replace native</cref>
native mode can be used.
</t>
```

4.17. <date>

The <date> element and its attributes inside a <front> element are either generated from an revdate document header or copied from a docinfo file.

The <date> element can also be used inside a <reference> element in a passthrough block in the bibliography.

4.18. <dd>

The <dd> element is generated from the definition part of a description list.

The following example

```
text:: data
```

is converted to

```
<dl>
<dt>text</dt><dd>data</dd>
</dl>
```

4.18.1. "anchor" Attribute

The <dd> anchor attribute cannot be generated.

4.19. <displayreference>

The <displayreference> element is generated from a bibliography entry bibref that differs from the ID of the entry itself.

The following example

```
[bibliography]
== References

* [[[id8]]]
+
[[id9]]
++++
<reference anchor="id9">
</reference>
++++
```

is converted to

```
<back>

<displayreference target="id9" to="id8" />


<references>
<name>References</name>
<reference anchor="id9">
</reference>
</references>
</back>
```

4.19.1. "target" Attribute

The <displayreference> target attribute cannot be generated.

4.19.2. "to" Attribute

The displayreference "to" attribute cannot be generated.

4.20. <dl>

The <dl> element is generated from a description list.

The following example

```
text:: data
```

is converted to

```
<dl>
<dt>text</dt><dd>data</dd>
</dl>
```

4.20.1. "anchor" Attribute

The <dl> "anchor" attribute is generated from the ID of a description list, if present.

The following example

```
[[id10]]
text:: data
```

is converted to

```
<dl anchor='id10'>
<dt>text</dt><dd>data</dd>
</dl>
```

4.20.2. "indent" Attribute

The <dl> "indent" attribute is generated from the indent attribute of a description list, if present.

The description list indent attribute is an extension to standard AsciiDoc.

The following example

```
[indent=3]
text:: data
```

is converted to

```
<dl indent='3'>
<dt>text</dt><dd>data</dd>
</dl>
```

4.20.3. "newline" Attribute

The <dl> "newline" attribute is generated from the indent attribute of a description list, if present.

The description list newline attribute is an extension to standard AsciiDoc.

The following example

```
[newline=true]
text:: data
```

is converted to

```
<dl newline='true'>
<dt>text</dt><dd>data</dd>
</dl>
```

4.20.4. "spacing" Attribute

The <dl> "spacing" attribute is generated from the spacing attribute of a description list, if present.

The description list spacing attribute is an extension to standard AsciiDoc.

The following example

```
[spacing=compact]
text:: data
```

is converted to

```
<dl spacing='contact'>
<dt>text</dt><dd>data</dd>
</dl>
```

4.21. <dt>

The <dt> element is generated from the term part of a definition list.

The following example

```
text:: data
```

is converted to

```
<dl>
<dt>text</dt><dd>data</dd>
</dl>
```

4.21.1. "anchor" Attribute

The <dt> anchor attribute cannot be generated.

4.22. <em>

The <em> element is generated from an italic text formatting.

The following example

```
Text with __emphasis__.
```

is converted to

```
<t>
Text with <em>emphasis</em>.
</t>
```

4.23. <email>

The <email> element inside a <front> element is either generated from the email document header or copied from a docinfo file.

The <email> element can also be used inside a <reference> element in a passthrough block in the bibliography.

The <email> element can also be used inside an <author> element in a passthrough block in a section.

4.24. <eref>

The <eref> element is generated from a link macro.

The following example

```
As cited in link:https://test[].
```

is converted to

```
<t>
As cited in <eref target='https://test' />.
</t>
```

4.24.1. "bracket" Attribute

The <eref> "bracket" attribute is generated from the bracket attribute in the link macro, if present.

The link macro bracket attribute is an extension to standard AsciiDoc.

The following example

```
As cited in link:https://test[brackets=true].
```

is converted to

```
<t>
As cited in <eref brackets='true' target='https://test' />.
</t>
```

4.24.2. "target" Attribute

The <eref> "target" attribute is generated from the target in the link macro.

The following example

```
As cited in link:https://test[].
```

is converted to

```
<t>
As cited in <eref target='https://test' />.
</t>
```

4.25. <figure>

The <figure> element is generated from an example block with the style "figure".

The figure style on an example block is an extension to standard AsciiDoc.

The following example

```
[figure]
====
image::test.jpg[]
====
```

is converted to

```
<figure>

<artwork src='test.jpg' />
</figure>
```

4.25.1. "anchor" Attribute

The <figure> "anchor" attribute is generated from the ID of a figure block, if present.

```
[[id12]]
[figure]
====
image::test.jpg[]
====
```

is converted to

```
<figure anchor='id12'>

<artwork src='test.jpg' />
</figure>
```

4.26. <front>

The <front> element is either generated from the author information in the document header or copied from a docinfo file.

The <front> element can also be used inside a <reference> element in a passthrough block in the bibliography.

4.27. <iref>

The <iref> element is generated either from an indexterm inline macro or from an indexterm block macro extension.

The indexterm block macro is an extension to standard AsciiDoc.

The following example

```
Text with an (((test))) index.
```

is converted to

```
<t>
Text with an <iref item='test' /> index.
</t>
```

The following example

```
indexterm::[test]
```

is converted to

```
<iref item='test'/>
```

4.27.1. "item" Attribute

The <iref> "item" attribute is generated from the positional attribute of the index term.

The following example

```
Text with an (((test))) index.
```

is converted to

```
<t>
Text with an <iref item='test' /> index.
</t>
```

4.27.2. "primary" Attribute

The <iref> "primary" attribute is generated from the primary attribute of the index term.

The primary indexterm block macro attribute is an extension to standard AsciiDoc.

The following example

```
Text with an (((test,,primary=true))) index.
```

is converted to

```
<t>
Text with an <iref item='test' subitem='' /> index.
</t>
```

The following example

```
indexterm::[test,primary=true]
```

is converted to

```
<iref item='test' primary='true'/>
```

4.27.3. "subitem" Attribute

The <iref> "subitem" attribute is generated from the secondary attribute of the index term.

The secondary indexterm block macro attribute is an extension to standard AsciiDoc.

The following example

```
Text with an (((test,test2))) index.
```

is converted to

```
<t>
Text with an <iref item='test' subitem='test2' /> index.
</t>
```

The following example

```
indexterm::[test,test2]
```

is converted to

```
<iref item='test' subitem='test2'/>
```

4.28. <keyword>

The <keyword> element inside a <front> element is copied from a docinfo file.

The <keyword> element can also be used inside a <reference> element in a passthrough block in the bibliography.

4.29. <li>

The <li> element is generated from a line in either an unordered or an ordered list.

The following example

```
. One line
```

is converted to

```
<ol>
<li>One line</li>
</ol>
```

The following example

```
* One line
```

is converted to

```
<ul>
<li>One line</li>
</ul>
```

4.29.1. "anchor" Attribute

The <li> "anchor" attribute is generated from the ID of a list item, if present.

The following example

```
. [[id13]]test
```

is converted to

```
<ol>
<li anchor='id13'>test</li>
</ol>
```

The following example

```
* [[id14]]test
```

is converted to

```
<ul>
<li anchor='id14'>test</li>
</ul>
```

4.30. <link>

The <link> element is provided in the docinfo file.

4.31. <middle>

The <middle> element is implicitly generated.

4.32. <name>

The <name> element is generated from the title in either a section, an example block with figure style, or a table block.

The possibility of using formatted text in an example block with figure style title or in a table title is an extension to standard AsciiDoc.

The following example

```
== section
```

is converted to

```
<section numbered='false'>
<name>section</name>

</section>
```

The following example

```
.title
[figure]
====
....
ASCII Art.
....
====
```

is converted to

```
<figure>
<name>title</name>
<artwork><![CDATA[ASCII Art]]></artwork>
</figure>
```

The following example

```
.title
|===
| Test
|===
```

is converted to

```
<table>
<name>title</name>
<tbody><tr><td align='left'>Test</td></tr></tbody>
</table>
```

4.33. <note>

The <note> element inside a <front> element is either generated from the revremark document header or copied from a docinfo file.

The <note> element can also be used inside a <reference> element in a passthrough block in the bibliography.

4.34. <ol>

The <ol> element is generated from an ordered list.

The following example

```
. One line
```

is converted to

```
<ol>
<li>One line</li>
</ol>
```

4.34.1. "anchor" Attribute

The <ol> "anchor" attribute is generated from the ID of a ordered list, if present.

The following example

```
[[id15]]
. test
```

is converted to

```
<ol anchor='id15'>
<li>test</li>
</ol>
```

4.34.2. "group" Attribute

The <ol> "group" attribute is generated from the group attribute of a ordered list, if present.

The group attribute on an ordered list is an extension to standard AsciiDoc.

The following example

```
[group=2]
. test
```

is converted to

```
<ol group='2'>
<li>test</li>
</ol>
```

4.34.3. "indent" Attribute

The <ol> "indent" attribute is generated from the indent attribute of an ordered list, if present.

The indent attribute on an ordered list is an extension to standard AsciiDoc.

The following example

```
[indent=2]
. test
```

is converted to

```
<ol indent='2'>
<li>test</li>
</ol>
```

4.34.4. "spacing" Attribute

The <ol> "spacing" attribute is generated from the spacing attribute of an ordered list, if present.

The spacing attribute on an ordered list is an extension to standard AsciiDoc.

The following example

```
[spacing=true]
. test
```

is converted to

```
<ol spacing='true'>
<li>test</li>
</ol>
```

4.34.5. "start" Attribute

The <ol> "start" attribute is generated from the start attribute of an ordered list, if present.

The following example

```
[start=0]
. test
```

is converted to

```
<ol start='0'>
<li>test</li>
</ol>
```

4.34.6. "type" Attribute

The <ol> "type" attribute is generated from the style of an ordered list, if present.

The prefix and suffix types are used to generate a "type" attribute that uses the "percent-letter" format. The prefix and suffix attributes on an ordered list are extensions to standard AsciiDoc.

The following example

```
[lowerroman]
. test
```

is converted to

```
<ol type='i'>
<li>test</li>
</ol>
```

The following example

```
[arabic,prefix="[REQ",suffix="]"]
. test
```

is converted to

```
<ol type='[REQ%d]'>
<li>test</li>
</ol>
```

4.35. <organization>

The <organization> element inside a <front> element is copied from a docinfo file.

The <organization> element can also be used inside a <reference> element in a passthrough block in the bibliography.

The <organization> element can also be used inside an <author> element in a passthrough block in a section.

4.36. <phone>

The <phone> element inside a <front> element is copied from a docinfo file.

The <phone> element can also be used inside a <reference> element in a passthrough block in the bibliography.

The <phone> element can also be used inside an <author> element in a passthrough block in a section.

4.37. <postal>

The <postal> element inside a <front> element is copied from a docinfo file.

The <postal> element can also be used inside a <reference> element in a passthrough block in the bibliography.

The <postal> element can also be used inside an <author> element in a passthrough block in a section.

4.38. <postalLine>

The <postalLine> element inside a <front> element is copied from a docinfo file.

The <postalLine> element can also be used inside a <reference> element in a passthrough block in the bibliography.

The <postalLine> element can also be used inside an <author> element in a passthrough block in a section.

4.39. <refcontent>

The <refcontent> element can be used inside a <reference> element in a passthrough block in the bibliography.

4.40. <reference>

The <reference> element is provided in a passthrough block in the bibliography.

4.41. <referencegroup>

The <referencegroup> element is provided in a passthrough block in the bibliography.

4.42. <references>

The <references> element is generated from an unordered list in a bibliography section.

The following example

```
[bibliography]
== References

* [[[id16]]]
+
++++
<reference anchor="id16">
</reference>
++++
```

is converted to

```
<back>

<references>
<name>References</name>
<reference anchor="id16">
</reference>
</references>
</back>
```

4.42.1. "anchor" Attribute

The <references> "anchor" attribute is generated from the ID of an unordered list in a bibliographic section, if present.

The following example

```
[[id17]]
[bibliography]
== References
```

is converted to

```
<back>

<references anchor='id17'>
</references>
</back>
```

4.43. <rfc>

The <rfc> element is generated by a top level section:

The following example

```
= Title
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
tocInclude='false' version='3'>
```

4.43.1. "category" Attribute

The <rfc> "category" attribute is generated from the category document header attribute.

The category attribute on a document header is an extension to standard AsciiDoc.

The following example

```
= Title
:category: std
```

is converted to

```
<rfc category='std' docName='test' ipr='trust200902'
tocInclude='false' version='3'>
```

4.43.2. "consensus" Attribute

The <rfc> "consensus" attribute is generated from the consensus document header attribute.

The consensus attribute on a document header is an extension to standard AsciiDoc.

The following example

```
= Title
:consensus: true
```

is converted to

```
<rfc category='info' consensus='true' docName='test'
ipr='trust200902' tocInclude='false' version='3'>
```

4.43.3. "docName" Attribute

The <rfc> "docName" attribute is generated from the doc-name document header attribute.

The doc-name attribute on a document header is an extension to standard AsciiDoc.

The following example

```
= Title
:doc-name: draft-petithuguenin-ex-00
```

is converted to

```
<rfc category='info' docName='draft-petithuguenin-ex-00'
ipr='trust200902' tocInclude='false' version='3'>
```

4.43.4. "indexInclude" Attribute

The <rfc> "indexInclude" attribute is generated from the "index-include" document header attribute if present.

The following example

```
= Title
:index-include: false
```

is converted to

```
<rfc category='info' docName='test' indexInclude='false'
ipr='trust200902' tocInclude='false' version='3'>
```

4.43.5. "ipr" Attribute

The <rfc> "ipr" attribute is generated from the "ipr" document header attribute if present, or with the "trust200902" content if not present.

The following example

```
= Title
:ipr: noModificationTrust200902
```

is converted to

```
<rfc category='info' docName='test'
ipr='noModificationTrust200902'
tocInclude='false' version='3'>
```

4.43.6. "iprExtract" Attribute

Specific to RFC, so not implemented.

4.43.7. "number" Attribute

Specific to RFC, so not implemented.

4.43.8. "obsoletes" Attribute

The <rfc> "obsoletes" attribute is generated from the "obsoletes" document header attribute if present.

The following example

```
= Title
:obsoletes: 3261,3262,3263,3264
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
obsoletes='3261,3262,3263,3264'
tocInclude='false' version='3'>
```

4.43.9. "prepTime" Attribute

Specific to RFC, so not implemented.

4.43.10. "seriesNo" Attribute

Specific to RFC, so not implemented.

4.43.11. "sortRefs" Attribute

The <rfc> "sortRefs" attribute is generated from the "sort-refs" document header attribute if present.

The following example

```
= Title
:sort-refs: true
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
sortRefs='true'
tocInclude='false' version='3'>
```

4.43.12. "submissionType" Attribute

The <rfc> "submissionType" attribute is generated from the "submission-type" document header attribute if present.

The following example

```
= Title
:submission-type: independent
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
submissionType='independent'
tocInclude='false' version='3'>
```

4.43.13. "symRefs" Attribute

The <rfc> "symRefs" attribute is generated from the "sym-refs" document header attribute if present.

The following example

```
= Title
:sym-refs: false
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
symRefs='false'
tocInclude='false' version='3'>
```

4.43.14. "tocDepth" Attribute

The <rfc> "tocDepth" attribute is generated from the "toclevels" document header attribute if present.

The following example

```
= Title
:toclevels: 4
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
tocDepth='4'
tocInclude='false' version='3'>
```

4.43.15. "tocInclude" Attribute

The <rfc> "tocInclude" attribute with a value of "false" is generated from the "toc" document header attribute if not present, or a value of "true" if present.

The following example

```
= Title
:toc:
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
tocInclude='true' version='3'>
```

4.43.16. "updates" Attribute

The <rfc> "updates" attribute is generated from the "updates" document header attribute if present.

The following example

```
= Title
:updates: 3261,3262,3263,3264
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
updates='3261,3262,3263,3264'
tocInclude='false' version='3'>
```

4.43.17. "version" Attribute

The <rfc> "version" attribute is implicitly generated.

The following example

```
= Title
```

is converted to

```
<rfc category='info' docName='test' ipr='trust200902'
tocInclude='false' version='3'>
```

4.44. <section>

The <section> attribute is generated from a document section.

The following example

```
== Section 1
```

is converted to

```
<section numbered='false'>
<name>Section 1</name>

</section>
```

4.44.1. "anchor" Attribute

The <section> "anchor" attribute is generated from section ID, if present.

The following example

```
[[id18]]
== Section 1
```

is converted to

```
<section anchor='id18' numbered='false'>
<name>Section 1</name>

</section>
```

4.44.2. "numbered" Attribute

The <section> "numbered" attribute is generated from the current value of the sectnums attribute. Note that AsciiDoc standards sections are unnumbered by default, but xml2rfc sections are numbered by default, so a ":sectnums:" attribute should be added to the document header.

Each AsciiDoc special sections with style "appendix" that needs to be unnumbered needs to be preceded with the attribute ":sectnums!".

The following example

```
:sectnums!:

== Section 1
```

is converted to

```
<section numbered='false'>
<name>Section 1</name>

</section>
```

4.44.3. "removeInRfc" Attribute

The <section> "removeInRfc" attribute is generated from the remove-in-rfc section attribute.

The remove-in-rfc attribute on a section is an extension to standard AsciiDoc.

The following example

```
[remove-in-rfc=true]
== Section 1
```

is converted to

```
<section numbered='false' removeInRFC='true'>
<name>Section 1</name>

</section>
```

4.44.4. "toc" Attribute

The <section> "toc" attribute is generated from the section attribute "toc".

The toc attribute on a section is an extension to standard AsciiDoc.

The following example

```
[toc=exclude]
== Section 1
```

is converted to

```
<section numbered='false' toc='exclude'>
<name>Section 1</name>

</section>
```

4.45. <seriesInfo>

The <seriesInfo> element inside a <front> element is copied from a docinfo file.

The <seriesInfo> element can also be used inside a <reference> element in a passthrough block in the bibliography.

4.46. <sourcecode>

The <sourcecode> element is generated from a listing block with the source style.

The following example

```
[source]
----
class Test {}
----
```

is converted to

```
<sourcecode><![CDATA[class Test {}]]></sourcecode>
```

4.46.1. "anchor" Attribute

The <sourcecode> "anchor" attribute is generated from the ID of a listing block with source style, if present.

The following example

```
[[id19]]
[source]
----
class Test {}
----
```

is converted to

```
<sourcecode anchor='id19'><![CDATA[class Test {}]]></sourcecode>
```

4.46.2. "markers" Attribute

The <sourcecode> "markers" attribute is generated from the markers attribute of a listing block with source style, if present.

The markers attribute on a listing block with source style is an extension to standard AsciiDoc.

The following example

```
[source,markers=true]
----
class Test {}
----
```

is converted to

```
<sourcecode markers='true'><![CDATA[class Test {}]]></sourcecode>
```

4.46.3. "name" Attribute

The <sourcecode> "name" attribute is generated from the name attribute of a listing block with source style, if present.

The name attribute on a listing block with source style is an extension to standard AsciiDoc.

The following example

```
[source,name=Test.java]
----

class Test {}

----
```

is converted to

```
<sourcecode name='Test.java'><![CDATA[
class Test {}
]]></sourcecode>
```

4.46.4. "src" Attribute

The <sourcecode> "src" attribute is generated from the src attribute of a listing block with source style, if present.

The src attribute on a listing block with source style is an extension to standard AsciiDoc.

The following example

```
[source,src=Test.java]
----
----
```

is converted to

```
<sourcecode src='Test.java' />
```

4.46.5. "type" Attribute

The <sourcecode> "type" attribute is generated from the positional attribute of a listing block with source style, if present.

The following example

```
[source,java]
----
class Test {}
----
```

is converted to

```
<sourcecode type='java'><![CDATA[class Test {}]]></sourcecode>
```

4.47. <stream>

The <stream> element inside a <front> element is copied from a docinfo file.

The <stream> element can also be used inside a <reference> element in a passthrough block in the bibliography.

4.48. <strong>

The <strong> element is generated from a bold text formatting.

The following example

```
**Strong** text.
```

is converted to

```
<t>
<strong>Strong</strong> text.
</t>
```

4.49. <sub>

The <sub> element is generated from a subscript text formatting.

The following example

```
Base~16~.
```

is converted to

```
<t>
Base<sub>16</sub>.
</t>
```

4.50. <sup>

The <sup> element is generated from a superscript text formatting.

The following example

```
E = mc^2^
```

is converted to

```
<t>
E = mc<sup>2</sup>
</t>
```

4.51. <t>

The <t> element is generated from a paragraph.

The following example

```
Some text.
```

is converted to

```
<t>
Some text.
</t>
```

4.51.1. "anchor" Attribute

The <t> "anchor" attribute is generated from the ID of a paragraph, if present.

The following example

```
[[id20]]
Some text.
```

is converted to

```
<t anchor='id20'>
Some text.
</t>
```

4.51.2. "indent" Attribute

The <t> "indent" attribute is generated from the indent attribute of a paragraph, if present.

The indent attribute on a paragraph is an extension to standard AsciiDoc.

The following example

```
[ident=3]
Some text.
```

is converted to

```
<t indent='3'>
Some text.
</t>
```

4.51.3. "keepWithNext" Attribute

The <t> "keepWithNext" attribute is generated from the keep-with-next attribute of a paragraph, if present.

The keep-with-next attribute on a paragraph is an extension to standard AsciiDoc.

The following example

```
[keep-with-next=true]
Some text.
```

is converted to

```
<t keepWithNext='true'>
Some text.
</t>
```

4.51.4. "keepWithPrevious" Attribute

The <t> "keepWithPrevious" attribute is generated from the keep-with-previous attribute of a paragraph, if present.

The keep-with-previous attribute on a paragraph is an extension to standard AsciiDoc.

The following example

```
[keep-with-previous=true]
Some text.
```

is converted to

```
<t keepWithPrevious='true'>
Some text.
</t>
```

4.52. <table>

The <table> element is generated from a table.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.52.1. "align" Attribute

The <table> "align" attribute is generated from the ID of a table, if present.

The following example

```
[align=left]
|===
| Cell
|===
```

is converted to

```
<table align='left'>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.52.2. "anchor" Attribute

The <table> "anchor" attribute is generated from the ID of a table, if present.

The following example

```
[[id21]]
|===
| Cell
|===
```

is converted to

```
<table anchor='id21'>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.53. <tbody>

The <tbody> element is generated from a table with rows.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.53.1. "anchor" Attribute

The <tbody> anchor attribute cannot be generated.

4.54. <td>

The <td> element is generated from a table with at least one cell.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.54.1. "align" Attribute

The <td> "align" attribute is generated from the horizontal alignment operator of a cell in a table, if present.

The following example

```
|===
|> Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>&gt; Cell</td></tr></tbody>
</table>
```

4.54.2. "anchor" Attribute

The <td> anchor attribute cannot be generated.

4.54.3. "colspan" Attribute

The <td> "colspan" attribute is generated from the horizontal alignment operator of a cell in a table, if present.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.54.4. "rowspan" Attribute

The <td> "rowspan" attribute is generated from the horizontal alignment operator of a cell in a table, if present.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.55. <tfoot>

The <tfoot> element is generated from a table with footer.

The following example

```
[%footer]
|===

| Cell

| Footer
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
<tfoot><tr><td align='left'>Footer</td></tr></tfoot>
</table>
```

4.55.1. "anchor" Attribute

The <tfoot> anchor attribute cannot be generated.

4.56. <th>

The <th> element is generated from a table with at least one cell.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.56.1. "align" Attribute

The <th> "align" attribute is generated from the horizontal alignment operator of a cell in a table, if present.

The following example

```
|===
|> Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>&gt; Cell</td></tr></tbody>
</table>
```

4.56.2. "anchor" Attribute

The <th> anchor attribute cannot be generated.

4.56.3. "colspan" Attribute

The <th> "colspan" attribute is generated from the horizontal alignment operator of a cell in a table, if present.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.56.4. "rowspan" Attribute

The <th> "rowspan" attribute is generated from the horizontal alignment operator of a cell in a table, if present.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.57. <thead>

The <thead> element is generated from a table with header.

The following example

```
[%header]
|===
| Header

| Cell
|===
```

is converted to

```
<table>
<thead><tr><td align='left'>Header</td></tr></thead>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.57.1. "anchor" Attribute

The <tfoot> anchor attribute cannot be generated.

4.58. <title>

The <title> element and its attributes are provided in the docinfo file or in a passthrough block containing a top <reference> element.

4.59. <toc>

The <toc> element is provided in the docinfo file.

4.60. <tr>

The <tr> element is generated from a table with rows.

The following example

```
|===
| Cell
|===
```

is converted to

```
<table>
<tbody><tr><td align='left'>Cell</td></tr></tbody>
</table>
```

4.60.1. "anchor" Attribute

The <tr> anchor attribute cannot be generated.

4.61. <tt>

The <tt> element is generated from a monospace text formatting.

The following example

```
The function ``main()``
```

is converted to

```
<t>
The function <tt>main()</tt>
</t>
```

4.62. <u>

The <u> element is generated from an u inline macro.

The inline macro is an extension to standard AsciiDoc.

The following example

```
This is u:[test]
```

is converted to

```
<t>
This is <u>test</u>
</t>
```

4.62.1. "anchor" Attribute

The <u> "anchor" attribute is generated from the ID of a u inline macro, if present.

The following example

```
This is u:[test,id=id26]
```

is converted to

```
<t>
This is <u anchor='id26'>test</u>
</t>
```

4.62.2. "ascii" Attribute

The <u> "ascii" attribute is generated from the ascii attribute of a u inline macro, if present.

The following example

```
This is u:[test,ascii=test]
```

is converted to

```
<t>
This is <u ascii='test'>test</u>
</t>
```

4.62.3. "format" Attribute

The <u> "format" attribute is generated from the format attribute of a u inline macro, if present.

The following example

```
This is u:[test,format=lit-num-name]
```

is converted to

```
<t>
This is <u format='lit-num-name'>test</u>
</t>
```

4.63. <ul>

The <ul> element is generated from an unordered list.

The following example

```
* One line
```

is converted to

```
<ul>
<li>One line</li>
</ul>
```

4.63.1. "anchor" Attribute

The <ul> "anchor" attribute is generated from the ID of an unordered list, if present.

The following example

```
[[id27]]
* test
```

is converted to

```
<ul anchor='id27'>
<li>test</li>
</ul>
```

4.63.2. "bare" Attribute

The <ul> "bare" attribute is generated from the unstyled style on an unordered list, if present.

The following example

```
[unstyled]
* test
```

is converted to

```
<ul bare='true' empty='true'>
<li>test</li>
</ul>
```

4.63.3. "empty" Attribute

The <ul> "empty" attribute is generated from the no-bullet style on an unordered list, if present.

The following example

```
[no-bullet]
* test
```

is converted to

```
<ul empty='true'>
<li>test</li>
</ul>
```

4.63.4. "indent" Attribute

The <ul> "indent" attribute is generated from the indent attribute on an unordered list, if present.

The indent attribute on am unordered is an extension to standard AsciiDoc.

The following example

```
[indent=3]
* test
```

is converted to

```
<ul indent='3'>
<li>test</li>
</ul>
```

4.63.5. "spacing" Attribute

The <ul> "spacing" attribute is generated from the spacing attribute on an unordered list, if present.

The spacing attribute on an unordered list is an extension to standard AsciiDoc.

The following example

```
[spacing=compact]
* test
```

is converted to

```
<ul spacing='compact'>
<li>test</li>
</ul>
```

4.64. <uri>

The <uri> element inside a <front> element is copied from a docinfo file.

The <uri> element can also be used inside a <reference> element in a passthrough block in the bibliography.

The <uri> element can also be used inside an <author> element in a passthrough block in a section.

4.65. <workgroup>

The <workgroup> element inside a <front> element is copied from a docinfo file.

The <workgroup> element can also be used inside a <reference> element in a passthrough block in the bibliography.

The <workgroup> element can also be used inside an <author> element in a passthrough block in a section.

4.66. <xref>

The <xref> element is generated from a cross-reference.

The following example

```
As described in <<RFC7991>>.
```

is converted to

```
<t>
As described in <xref target='RFC7991' />.
</t>
```

4.66.1. "format" Attribute

The <xref> "format" attribute is generated from the format attribute in a cross reference, if present.

The format attribute on a cross reference is an extension to standard AsciiDoc.

The following example

```
As described in <<RFC7991,format=counter>>.
```

is converted to

```
<t>
As described in <xref target='RFC7991' format='counter'></xref>.
</t>
```

4.66.2. "relative" Attribute

The <xref> "relative" attribute is generated from the relative attribute in a cross reference, if present.

The relative attribute on a cross reference is an extension to standard AsciiDoc.

The following example

```
As described in <<RFC7991,relative=#rel>>.
```

is converted to

```
<t>
As described in <xref target='RFC7991' relative='#rel'></xref>.
</t>
```

4.66.3. "section" Attribute

The <xref> "section" attribute is generated from the section attribute in a cross reference, if present.

The section attribute on a cross reference is an extension to standard AsciiDoc.

The following example

```
As described in <<RFC7991,section=id28>>.
```

is converted to

```
<t>
As described in <xref target='RFC7991' section='id28'></xref>.
</t>
```

4.66.4. "sectionFormat" Attribute

The <xref> "sectionFormat" attribute is generated from the section-format attribute in a cross reference, if present.

The section-format attribute on a cross reference is an extension to standard AsciiDoc.

The following example

```
As described in
<<RFC7991,section-format=comma>>.
```

is converted to

```
<t>
As described in
<xref target='RFC7991' sectionFormat='comma'></xref>.
</t>
```

4.66.5. "target" Attribute

The <xref> "target" attribute is generated from the target in a cross reference.

The following example

```
As described in <<RFC7991>>.
```

is converted to

```
<t>
As described in <xref target='RFC7991' />.
</t>
```

5. Informative References

[AsciiDoc] "AsciiDoc", Accessed 23 April 2021, 8 March 2021, <https://en.wikipedia.org/wiki/AsciiDoc/>.

[BetterBibTeX] "Better BibTeX for Zotero :: Better BibTeX for Zotero", Accessed 29 January 2023, <https://retorque.re/zotero-better-bibtex/>.

[draft-petithuguenin-rfc-ontology] Petit-Huguenin, M., "An Ontology for RFCs", Work in Progress, Internet-Draft, draft-petithuguenin-rfc-ontology-03, 30 January 2023, <https://datatracker.ietf.org/doc/draft-petithuguenin-rfc-ontology/03>.

[I-D.irse-draft-irse-xml2rfcv3-implemented-03] Levine, J. R., "The "xml2rfc" version 3 Vocabulary as Implemented", Accessed 31 December 2022, Work in Progress, Internet-Draft, draft-irse-draft-irse-xml2rfcv3-implemented, 19 September 2022, <https://datatracker.ietf.org/doc/draft-irse-draft-irse-xml2rfcv3-implemented>.

[RFC7322] Flanagan, H. and S. Ginoza, "RFC Style Guide", RFC 7322, DOI 10.17487/RFC7322, September 2014, <https://www.rfc-editor.org/info/rfc7322>.

[RFC7991] Hoffman, P. E., "The "xml2rfc" Version 3 Vocabulary", Accessed 1 January 2023, RFC 7991, DOI 10.17487/RFC7991, December 2016, <https://datatracker.ietf.org/doc/rfc7991>.

[WebPortionStyle] "Web Portion of the Style Guide » RFC Editor", Accessed 29 January 2023, <https://www.rfc-editor.org/styleguide/part2/>.

[Zotero] Corporation for Digital Scholarship, "Zotero | Your personal research assistant", Accessed 20 January 2023, <https://www.zotero.org/>.

6. Example References

[RFC3080] Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC 3080, DOI 10.17487/RFC3080, March 2001, <https://www.rfc-editor.org/info/rfc3080>.

[RFC6323] Renker, G. and G. Fairhurst, "Sender RTT Estimate Option for the Datagram Congestion Control Protocol (DCCP)", RFC 6323, DOI 10.17487/RFC6323, July 2011, <https://www.rfc-editor.org/info/rfc6323>.

[RFC6429] Bashyam, M., Jethanandani, M., and A. Ramaiah, "TCP Sender Clarification for Persist Condition", RFC 6429, DOI 10.17487/RFC6429, December 2011, <https://www.rfc-editor.org/info/rfc6429>.

[RFC7322bis] Levine, J., Ed. and S. Ginoza, "RFC Style Guide", Work in Progress, Internet-Draft, draft-flanagan-7322bis-07, 7 April 2021, <https://datatracker.ietf.org/doc/draft-flanagan-7322bis-07>.

[STD13] Mockapetris, P., "Domain names - concepts and facilities", RFC 1034, November 1987.
Mockapetris, P., "Domain names - implementation and specification", RFC 1035, November 1987.

[STD72] Gellens, R. and J. Klensin, "Message Submission for Mail", RFC 6409, STD 72, November 2011.

Appendix A. Command Line Tool

The "asciidoctor-xml" Ruby gem is distributed as a git repository that can be installed as follow:

```
git clone git://shalmaneser.org/sniElnixoip1 asciidoctor-xml2rfc
cd asciidoctor-xml2rfc
gem build asciidoctor-xml2rfc.gemspec
sudo gem install asciidoctor-xml2rfc-0.1.gem
```

Then an AsciiDoc file can be converted into an XML2RFC v3 file like this:

```
asciidoctor-xml2rfc mapping-xml2rfc-asciidoc.adoc
```

Note that, if present, the value of the "doc-name" document header attribute will be used as the name of the output file. This permits to keep the same file name but generate different versions by just changing the doc-name header.

The command will write the name of the generated file on the standard output, so it can be used to generate the final document:

```
xml2rfc -P --html --text \
  $(asciidoctor-xml2rfc mapping-xml2rfc-asciidoc.adoc)
```

The "-P" option generates the text file without pagination, which permits to do a local diff between different versions of a draft (e.g., with the ":vert diffs <previous-file>" command in VIM).

Appendix B. Build a Bibliography with Zotero

The simplest way to build a bibliography is to use [Zotero] and the AsciiBib exporter provided in the git repository. The Better BibTex for Zotero [BetterBibTeX] add-on is needed to manage the citation keys for the bibliographic items exported.

First the "AsciiBib.js" file must be copied, or better symbolically linked, in the "data/translators" directory under the Zotero configuration directory (most likely "~/.zotero").

To build a bibliography for an Internet-Draft, first create a new collection in Zotero and add bibliographic items in it. E.g., a collection named "mapping-xml2rfc-asciidoc" contains a sub-collection named "informative", which itself contains all the references for this document.

Bibliographic items can be created from scratch, scrapped from the Internet using web translators, or imported from an existing bibliography. The git repository distributed with [draft-petithuguenin-rfc-ontology] contains a Zotero importer that can create the full collection of RFCs items in Zotero from the "rfc-index.xml" file.

A Zotero collection can then be converted into a AsciiBib file by right-clicking on it, choosing "Export Collection...", select the "AsciiBib" format, then finally the target directory. This will create an AsciiDoc file that can be imported in a bibliography section:

```
include::examples.adoc[]
```

The Zotero exporter tries to export bibliographic items in a form that is as close as possible to the RFC Style guides [RFC7322] [WebPortionStyle]. The Section 6 bibliography in this document shows examples of bibliographic items converted from Zotero:

RFC with one author or editor: [RFC3080]

RFC with two authors or editors: [RFC6323]

RFC with three or more authors or editors: [RFC6429]

STD or BCP that contains one RFC: [STD72]

STD or BCP that contains two or more RFCs: [STD13]

Internet-Draft: [RFC7322bis]

Changelog

draft-petithuguenin-xml2rfc-asciidoc-01: Document:

-Add explanations for citation inclusion.
-Add new appendix that explains how to use Zotero to generate a bibliography.

*Tooling:

-Implemented citation extraction.
-The name of the generated file is written on the standard output.
-AsciiBib Zotero exporter.

Author's Address

Marc Petit-Huguenin
Impedance Mismatch LLC

Email: marc@petit-huguenin.org