Writing I-Ds and RFCs using XML2RFC

Elwyn Davies
Presented by Margaret Wasserman
66th IETF
Montreal, Canada
Why are we here?
To learn an easy way to create well-presented I-Ds ...
... and help avoid sleepless night with idnits

xml2rfc
• makes it easy to create a well-presented document
  • looks after the boiler plate for you
  • leaves you more time to do the engineering
Agenda

- What is xml2rfc?
- Demonstration of xml2rfc using XMLmind
- The way of XML
- Describing your document
  The Language of xml2rfc
- Tools for the job
- Fine tuning the result
  Processing Instructions
- Extra Tips and Tricks
- Resources to help you
- Questions and (hopefully) Answers
The nature of RFCs and I-Ds

RFCs and I-Ds have a relatively simple format

See Instructions to RFC Authors (rfc2223bis) at:
ftp://ftp.rfc-editor.org/in-notes/rfc-editor/instructions2authors.txt

‘Front, Middle, Back’

Front & Back contain a lot of standard ‘boiler plate’

BUT – ‘boiler plate’ is important legal stuff – RFC3978/9
- Has to be there! – IPR and Copyright positions
- Has to be right (and this week’s version)!

The technical core is the ‘filling’ in the Middle
What is in the sandwich filling?

Numbered sections – ‘outline numbered’

Tree structure of sections/sub-sections/sub-sub-sections...

In the sections...

Paragraphs of text

Lists and indentation defined by the author

Author (mostly) doesn’t need to micro-manage word layout

Words can be laid out by the tools

Tables, Figures, Pieces of Example Code, ABNF, MIBs, etc

Layout of these is critical – needs author control

References

Cross references to other parts of the document

References to external documents
Requirements for xml2rfc

Technical requirements:

Automate:
- structure and numbering
- references
- producing table of contents and reference lists
- producing correct overall document & page layout

Insert the right boiler plate

Political Requirements

ASCII input
- Standards compliant solution
- Easy learning and editing
- Free tools
- Simple and fast operation
So why should I do it this way???

A small demonstration....

There are trade-offs today:
  Speed & Convenience vs Absolute Precision Control of Layout
  'Good enough' in multiple formats vs Exact control of content per format

Adopting xml2rfc today doesn't stop it getting better!
  The trade-offs are not fundamental problems
  Improvements are possible and happening
  User input is essential:
    Minimum complexity for maximum functionality
Markup Languages and xml2rfc

Basic Solution for requirements:
1. Document Description Language (aka Markup Language)
2. Transformation Tool

A standard markup language is XML (from W3C)
XML = eXtensible Markup Language

RFC 2629 (and its unofficial successor) define an XML Document Type Description (DTD) for RFCs and I-Ds
Reflects the required structure of I-Ds and RFCs
Also good for other sorts of technical memos
The xml2rfc Tool

A tool to transform source text into output text
Source text: conforms to RFC/I-D DTD
Output formats:
• ASCII text (standard form or unpaginated), or
• HTML (with hyperlinks & more elegant formatting)
• nroff markup language
  (because that is what the RFC Editor archives)
DISCLAIMER:
This is NOT a course on XML
   Just enough XML to understand and use the xml2rfc DTD
Syntax is (deceptively) simple!
Basic Principles of XML

XML markup uses markup ‘elements’ embedded in the ordinary text

Elements have three purposes
  - Provide document structure
  - Provide semantic context for the content
    i.e., what it ‘means’ in some sense
  - Control the formatting of an output document

Elements impose a strict tree structure

Exactly one root element in each document
Characters

Special Characters in XML – need to be ‘escaped’ in normal text

< introduces ‘elements’

& introduces ‘entities’

**WARNING**: Letter case is significant in XML:

so…

A ≠ a  <rfc> ≠ <RFC>
XML Elements

An **Element** consists of 3 parts:
- **Start tag** containing the element name
- **End tag** repeating the element name
- **All the characters in between**

**Example for an element named ‘example’:**

```
<example>text and/or NESTED elements</example>
```

**Elements** must be properly nested (unlike HTML)

**Shorthand - if the “text in between” is empty**

```
<example/> ≡ <example></example> - an ‘empty element’
```
Attributes

Attributes are part of elements

If an element has attributes
they appear in the start tag
after the element name

Example:

```xml
<example name='value'>blurb<example />
```

They can also appear in empty elements:

```xml
<empty name="value" />
```

The value **MUST** be quoted

Use either matched single ('') or double quotes (""")

If the value has one sort of quote in it, use the other one
Entities

An **Entity** is a textual macro

Example:

```
&macro_name;
```

The ‘value’ of the macro replaces the complete entity in the output.

Mostly needed as escapes for `&` and `<`

```
&amp; \equiv & 
&lt; \equiv <
```

xml2rfc also predefines some other entities, e.g.,

```
&quot; ("), &apos; (') -- needed in attributes with both " and '
&gt; (>), &nbsp; (non-breaking space)
&ndash; (a short dash ‘-’), &mdash; (a longer dash ‘—’ or ‘–’)
```
Tokens

A **Token** is a string of characters
Starts with letter or underscore
Followed by
  - letters, numbers, underscore, hyphen or period (.)
For regular expression fiends: \[A-Za-z_\] \[A-Za-z0-9_\.\-] *

Examples:
  - _Token_string19
  - a

XML names have to be **tokens**, including
  - Element names
  - Attribute names
  - Entity names

**NB**
No Spaces

Technically, you can also use ':' in tokens but this should be avoided because it has special meaning in XML (namespaces).
Other things starting with <

Comments
All the text between <!-- and -->
BEWARE: Nested comments are not possible!

Processing Instructions
<?processor_target pi_name='value' ?>
processor_target for xml2rfc is rfc

Defining Entities
<!ENTITY name "value" > (and some variants)
Note: no ‘=’ between name and value

Literal text – CDATA block
All the text between <![CDATA[ and ]]>
When is a space not a space?
The significance of white space in xml2rfc

First the easy one:
   Inside CDATA blocks
      white space is copied literally to output

Then where it just makes the XML more readable
   Inside tags extra white space around tokens doesn’t change meaning
   BUT beware of splitting up multi-character ‘atoms’ … watch out for
      comment delimiters:          <!-- and -->
      end tag markers:            </ and />

Example:
   <example name="value"/> ≡ <example name ="value" />

In the text between the tags of an element (outside CDATA)
   Generally any amount of white space together is treated like ‘one space’
   Output layout depends on the formatting tool
   Allows ‘tidy’ XML source
      Indented to show structure.
Describing your document
The language of xml2rfc

Alpha and Omega:

```xml
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE rfc SYSTEM 'rfcXXXX.dtd'>
<rfc>
   ...
</rfc>
```

1. XML Declaration: Must be first line; ‘encoding’ is optional
2. Reference to DTD used: currently rfcXXXXX.dtd => rfc2629.dtd
3. The root ‘rfc’ element start & end tags – No text after end tag!

Treat 1 and 2 as opaque strings for now

We’ll look at the attributes of <rfc> later
Matching and Nesting

ALL elements MUST be properly MATCHED and NESTED

Matching: <example> must eventually be followed by </example>

Empty elements are inherently matched

Nesting: Elements are properly nested if they don’t overlap

Elements not overlapped:

```xml
<outer>
  ...
  <inner>
    ...
  </inner>
  ...
</outer>
```

Elements overlapped:

```xml
<outer>
  ...
  <inner>
    ...
  </inner>
  ...
</outer>
```

Properly nested ✔

NOT properly nested ✗
Overall Structure

RFCs and I-Ds have a `<front>`, `<middle>` & `<back>`

```xml
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE rfc SYSTEM 'rfc2629.dtd'>
<rfc>
  <front>
    <!-- The front matter goes here -->
  </front>
  <middle>
    <!-- The technical sections go here -->
  </middle>
  <back>
    <!-- The back matter goes here -->
  </back>
</rfc>
```
Front Matter

<front> follows straight after <rfc>

Order of elements in <front> matters!

<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE rfc SYSTEM 'rfc2629.dtd'>
<rfc>
  <front>
    <title ...> Must be present
    <author ...> One or more
    <date ...> Must be present
    <area ...> Zero or more
    <workgroup ...> Zero or more
    <keyword ...> Zero or more
    <abstract ...> One (or zero) - must have for I-D/RFC
    <note ...> Zero or more
  </front>
  ...
</rfc>
The title Element

Specifies the title of the document:

```
<title abbrev='Much Ado about Nothing'>
   The IETF's Discussion on
   "Source Format of RFC Documents"
</title>
```

**Abbreviation** gives short form for page headers

Needed if full title longer than 39 characters

Actual space available varies according to the month in date!

Full title used if omitted
The author Element

One for each document author

... the ones with their names on the front page

Each author that is a person must have attributes

- **initials**
- **surname**
- **fullname**

```xml
<author initials='F.J.' surname='Flintstone'
    fullname='Frederick Flintstone'>
```

Optional **role** attribute: must have value 'editor' if used

**author** element consists of

- **organization** element (exactly one required), plus
- **address** element (optional)
The organization Element

Very similar to title element

<organization abbrev='IETF'>
    Internet Engineering Task Force
</organization>

The abbreviation will be used on the front page
Full organization name used in ‘Authors’ section

Must be present but can be empty if not relevant
The address Element

Consists of up to 5 elements – each is optional

 postal - phone - facsimile - email - uri

 postal element consists of

 One or more street elements, followed by

 Any combination of up to one each of elements

city - region (state/province) - code (zip/postal) - country

 Allows for different national flavours of postal addressing

 Formatters have to preserve the order of elements

country text should be a two letter code from ISO3166

The good news: there are no attributes to remember

Tip: Exchange author elements with fellow authors
address Element Example

Notice how indentation is used to highlight structure

```xml
<address>
  <postal>
    <street>301 Cobblestone Way</street>
    <city>Bedrock</city>
    <region>CA</region>
    <code>94110</code>
    <country>US</country>
  </postal>
  <phone>+1 916 555 1234</phone>
  <email>fred@example.com</email>
  <uri>http://example.com/</uri>
</address>
```

Please use full international phone numbers with country codes in all cases!
The date Element

Specifies the publication date of the document

date element has day, month and year attributes

No text between tags – so always an empty element

Current rules (@ xml2rfc v1.30, under review):

Day and month are optional, year is currently required
If day and month are not specified
   Today’s day and month are used by xml2rfc tool
      irrespective of year (silly if not current year)
If month is specified but not day:
   Today’s day is used if month and year match today’s date
   Otherwise, the day is not output

<date month='March' year='2006' />
Meta-data Elements

Document meta-data is specified in area, workgroup and keyword elements.

Zero or more of each type is allowed – order matters.

What happens to meta-data?

- **workgroup**: Replaces “Network Working Group” in page 1 header.
- **area**: Is not used in any format as far as I can tell!
- **keyword**: In HTML they are output in meta keywords tag; not used in text/nroff.

```html
<area>General</area>
<workgroup>RFC Beautification Working Group</workgroup>
<keyword>I-D</keyword>
<keyword>XML</keyword>
<keyword>Extensible Markup Language</keyword>
<keyword>Anything else that might be relevant</keyword>
```
The abstract Element

A document MAY have an abstract element

But the I-D Editor and RFC Editor get upset if they don’t

The abstract contains one or more t elements

(t element = paragraph of text – more later)

Generally one t element is considered enough for an abstract

<abstract>
  <t>This memo presents a technique for using XML (Extensible Markup Language) as a source format for documents in the Internet-Drafts (I-Ds) and Request for Comments (RFC) series.</t>
</abstract>
The note Element

Documents may have one or more note elements
note element consists of one or more t elements
Mandatory title attribute printed before note
Usual usage is for comments from the IESG

<note title='IESG Note'>
  <t>The IESG has something to say.</t>
</note>
What about the Boiler Plate?

How to distinguish an I-D from an RFC....

RFC and I-D have different `rfc` element attributes

For I-D: Specify Document Name (`docName`) & IPR Position (`ipr`)
If relevant: numbers of RFCs it `obsoletes` and/or `updates`

For RFC: Replace Document Name with RFC Number
See RFC2629 for more details ... mostly for RFC Editor use

For ‘usual’ I-D (default IPR terms):

```xml
<rfc ipr='full3978' docName='draft-mrose-writing-rfcs-01'>
```

Alternative IPR – see RFC 3978 for meaning:

Use `ipr='noModification3978'/'noDerivatives3978'`

Optional `iprExtract` gives ‘anchor’ of section which can be extracted for separate use (like a MIB)

xml2rfc will now handle all the boiler plate

Remember to change the version #. This is NOT the file name (but xml2rfc won’t check)!
A Whole Lot of Front

<title>
Writing I-Ds and RFCs using XML
</title>

<author initials='F.J.' 
surname='Flintstone' 
fullname='Frederick Flintstone'>
<organization>
Slate Rock and Gravel, Inc.
</organization>

<address>
<postal>
street>301 Cobblestone Way</street>
<city>Bedrock</city>
<region>CA</region>
<code>94110</code>
<country>US</country>
</postal>
</address>

<phone>+1 916 555 1234</phone>
<email>fred@example.com</email>
<uri>http://example.com</uri>
</author>

<date month='February' year='1999' />

<!-- Meta-data -->
<area>General</area>
<workgroup>
RFC Beautification Working Group
</workgroup>
<keyword>RFC</keyword>
<keyword>Request for Comments</keyword>
<keyword>I-D</keyword>
<keyword>Internet-Draft</keyword>
<keyword>XML</keyword>
<keyword>Extensible Markup Language</keyword>

<abstract>
This memo presents a technique for using XML (Extensible Markup Language) as a source format for documents in the Internet-Drafts (I-Ds) and Request for Comments (RFC) series.
</abstract>

<!-- continued... -->
The Middle

The **middle** element contains all document sections

Except for bibliography (references), the boilerplate & appendices

i.e., all the really interesting bits!

It is very simple...

```xml
...</front>
<middle>
  <section ...>
  <section ...>
  ...>
  <section ...>
</middle>
<back>
... 
```

Note:
Ignore the bit in RFC2629bis about appendices in **middle**!
The section Element

section elements are the core of a document

Must have a title attribute

Optionally has

anchor attribute – needed for cross-referencing with xref

anchor value must be an XML Token – no spaces, limited punctuation!
(xml2rfc may be more forgiving about this!)

toc attribute – controls if title is in Table of Contents

Choices are

include – force it in
exclude – force it out
default – in or out depends on the ‘level’ of the section
(default is the default)

<section anchor='intro' title='Introduction'>
  ...
</section>
What’s in a section?

Each section contains any number & combination of t, figure, texttable, xref & nested section elements

    <section title='The Middle'>
        ...
        <section title='The section Element'>
            ...
            <section title='The t Element'>...
            <section title='The list Element'>...
            <section title='The figure Element'>...
            <section title='The texttable Element'>...
            <section title='The xref Element'>...
            <section title='The eref Element'>...
            <section title='The iref Element'>...
            <section title='The cref Element'>...
            <section title='The vref Element'>...
        </section>
    </section>
Outline Numbering

The `section` element is recursive...

Recursion level determines numbering:

<table>
<thead>
<tr>
<th>XML</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;section title=&quot;Top Level&quot;&gt;</code></td>
<td>1. Top Level</td>
</tr>
<tr>
<td><code>&lt;section title=&quot;2nd Level&quot;&gt;</code></td>
<td>1.1 2nd Level</td>
</tr>
<tr>
<td><code>&lt;section title=&quot;3rd Level&quot;&gt;</code></td>
<td>1.1.1 3rd Level</td>
</tr>
<tr>
<td><code>&lt;/section&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;/section&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>&lt;section title=&quot;Next Top Level&quot;&gt;</code></td>
<td>2. Next Top Level</td>
</tr>
<tr>
<td><code>&lt;/section&gt;</code></td>
<td></td>
</tr>
</tbody>
</table>
The ‘t’ element

Fundamental but slightly odd element!

Basically a paragraph of text
   Output is rearranged to form ‘right ragged’ filled lines

Text can contain elements to produce…
   Embedded lists (list)
   References of various kinds (xref, eref, iref and cref)
   Formatting guidance
      Layout hints (vspace)
      Parts of text that should be rendered specially (spanx)

Originally figure elements could be in t elements
   This is now deprecated – they should be directly in sections

Note: RFC2629 is misleading: the t element is NOT directly recursive (but lists can contain more t’s)
Lists

The list element contains one or more items

Each item is a t element

Means list elements can be (indirectly) recursive

Some text before the list.

<list style='numbers'>
  <t>The first item.</t>
  <t>The second item which contains two bulleted sub-items:
      <list style='symbols'>
        <t>The first sub-item.</t>
        <t>The second sub-item.</t>
      </list>
  </t>
</list>

Some text after the list.

Some text before the list.

1. The first item.
2. The second item which contains two bulleted sub-items:
   - The first sub-item.
   - The second sub-item.

Some text after the list.
Lots of Styles of Lists

The list element has an optional style attribute

- style='empty': Generates indented paragraph (default)
- style='numbers': Numbered items using arabic numbers
  - Each new (sub-)list starts again from item #1
- style='letters': Alphabetic lists using lower case (a, b, ...)
- style='symbols': Bulleted lists
  - Level determines bullet symbol - use 'format' for alternatives
- style='hanging': Items with 'hanging' labels
  - Label taken from optional hangText attribute on t
- style='format {str}': Auto-formatted lists
  - {str} is used as label
  - Can contain either %d or %c exactly once
  - counter attribute specifies an auto-increment variable substituted for %d (decimal #) or %c (letter)

Space between items depends on formatter. More later.
Hanging Labels

- **counter:** the "counting designation" is rendered (e.g., "2.1" or "A.2");

- **title:** the title attribute of the corresponding element is rendered (e.g., "XML Basics");

- **none:** no additional designation is rendered; or,

- **default:** a suitable designation is rendered, e.g., "Section 2.1" or "&lt;a href='#xml_basics'>XML Basics&lt;/a>" (the default).
Auto-formatted Lists

R1: Text for R1.
R2: Text for R2.
...
Directive a: Text for A.
Directive b: Text for B.
...
R3: Text for R3.
Controlling Indentation

The indentation of the item text can be adjusted...
for all kinds of list elements

```xml
<list style='format R%d:'
    hangIndent='5'>
    <t>Text for R1.</t>
    <t>Text for R2.</t>
    <t>Text for R3.</t>
    <t>Text for R4.</t>
    <t>Text for R5.</t>
    <t>Text for R6.</t>
    <t>Text for R7.</t>
    <t>Text for R8.</t>
    <t>Text for R9.</t>
    <t>Text for R10.</t>
</list>
```

R1: Text for R1.
R2: Text for R2.
R3: Text for R3.
R5: Text for R5.
R7: Text for R7.
R8: Text for R8.

PS: Lists are being improved from v1.31, including more auto-formats and bug fixes to nested lists.
Figures

Used to display ASCII ‘artwork’ - where horizontal and vertical whitespace is significant!

figure element contains elements:

- preamble (optional) - contains text - rendered like a t element *
- artwork (required) - all whitespace is significant here **

Use a CDATA block if lots of < or & in the figure

- postamble (optional) - as for preamble

figure has attributes anchor, title and align

- anchor, title: Same as attributes for section element
- align: Alignment for all components

WARNING: Figure numbering only works properly if all figures have non-empty anchor attributes

* except no list elements allowed
** no elements allowed in artwork - pure text
‘Typed’ Artwork

*figure* element also good for text where layout is significant, e.g.,
Code samples, algorithms, ABNF, MIBs and PIBs

*artwork* element has optional attributes

  **type**: May do some clever verification/display for special values
currently only for *abnf* in v1.31 and up - colorized HTML!
may work for *mib* and *pib* in v1.31; *xml* in future?

  **align**: 'left', 'center' or 'right' - overrides *figure* alignment

Default for **align** is same as for parent *figure* element

**name**: Something to do with filenames - ignore it for now!

*figure* and *artwork* have extra attributes -
only used for HTML output - see xml2rfc README
<figure anchor="my_figure"
title="Some boxes">
  <preamble>The preamble is printed before the figure.</preamble>
  <artwork><![CDATA[
      ***** +-------+
      * && * <-->| box |
      ***** +-------+
    ]]>"
  </artwork>
  <postamble>The postamble ambles along after it.</postamble>
</figure>

The preamble is printed before the figure.

***** +-------+
* && * <-->| box |
***** +-------+

The postamble ambles along after it.

Figure 1: Some boxes

None of this appears if the anchor attribute is not present.
The texttable Element

Used for generating tables (surprise!)

Very similar to **figure**

Has **preamble** and **postamble**, plus same attributes

Default for texttable element **align** is 'center'

**artwork** is replaced by

**ttcol** elements (at least one) - column headers with attributes

**width** (optional) - % of available space occupied (e.g. '30%')

Rest distributed equally over columns without width attribute

**align** (optional) - how cell contents are justified

'left' (default), 'center' or 'right'

**c** elements - contents of each cell

Order: left to right along row 1, then repeat for other rows

Can include references and index elements
A Very Simple Table

So, putting it all together, we have, e.g.,

| ttcol #1 | ttcol #2 |
+----------+----------+
| c #1     | c #2     |
| c #3     | c #4     |
| c #5     | c #6     |

which is a very simple example with no title.

Table 1
Internal Cross-References

Almost all pieces of text can contain \texttt{xref} elements

Exception: \texttt{artwork}

Cross-reference can refer to any \texttt{anchor} attribute

From \texttt{section}, \texttt{figure}, \texttt{texttable}, bibliographic reference

What gets into the output?

If \texttt{xref} is an empty element... e.g.,
\begin{verbatim}
as described in <xref target='xml_basics' />.
\end{verbatim}
xml2rfc inserts ‘an appropriate phrase’
.... and that depends on optional \texttt{format} attribute
.... and, also, on the output format - HTML gets hyperlinks

\begin{itemize}
  \item \texttt{counter} Just section number, figure/table number or reference index
  \item \texttt{title} Value of \texttt{title} attribute (doesn’t work for bibliographic refs)
  \item \texttt{none} Same as default for empty elements. Otherwise nothing extra.
  \item \texttt{default} "Section 2.1"/"Section A.4", "Figure 5", "Table 2", "[17]"/"[RFC2233]"
\end{itemize}
Non-empty Cross References

If the xref element has content, e.g.,

You will find it at `<xref target='intro'>the start</xref>`.

xml2rfc adds ‘appropriate designation’ to content...
something like ‘the start (Section 1)’

... and again that depends on `format` attribute and output format

Guarantee: The choice will be consistent over one document!

`format='none'` is useful for HTML output....

The text of the hyperlink is just the `xref` element content

You need to experiment to see the possibilities!
Hints about Formatting - 1

The `vspace` element can only be used in `t` elements

Tells formatter to leave some blank lines

`blankLines` attribute [NB upper case L!] indicates how many
Default is 0 - this forces a physical line break but no blank lines
The `vspace` element is *always* empty - contents discarded (or error)

The amount of blank inserted should never extend beyond the end of the current page in (text) output

Using `blanklines='100'` will force a page break (100 > page length!).

```xml
<list style='numbers'>
   <t>This is a list item.
       <vspace blankLines='1' />
   This is part of the same list item, although when displayed, it appears as a separate paragraph.</t>
</list>
```

This is a useful trick!
More possibilities for the Middle

For most I-Ds, this should be enough

Some extra capabilities not used in most I-Ds

eref element - external references
iref element - index mechanism
cref element - for review comments
spanx element - for fount hints and controlling line breaks
(more relevant to HTML output)
After the ‘Middle’

What’s left to do?
- Bibliographic references
- Appendices

What comes automatically?
- Authors’ Addresses section
  - Doesn’t cover contributors - not automated now (but may be)
- More boiler plate
  - IPR Statement
  - Disclaimer of Validity
  - Copyright Statement
  - ISOC Acknowledgement (only)

You have to do the ‘Oscars Speech’ thanks section in ‘middle’!!
The Back Matter

References and Appendices are here...

... 
</middle>
<back>
  <references>
    <reference ...>
    <reference ...>
  </references>
</back>

(The end of the ‘middle’)

Zero or more references elements each containing one or more reference elements

Zero or more section elements which will be labelled as ‘Appendix A.’ etc in the output

(REALLY THE END)

Appendix sections can have nested sub-section elements
Labelled A.1, A.1.2, etc.
References Sections

In the beginning...
   There was a single and undivided ‘References’ section
But then there was an edict from on high...
   References shall be divided one from the other and they shall be ...
   Normative References, and
   Informative References

So the references element got a title attribute...

<references title="Normative References">
   <reference> ... </reference>
</references>

The normative references

<references title="Informative References">
   <reference> ... </reference>
</references>

The illustrative references

Output as sub-sections of a numbered ‘References’ section

URIs from any erefs get put in third unnumbered ‘URIs’ reference section
References the Hard Way

The Bad News:
Computing a reference manually is tedious
Arguably the worst task in xml2rfc!

```xml
<reference anchor='RFC2200' target='http://a.org/doc'>
  <front>
    <!-- Provides title, author(s) and date info -->
    <title>A Good Read</title>
    <author> ... </author>
    <date month='June' year='1997' />
  </front>
  <seriesInfo name='RFC' value='2200' />
  <seriesInfo name='STD' value='1' />
  <format type='TXT' octets='94506'
    target='ftp://ftp.isi.edu/in-notes/rfc2200.txt' />
</reference>
```

xref points at anchor
same as front in rfc
but only these bits...
Compulsory title
One or more authors
Compulsory date
(no meta-data, abstract)
Zero or more
seriesInfo
Format is optional
(only used for HTML)

... but now some tips to reduce the pain ➔
If You Have to do it the Hard Way...

If you want to show a URL for the reference material ... put it in `target` attribute of `reference` element, NOT in `format` element

Authors need an `organization` attribute but...

If the author has a name, the `organization` isn’t displayed...

So, you can leave it empty

If the ‘author’ is an organization, put it in `organization` attribute, and ... Omit the `fullname`, `initials` and `surname` attributes

If there are ‘many’ authors - no way to get ‘et al’ but...

Put in first few - last one has no name and has org of ‘others’

Authors don’t need `address` attribute - it isn’t displayed

The values of the `seriesInfo` attributes are just text

They are concatenated (with a space in between) and displayed as is

Useful for any other info (like a book publisher or ISBN #)

`format` elements are optional - only used in HTML output
The Easier Way

The Good News:
You mostly don’t have to do it the Hard Way
Tools can take the pain out of the remaining ones (e.g., XMLmind)

Using bibliography databases
The author(s) of xml2rfc maintain ‘citation libraries’
Libraries at http://xml.resource.org/public for
  IETF RFCs and Internet Drafts (automatically updated hourly!)
  W3C and 3GPP documents
  Miscellaneous (selected documents from
    ANSI, CCITT, FIPS, IEEE, ISO, ITU. NIST. OASIS and PKCS)
Also Jabber Enhancement Proposals from jabber.org

Reference citations can be imported automatically
Directly from the original libraries with a network connection
Or from a local copy (but you have to keep it up to date!)

Citation libraries have a file per ref with
xml2rfc reference element source
Howto for Citations - 1

Two things to do to add (say) a reference to an RFC:

Remember the two lines at the start of the file

<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE rfc SYSTEM 'rfcXXXX.dtd'>

For each RFC reference you need, add an Entity to:

<!DOCTYPE rfc SYSTEM 'rfcXXXX.dtd'[
<!ENTITY RFC2119 SYSTEM

This defines a new entity
name: RFC2119
value: XML for the reference
to RFC 2119

To use the new entity:
In the references element
Insert &RFC2119; instead of <reference>...</reference>

SYSTEM could be PUBLIC ""
That’s an empty string
URL could also be a local file name - then you don’t need network access to
process doc... see also ‘include’ PI
Howto for Citations - 2

The entity name for the entity is *your choice*

The citation file chooses the **anchor** for the ref

For RFCs it is ‘RFCxxxx’ - *Always* 4 digits - left padded with 0

It is OK to choose the entity name to be the same as anchor!

For Internet Drafts, e.g. draft-aboba-802-context-02.txt

anchor is I-D.aboba-802-context

Always references the most current version - convenient!

For other series... Go look at the files!

If you *must* reference expired drafts......... !

Cannot rely on the citation database forever!

To use the reference - use the **anchor** in an *xref*

Just like if you had defined it the hard way!
Now You are Ready to Use xml2rfc!

Getting started:

Use the template... Helps you remember the ‘clichés’!
Plagiarize somebody else’s source!
Use some helpful tools.

Basic resources:

The xml2rfc website: http://xml.resource.org/
The xml2rfc mailing list mailto:xml2rfc@lists.xml.resource.org
    and its archives
    http://drakken.dbc.mtview.ca.us/pipermail/xml2rfc/
Tools: What do You Need?

Absolute minimum:
  A ‘bog standard’ text editor - e.g., vi or emacs
  Access to a web browser to use the online xml2rfc tools
  Oh! And a computer to run them on!

Reasonable Outfit adds:
  XML Syntax-aware text editor (suggestions at back of slide pack)
    Colorizing XML syntax elements
    Doing smart indentation of structure (‘pretty printing’)
    Optionally, checking XML structure using the xml2rfc DTD
  TCL installation allowing local use of xm2rfc tools offline

Desirable:
  XMLmind + Bill Fenner’s xxe plugin - gives WYSIKN editing
xml2rfc - The Tool

Written in TCL scripting language
  Runs on any platform that supports TCL
  Command line or GUI operation
  Deals with vagaries of Windows vs Unix filing systems

Online tool available on web site -
  http://xml.resource.org/
  Many people prefer the convenience of this
  Hassle-free access to citation libraries

Or download the tool for local use
  You may wish to download the citation libraries also...
    Gives you ability to work freely without net access

Tool is still developing - contribute your ideas!
xml2rfc Screen Shots - Windows

xml2rfc - web service

xml2rfc - local tool
The XMLmind Plug-in

Available (free) from  
http://rtg.ietf.org/~fenner/ietf/xml2rfc-xxe/  

Gives you WYSIKN editing  

Plug-in gives you
Automatic conformance to DTD structure  
‘Graphical’ editing of sections, anchors, lists, cross-ref, etc.
Reminds you of available attributes  
Word processor-like behavior of ‘enter’ key  
Creates new paragraph or list item  
Menu items to validate/format document from within XMLmind

A few limitations at present
Limited handling of include files  
Limited texttable support  
Adding new citation library entries.

xml2rfc tools - 5
XMLmind Screenshot
Extra Tools

Alternative formatter: XSLT transformation

Various utilities:
- XML validators
- rfcdiff
- htmltidy - pretty printer for XML
- JavE - ASCII artwork editor
- Internet Explorer as an XSLT transformer

More details on all of these at the back of the pack
Templates and Scripts

From contrib directory in xml2rfc source code archives

xml2rfcpp.pl - perl script to merge text from include PIs (Alex Rousskov)
  Output doesn’t need access to any local files
  useful before sending xml2rfc to RFC Editor.
new-draft.xml - bare bones template for new I-Ds (Fred Baker)
template*.xml - 3 increasingly complete templates (Pekka Savola)
  (template1b.xml developed into template-edu-full-01.xml)
concat.pl - another perl script to merge include PIs (Rob Austein)
fast-sync.sh - script to fast sync a local citation cache (Rob Austein)

Developed for this course

template-edu-full.xml - examples of many techniques in slides
template-edu-bare.xml - above with commentary stripped
  intended as a starting point for new drafts
pi-sorted.xml - a complete commented list of PIs ready for use
  arranged according to categories used in these slides
Processing Instructions

<?rfc pi_name="value" (possibly more PIs) ?>

PIs change the behavior of xml2rfc applications

Many have ‘boolean’ values: value is either "yes" or "no"

Categories of PIs:

- File Inclusion
- Rigor Control
- Rendering Control
- Table of Contents Control
- Format Control
- HTML Specials
- Debugging Assistance

A couple of the directives are new in v1.31

plus there will be one minor change of defaults
Most Popular PIs...

I-Ds will mostly want to use

include: Allows maintaining a document in manageable chunks
strict: Innoculates you against many idnits complaints
symrefs: RFCs use anchors rather than numeric references
sortrefs: Saves having to think about the order of references
toc: Many I-Ds will benefit from a Table of Contents
compact: To save the planet when printing out the I-D
subcompact: Compromise between æsthetics and planet saving
needLines: To avoid annoying splitting of figures across pages
   (This one is used wherever this is a problem)

Complete list at the end of this slide pack
Where to Put PIs in Source

Most should be once at beginning of file
A few can be used multiple times embedded in text

If editing with XMLmind
Best placed just after `<rfc>` rather than before `<rfc>`
At Position 2 below...

```xml
<?xml version="1.0" encoding="US-ASCII" ?>
<!DOCTYPE rfc SYSTEM 'rfc2629.dtd'>
<!-- PIs can go here - Position 1 -->
<?rfc strict="yes" ?>
<rfc>
  <!-- Or PIs can go here - Position 2 -->
  <?rfc sortrefs="yes" symrefs="yes" ?>
  <?rfc toc="yes" ?>
  <front>
    ...
</rfc>
```

XMLmind will display these but can’t edit them
XMLmind displays these differently: can add new ones and edit them.
Remember to set PI target to 'rfc'

xml2rfc PIs - 3
File Inclusion

Split up a large doc or include bits of ‘boiler plate’
  e.g., Author element blocks, reference elements
  Can be used as an alternative to external entities, BUT...
  WARNING: XSLT transformations don’t understand ‘include’

<?rfc include "file_or_uri_to_include.xml" ?>

Remember file names may be case sensitive in some OSs!
Content interpreted immediately
  Other directives in same PI may cause unexpected effects

Finding the included file if the name is ‘relative’...

If XML_LIBRARY environment variable is set
  Gives search path of possible locations for ‘relative’ file names
  Directory separator is ‘usual’ one for OS (; or :)

Otherwise: In the directory where the file with the PI is found
See the README file for ways to set up XML LIBRARY

Should you use include or ENTITY?
Less typing vs more portability: YMMV
Hints, Tips and Tricks

An eclectic collection of stuff to help you...

Need to learn more about XML:
Try http://www.w3schools.com/xml/default.asp

xml2rfc is still a work in progress
The latest developments can be found at http://xml.resource.org/experimental.html
You may find something to fix your problem there!
Example: extra wide figures - v1.31 is trialling extra artwork attribute values for finer control of alignment - figures can now use the whole width of the page if needed.
Getting Your First Output..

... can be frustrating!

Invalid XML can be hard to debug:
  Missing </t>’s, overlapped elements etc are not easy to find
  Catch 22 situation when tools that would help...
  won’t read in your broken XML

Advice:
  If starting from scratch on a new project:
    XMLmind + plug-in makes it very difficult to write broken XML
  If you are converting an old project:
    Use the validator (see tools):
      Does better error messages (although xml2rfc is much improved)
    Start with strict="no" when using xml2rfc
      See multiple warnings per run (instead of one fatal error)
Character Set and Entities

I-Ds and RFCs can only use basic US-ASCII

- No accented or extension (like ¥ or β) characters
- Only basic mathematical symbols

Only use limited set of entities which are predefined and can be rendered in US-ASCII

- &amp; &lt; &gt; &quot; &apos; &ndash; &mdash; &rsqb;

Two other entities help with formatting:

- &nbsp; Non-breaking space: lines will not be broken here
- &nbhy; Non-breaking hyphen: --- ditto ---

Some special entities:

- &rfc.number; The RFC number of this document
  (xxxxx while it is still an I-D)
Collected Hints from Earlier Slides

Case is significant in XML names
Remember to escape < (&lt;) and & (&amp;)
Tokens (names) mustn't include spaces ( _ - . are allowed)
You can't nest comments
Remember to change the version # in rfc:docName
t and list are mutually, but not directly, recursive
Use <vspace blankLines="100"/> to force new page
Figures and tables won't be numbered if no anchor
Use <vspace blankLines="1"/> to simulate paragraph breaks in list items
3 pages of hints on easy ways to do references!
Put PI's just after <rfc> to allow XMLmind to edit them
Extra Tips

More on hanging labels, indentation and lists
Handy way to switch between private & public draft
Using unpaginated text output
Undocumented spanx styles

Details on these at the back of the slide pack
Useful Links

xml2rfc home page: http://xml.resource.org/
Julian Reschke's XSLT Transformer:
    http://greenbytes.de/tech/webdav/rfc2629xslt/rfc2629xslt.html
A review page for XML editors: http://www.ivritype.com/xml/
XML Tutorial: http://www.w3schools.com/xml/default.asp
IETF Tools (rfcdiff, idnits, etc): http://tools.ietf.org/tools/
Other Documents

The xml2rfc website: http://xml.resource.org/

Provides:

2. Link to RFC2629 - The original specification of xml2rfc format
3. Link to ‘RFC2629bis’ - The ‘unofficial’ successor of RFC2629
4. Link to xml2rfc README file - how to drive the tool
5. The online web based xml2rfc converter
6. Links to download xml2rfc (current version)
7. Links to citation libraries
8. Some helpful hints
9. Link to a simple sample file (bigger one with this course)
10. Link to Julian Reschke’s XSL transformation tool
11. Link to the xml2rfc DTD
12. Link to the developers’ ‘bleeding edge’ next version snapshot
Acknowledg(e)ments*

For the original idea, 1st implementation, and ongoing drive:
  Marshall Rose (mrose at dbc.mtview.ca.us)

For current versions of xml2rfc tool:
  Charles Levert (charles.levert at gmail.com)

For XSLT transformer:
  Julian Reschke (julian.reschke at greenbytes.de)

For XMLmind plug-in and XML validator:
  Bill Fenner (fenner at gmail.com)

For IETF tools:
  Henrik Levkowetz (and others)

For Hints, Tips and Review:
  All the above plus Fred Baker, Frank Ellerman and Tony Hansen

* For explanation see the definition of rfcedstyle PI in v1.31 README
Questions?
Thank you!
Reference Section
Less Popular Middle Elements
External References

Very simple: **eref** element has a **target** attribute

*target* attribute MUST be a URI of an external ‘document’
xml2rfc generates ‘an appropriate designation’ again

Convenient for email addresses

```xml
<eref target='mailto:xml2rfc@lists.xml.resource.org' />
```

If the eref element is empty

The URI is rendered in the text where the eref is placed

```xml
<mailto:xml2rfc@lists.xml.resource.org>
```

If the eref element text is not empty

The text is rendered at the eref position plus a reference link

xml2rfc mailing list [1]

New reference section titled 'URIs' is created at end of document with

[1]  `<mailto:xml2rfc@lists.xml.resource.org>`

**HTML output will generate a hyperlink**

If **eref** has content this is the text of the hyperlink

**WARNING:** xml2rfc has some 'features' around eref: complains inappropriately.
Creating an Index

Insert `iref` elements at appropriate points in text

Attributes:

- **item**: Main heading or only index term
- **sub-item** (optional): sub-items sharing a common item heading are pulled together in the index
- **primary**: boolean - if true, item is emphasised in (HTML) index (its page number is in bold)

If there is one or more `iref`, Index is generated

Alphabetically sorted on **item** and then **sub-item**

Placed towards the end of the document - no control over where!

Has page number references - hyperlinked in HTML output

`iref` doesn’t put any text into the main body of doc
Comments

Reviewers can insert comments into a document

Comment is text in a `<cref>` element

Optional `source` attribute identifies reviewer

```
<cref source='Black Dog'>This is wrong!</cref>
```

Comments can be rendered, alternatively:

1. In a special section at the end of the document
   Cross-references are inserted at location of `<cref>`
2. Inline at the point the `<cref>` is placed

Controlled by processing directives
Hints about Formatting - 2

The `span` element can be used in most text

**style** attribute indicates how text should be rendered

No fixed set of styles: **emph**, **strong** and **verb** usually available

**NB:** Line breaks will not occur in `span` text

<table>
<thead>
<tr>
<th>XML</th>
<th>Text Output</th>
<th>HTML Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;span style=&quot;emph&quot;&gt; with emph &lt;/span&gt;</code></td>
<td><code>_ with emph _</code></td>
<td>`</td>
</tr>
<tr>
<td><code>&lt;span style=&quot;emph&quot;&gt;with emph&lt;/span&gt;</code></td>
<td><code>_with emph_</code></td>
<td>`</td>
</tr>
<tr>
<td><code>&lt;span style=&quot;strong&quot;&gt; with strong &lt;/span&gt;</code></td>
<td><code>* with strong *</code></td>
<td>`</td>
</tr>
<tr>
<td><code>&lt;span style=&quot;strong&quot;&gt;with strong&lt;/span&gt;</code></td>
<td><code>*with strong*</code></td>
<td>`</td>
</tr>
<tr>
<td><code>&lt;span style=&quot;verb&quot;&gt; with verb &lt;/span&gt;</code></td>
<td><code>&quot; with verb &quot;</code></td>
<td>`</td>
</tr>
<tr>
<td><code>&lt;span style=&quot;verb&quot;&gt;with verb&lt;/span&gt;</code></td>
<td><code>&quot;with verb&quot;</code></td>
<td>`</td>
</tr>
</tbody>
</table>

**emph** and **strong** give grades of emphasis

**verb** is intended for program code and sample input

Notice what happens to white space at the beginning and end of the `span` element text
Suggestions for XML Editors and other tools
XML Syntax-aware Editors
Recommended by various users - YMMV!

  Free - builds for most OS’s. Three XML major modes
  PSGML - http://www.lysator.liu.se/projects/about_psgml.html
  ttda - http://www.menteith.com/ttda/

jEdit - http://www.jedit.org/
  Free - Java based, so usable on Windows, Mac, Linux etc.
  JTidyPlugin will do validation and ‘tidying’ of XML

XMLSpy - http://www.altova.com/
  Home edition is free - $$$ for Professional & Enterprise editions
  Windows only

TextWrangler -
http://www.barebones.com/products/textwrangler/
  Free - also its paid-for big brother BBEdit
  Mac only

The IETF does not endorse any products.
You use them at your own risk.
XML Syntax-aware Editors – cont’d
Recommended by various users - YMMV!

- $$ - $$$ - Windows, Mac, Linux, Java for any OS

- Standard Edition is free - $$$ for Professional
- Bill Fenner's plugin makes this highly desirable

One pseudo-issue with ‘tool compatibility’:
Each editor has its own idea of what is ‘pretty printing’!
Makes diffs on source from different tools more or less hopeless

All trademarks acknowledged.
Check license terms before using any of these.

The IETF does not endorse any products. You use them at your own risk.
Alternative Tools

For HTML, PDF and other types of output:

Julian Reschke's XSLT transformation suite
- Many added features
- Elegant display
- Lots of hyperlinks

Useful for other things than just I-Ds and RFCs

See http://greenbytes.de/tech/webdav/rfc2629xslt/rfc2629xslt.html
Example Output from XSLT
Assorted Helpful Tools - 1

Validation of xml2rfc source:

Bill Fenner’s validator: http://rtg.ietf.org/~fenner/ietf/xml2rfc-valid/
This uses xmllint, part of Gnome libxml2: http://xmlsoft.org/
Essential when converting old text to xml2rfc

Many sophisticated editors won’t read in broken XML.
XMLmind insists on having almost right xml2rfc!

rfcdiff

Comparison of two textual I-Ds or RFCs: http://tools.ietf.org/tools/

htmltidy

Standalone pretty printer: http://tidy.sourceforge.net/
Useful when converting old text to xml2rfc
Assorted Helpful Tools - 2

**JavE**
Tool for drawing ASCII artwork: [http://www.jave.de](http://www.jave.de)

**Internet Explorer**
Will display colorized xml2rfc from a file using just DTD
- Need rfc2629.dtd, rfc2629-xthml.ent & rfc2629-other.ent
  in same directory
Built-in XSLT capabilities display files with Julian’s XSLT
- Need dtd files + rfc2629.xslt in same directory

**Mozilla Firefox**
Firefox 1.5.0.1 displays most of xml2rfc using XSLT
- Apparently there is a patch which would help
- Raw xml2rfc is not displayed usefully.
- It is fine with HTML and text generated by xml2rfc!
Processing Instructions Directory
File Inclusion/Rigor Control

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>include</td>
<td>n/a</td>
<td>Incorporate contents of file specified as value of parameter. Searches for file on search path specified in XML_LIBRARY environment variable, or in directory of containing file if XML_LIBRARY not defined.</td>
</tr>
<tr>
<td>strict</td>
<td>no</td>
<td>XML_LIBRARY not defined. Check more rigorously for adherence to the letter of the DTD law and compliance with idnits rules for I Ds. See next slide.</td>
</tr>
</tbody>
</table>
Rigor Control
The IETF Thought Police are coming!

Try to rigorously
  - enforce some ID-nits conventions
  - check accurate DTD validity

```xml
<?rfc strict="yes" ?> <!-- default "no" -->
```

Some of the things `strict="yes"` does:
- Validates the XML tree structure against the DTD
- Checks there is an Abstract & a Security Considerations section
- No eref or xref elements in Abstract
- No more than 5 authors
- Strictly limits line lengths
- Must have a ToC if more than 15 pages
- Problems which xml2rfc could workaround become fatal
## Rendering Control - 1

What gets output and how

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>topblock</td>
<td>yes</td>
<td>Put the famous header block on the first page. I-D/RFC identification on the left; authors and date on the right.</td>
</tr>
<tr>
<td>iprnotified</td>
<td>no</td>
<td>Needed to acknowledge if the IETF is notified of IPR encumbrances. Include boilerplate from Section 10.4(d) of RFC 2026 (Bradner, S., “The Internet Standards Process -- Revision 3,” October 1996.)</td>
</tr>
<tr>
<td>symrefs</td>
<td>no</td>
<td>Use anchors rather than numbers as tags for references</td>
</tr>
<tr>
<td>sortrefs</td>
<td>no</td>
<td>Sort references - this only has an effect if symrefs are used. Otherwise references are numbered as they appear in the source.</td>
</tr>
<tr>
<td>comments</td>
<td>no</td>
<td>Render &lt;cref&gt; comment information</td>
</tr>
<tr>
<td>inline</td>
<td>no</td>
<td>If comments is &quot;yes&quot;, then render comments inline at the &lt;cref&gt;; otherwise render them in an “Editorial Comments” section.</td>
</tr>
<tr>
<td>editing</td>
<td>no</td>
<td>Insert editing marks for ease of discussing draft versions. Editing marks are numeric labels on each paragraph of text. Applies to text output only - no effect on HTML output</td>
</tr>
</tbody>
</table>
## Rendering Control - 2

### What gets output and how

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>private</strong></td>
<td>&quot;&quot;&quot;&quot;</td>
<td>Produce a private memo rather than an RFC or Internet-Draft. The parameter is the title of the document.</td>
</tr>
<tr>
<td><strong>footer</strong></td>
<td>&quot;&quot;&quot;&quot;</td>
<td>Override the center footer string with the parameter.</td>
</tr>
<tr>
<td><strong>header</strong></td>
<td>&quot;&quot;&quot;&quot;</td>
<td>Override the leftmost header string with the parameter.</td>
</tr>
<tr>
<td><strong>rfcedstyle</strong></td>
<td>no</td>
<td>Attempt to closely follow finer details from the latest observable RFC-Editor style so as to minimize the probability of being sent back corrections after submission. This directive is a kludge whose exact behavior is likely to change on a regular basis to match the current flavor of the month; The README file has more details of current effects.</td>
</tr>
<tr>
<td><strong>rfcprocack</strong></td>
<td>no</td>
<td>If there already is an automatically generated Acknowledg(e) ment section, pluralize its title and add a short sentence acknowledging that xml2rfc was used in the document's production to process an input XML source file in RFC 2629 format.</td>
</tr>
</tbody>
</table>
# Table of Contents Control

What is in it - if anything - & format

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>toc</td>
<td>no</td>
<td>If &quot;yes&quot;, generate a Table-of-Contents.</td>
</tr>
<tr>
<td>toccappendix</td>
<td>yes</td>
<td>Control whether the word “Appendix” appears in the Table-of-Contents entries for the relevant sections.</td>
</tr>
<tr>
<td>tocdepth</td>
<td>3</td>
<td>if toc is &quot;yes&quot;, then this determines the ‘depth’ of the Table-of-Contents, i.e., the number of levels of sub-sections that have entries in the Table-of-Contents.</td>
</tr>
<tr>
<td>tocindent</td>
<td>yes</td>
<td>if toc is &quot;yes&quot;, then setting this to &quot;yes&quot; will indent entries for subsections in the Table-of-Contents.</td>
</tr>
<tr>
<td>tocompact</td>
<td>yes</td>
<td>if toc is &quot;yes&quot;, then setting this to &quot;no&quot; will make the Table-of-Contents a little less compact. Typically this involves inserting blank lines at the end of the entries for a top level section and its sub-sections.</td>
</tr>
</tbody>
</table>
Format Control
Details of layout in text and nroff output

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>colonspace</td>
<td>no</td>
<td>Put two spaces instead of one after each colon (&quot;:&quot;) in txt or nroff files.</td>
</tr>
<tr>
<td>autobreaks</td>
<td>yes</td>
<td>Automatically force page breaks to avoid widows and orphans (not perfect).</td>
</tr>
<tr>
<td>compact</td>
<td>no</td>
<td>From v1.31 (rfcedstyle) When producing a txt/nroff file, try to conserve vertical whitespace (the default value was &quot;no&quot; up to v1.30; from v1.31 the default is the current value of the rfcedstyle PI).</td>
</tr>
<tr>
<td>subcompact</td>
<td>(compact)</td>
<td>If compact is &quot;yes&quot;, then you can make things a little less compact by setting this to &quot;no&quot; (the default value is the current value of the compact PI).</td>
</tr>
<tr>
<td>needLines</td>
<td>n/a</td>
<td>An integer hint indicating how many contiguous lines are needed at this point in the output. This PI can appear anywhere in the source file.</td>
</tr>
</tbody>
</table>

**If compact="no":**

- Top level sections start on a new page
- A blank line is forced between list items
# HTML Specials

Things to do differently when doing HTML output

These PIs only have an affect on HTML output
Text and nroff output are unaffected.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Default</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>&quot;&quot;</td>
<td>When producing a html file, use the image in the file specified in the parameter.</td>
</tr>
<tr>
<td>emoticonic</td>
<td>no</td>
<td>Automatically replaces input sequences such as</td>
</tr>
<tr>
<td>linkmailto</td>
<td>yes</td>
<td>Generate mailto: URL, as appropriate.</td>
</tr>
<tr>
<td>slides</td>
<td>no</td>
<td>When producing a html file, produce multiple files for a slide show</td>
</tr>
<tr>
<td>useobject</td>
<td>no</td>
<td>When producing a html file, use the &lt;object&gt; html element with inner replacement content instead of the &lt;img&gt; html element, when a source xml element includes an src attribute.</td>
</tr>
</tbody>
</table>
Debugging Assistance

Two PIs useful for locating problems

Both can be placed anywhere in the file

**linefile**

A way to override xml2rfc’s reckoning of the current input position as used for warning & error reporting purposes.

   cf. #line and #file macros in C (header) files.

The change takes effect right after this PI.

Value: a string such as “35:file.xml” which changes both line number and file name or just “35” which changes the line number but leaves the file name as the containing file’s real name or whatever the previous `linefile` PI set it to.

**typeout**

Applies only in processing pass 2.

Print the PI value to standard output at that point in processing.
Additional Useful Tips
More on Hanging Labels in Lists

short  With a label shorter than the `hangIndent` there is white space after the label and before the item text starts although it starts on the same line - clearly separating the label from the column of items.

longer_label  With a label longer than the `hangIndent` the label runs on into the text item and the separation is lost.

vspace_trick  

Inserting a `<vspace />` at the start of the item forces the new item to start on a new line emphasizing the separation again.

```xml
<list hangIndent="6" style="hanging">
  <t hangText="short">With a label shorter than the `hangIndent` there is white space after the label and before the item text starts although it starts on the same line - clearly separating the label from the column of items.</t>
  <t hangText="longer">With a label longer than the `hangIndent` the label runs on into the text item and the separation is lost.</t>
  <t hangText="vspace_trick"><vspace blankLines="0" /></t>Inserting a `<vspace />` at the start of the item forces the new item to start on a new line emphasizing the separation again.</t>
</list>
```
Quick Public-Private Switch

The difference between a private and a real draft...
can be only two characters:

<?rfc private="Creative Commons License:
  ATTRIBUTIONS + SHAREALIKE" ?>
<?rfc header="Interim draft" ?>
<?rfc footer="draft-update-manually-for private-00" ?>

<?rfc private="Creative Commons License:
  ATTRIBUTIONS + SHAREALIKE" ?>
<?rfc header="Interim draft" ?>
<?rfc footer="draft-update-manually-for private-00" ?>

This is a single comment...
PIs are NOT interpreted!

Here there are two comments and three PIs
that ARE interpreted.
Unpaginated Output

Unpaginated text output is
a useful way to compare output to get diffs for editing, and
useful for reading into a word processor for reading & making comments

Controlled by
command line option (-unpg),
file name extension (.unpg) on GUI, or
radio button on web service

Undocumented spanx Styles

Code inspection yields extra styles
vbare - vemph - vstrong - vdeluxe:
Fixed width fount forms - plain/italic/bold/italic-bold
nobreak: Normal rendering but no line break allowed in text