Writing RFCs and Internet-Drafts in markdown and a bit of YAML

Writing RFCs

Traditional: `nroff`

- No automation, not even a TOC

WYSIWYG: Word Template

- YNGWYT: You never get what you think

XML2RFC

- Now we need an XML editor
  and get lots of pointy-bracket noise

Why we like text formats

- Enables automation
  - automatic generation
  - consistency checking
- Enables collaboration
  - Use programmers' tools:
    SVN, git, github
- Enables evolution

Version Control

Distraction-free writing

How to focus on the content?

Get noise out of the way!

In summary, this specification adds a pair of Block options to CoAP that can be used for block-wise transfers. Benefits of using these options include:

- Transfers larger than what can be accommodated in constrained-network link-layer packets can be performed in smaller blocks.

- No hard-to-manage conversation state is created at the adaptation layer or IP layer for fragmentation.

- The transfer of each block is acknowledged, enabling individual retransmission if required.

- Both sides have a say in the block size that actually will be used.

- The resulting exchanges are easy to understand using packet analyzer tools and thus quite accessible to debugging.

- If needed, the Block options can also be used (without changes) to provide random access to power-of-two sized blocks within a resource representation.

In summary, this specification adds a pair of Block options to CoAP that can be used for block-wise transfers. Benefits of using these options include:

* Transfers larger than what can be accommodated in constrained-network link-layer packets can be performed in smaller blocks.
* No hard-to-manage conversation state is created at the adaptation layer or IP layer for fragmentation.
* The transfer of each block is acknowledged, enabling individual retransmission if required.
* Both sides have a say in the block size that actually will be used.
* The resulting exchanges are easy to understand using packet analyzer tools and thus quite accessible to debugging.
* If needed, the Block options can also be used (without changes) to provide random access to power-of-two sized blocks within a resource representation.

“Rich Text” Humane Markup

Many proposals.

Asciidoc Creole MediaWiki Org-mode POD reStructuredText Textile.

- Markdown.

The markdown **ecosystem**

Created by John Gruber and Aaron Swartz in 2004.

Abandoned by John Gruber right there.

Organic growth since; a few 800 lb gorillas.

No official specification (apart from the obsolete one from 2004).

Markdown in 5 seconds

Just write.

Markdown in 5 minutes

How to apply formatting with markdown?

```
# chapter head
## section head
### subsection head
#### subsubsection head

*italic text*
**bold text**
***italics and bold together***
```

```
> blockquote text

* bulleted list item

1. ordered list item

[link label](http://www.example.com/)

| Table | Head  |
| cell1 | cell2 |

~~~~
code block (or just indent by ≥ 4)
~~~~
```

What Editor?

Many to choose from.

Classical programmers' editor: emacs, vim.

Modern programmers' editor: Sublime, Atom.

- Markdown editor?

Mashable SOCIAL MEDIA TECH BUSINESS ENTERTAINMENT WORLD LIFESTYLE WATERCOOLER MORE

78 Tools for Writing and Previewing Markdown

2.2k SHARES

Share on Facebook Share on Twitter +

Waiting for tap.rubiconproject.com

MacDown

The open source Markdown editor for OS X.

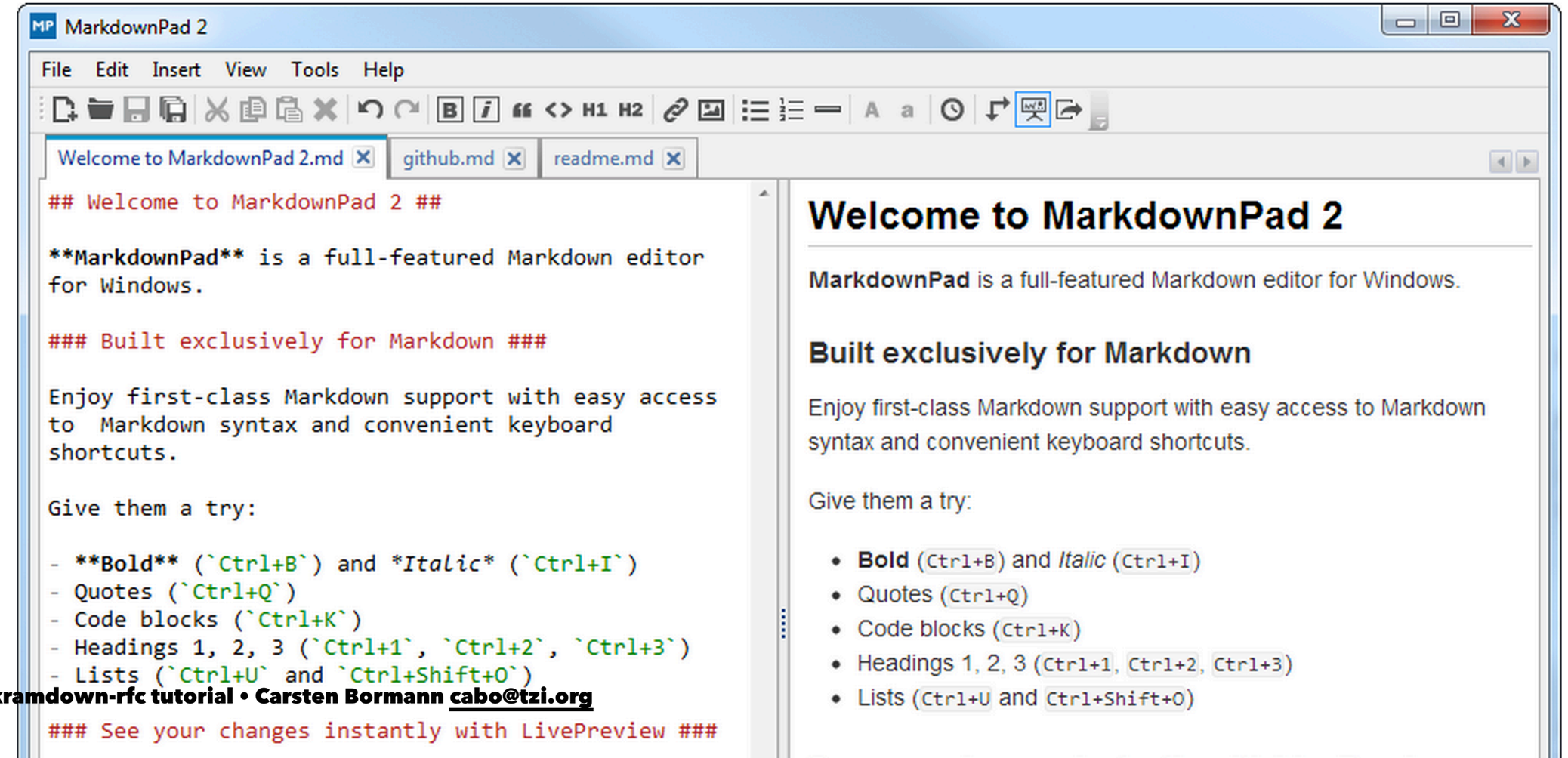
What is MacDown?

MacDown is an open source Markdown editor for OS X, released under the MIT License. It is heavily influenced by Chen Luo's Mou. This is how it looks:

MarkdownPad

MarkdownPad is a full-featured Markdown editor for Windows.

> **New: Introducing MarkdownPad 2!**



RFC-specific markup

Cross References (sections, tables, figures)

Literature References (normative, informative)

Captions, Tables, Figures, Artwork, ...

Some invention required.

Markdown Tools for RFCs

... differ in their inventions

pandoc-rfc

by Miek Gieben, documented in RFC 7328.
Miek is currently preparing a successor.

kramdown-rfc2629

Simple **things:**

easy

Complicated things:

possible

How kramdown-rfc2629 happened

Needed to write 3 I-Ds. What is faster?

[ ] Write them in XML

[x] Write a tool, then write them in markdown

Based on popular markdown parser **kramdown**.
Format stable since ~ 2010; tool published on 2010-06-09, gemified 2011-01-02.
33 Gem version updates so far; current: 1.0.24

https://rubygems.org/gems/kramdown-rfc2629/

TOTAL DOWNLOADS

20,423

FOR THIS VERSION

239

.

.

What?

.

.

(mnot in rfcformat WG:
"this one makes it
almost too easy")

.

.

More realistically:
Penetration in IETF ~ 4 %

Getting it installed

```
gem install kramdown-rfc2629
```

- may need to add **`sudo`** in front.

Also may want to install **`xml2rfc`** (version 2).

(If your OS is ancient: May need to add Ruby;
2.2 recommended, 1.9+ works.)

Don't forget to occasionally:

```
gem update
```

Using it

```
kramdown-rfc2629 mydraft.mkd >mydraft.xml
xml2rfc mydraft.xml
```

This is usually hidden in a Makefile, Gruntfile, ..., so you say something like:

```
make mydraft.txt
```

or

```
make mydraft.html
```

Right out of the core SVN:

```
OPEN=$(word 1, $(wildcard /usr/bin/xdg-open /usr/bin/open /bin/echo))
SOURCES?=${wildcard *.mkd}
DRAFTS=${SOURCES:.mkd=.txt}
HTML=${SOURCES:.mkd=.html}

all:    $(DRAFTS)
html:   $(HTML)

%.xml:  %.mkd
    kramdown-rfc2629 $< >$@.new
    mv $@.new $@

%.html: %.xml
    xml2rfc --html $<
    $(OPEN) $@

%.txt:  %.xml
    xml2rfc  $< $@
```

Overall structure of a draft

```
---
Frontmatter (YAML)

--- abstract

...abstract...

--- middle

...sections...

--- back

...appendices...
```

YAML???

Structured information in an RFC

```
---
title: Block-wise transfers in CoAP
docname: draft-ietf-core-block-17
date: 2015-03-09

stand_alone: true

ipr: trust200902
area: Applications
wg: CoRE Working Group
kw: Internet-Draft
cat: std

coding: UTF-8
pi: [toc, sortrefs, symrefs]

author:
      -
        ins: C. Bormann
        name: Carsten Bormann
        org: Universität Bremen TZI
        street: Postfach 330440
        city: Bremen
        code: D-28359
        country: Germany
        phone: +49-421-218-63921
        email: cabo@tzi.org
      -
        ins: Z. Shelby
        name: Zach Shelby
        org: ARM
        role: editor
        street: 150 Rose Orchard
        city: San Jose, CA
        code: 95134
        country: USA
        phone: +1-408-203-9434
        email: zach.shelby@arm.com

normative:
  RFC2119:
  RFC7252: coap
  I-D.ietf-core-observe: observe

informative:
  RFC7230: http
  RFC4919:   # 6lowpan
  REST:
    target: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
    title: Architectural Styles and the Design of Network-based Software Architectures
    author:
        ins: R. Fielding
        name: Roy Thomas Fielding
        org: University of California, Irvine
    date: 2000
    seriesinfo:
      "Ph.D.": "Dissertation, University of California, Irvine"
    format:
      PDF: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
  RFC6690: link
```

Actual front matter

```
---
title: Block-wise transfers in CoAP
docname: draft-ietf-core-block-17
date: 2015-03-09

ipr: trust200902
area: Applications
wg: CoRE Working Group
kw: Internet-Draft
cat: std

author:
      -
        ins: C. Bormann
...
```

Author information

```
author:
      -
        ins: C. Bormann
        name: Carsten Bormann
        org: Universität Bremen TZI
        street: Postfach 330440
        city: Bremen
        code: D-28359
        country: Germany
        phone: +49-421-218-63921
        email: cabo@tzi.org
      -
        ins: Z. Shelby
        name: Zach Shelby
        org: ARM
        role: editor
        street: 150 Rose Orchard
        city: San Jose, CA
        code: 95134
        country: USA
        phone: +1-408-203-9434
        email: zach.shelby@arm.com
```

References

```
normative:
  RFC2119:
  RFC7252: coap
  I-D.ietf-core-observe: observe

informative:
  RFC7230: http
  RFC4919:   # 6lowpan
  REST:
    target: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
    title: Architectural Styles and the Design of Network-based Software Architectures
    author:
        ins: R. Fielding
        name: Roy Thomas Fielding
        org: University of California, Irvine
    date: 2000
    seriesinfo:
      "Ph.D.": "Dissertation, University of California, Irvine"
    format:
      PDF: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
  RFC6690: link
```

Processing parameters

```
---
stand_alone: true

coding: UTF-8
pi: [toc, sortrefs, symrefs]
```

PI (processing instructions) can be more detailed:

```
pi:
  toc: yes
  tocdepth: 4
  sortrefs: yes
  symrefs: yes
```

Best start with a skeleton

```
---
coding: utf-8

title: Many fine lunches and dinners
abbrev: mfld
docname: draft-mfld-00
category: info

stand_alone: yes
pi: [toc, sortrefs, symrefs, comments]

author:
  -
    ins: C. Bormann
    name: Carsten Bormann
    org: Universität Bremen TZI
    street: Postfach 330440
    city: Bremen
    code: D-28359
    country: Germany
    phone: +49-421-218-63921
    email: cabo@tzi.org

--- abstract

insert abstract here

--- middle

# Introduction

This...
```

What you need to know about YAML

YAML = JSON superset (you can write everything in JSON if that makes you happier)

YAML proper:

- Use a colon for name/value separation
    - Or a "-" for arrays (where the entries have no name)
- Use indentation for subhashes/subarrays

Transparency in YAML

Protect text from interpretation using quotes:

```
        phone:  "+358407796297"
...
        title: "6LoWPAN: the Wireless Embedded Internet"
```

Convenience syntax for long titles:

```
title: >
  Information Technology – ASN.1 encoding rules:
  Specification of Basic Encoding Rules (BER), Canonical Encoding
  Rules (CER) and Distinguished Encoding Rules (DER)
```

Basic markdown

- Span features (inside a paragraph):
  e.g., italics, cross-references
- Block features:
  e.g., figures/artwork, tables, blockquotes
- Attributes: When the defaults are not enough

Ending blocks

Blocks are usually separated by a blank line

Line breaks don't matter, unless line ends in

- two spaces [NOT RECOMMENDED]
- `\\` (kramdown feature)

```
markdown RFC\\
with distraction-free writing\\
liberates your mind
```

Span features

In plain-text RFCs, we don't need those a lot...

```
*italic text*        **bold text**
***italics and bold together***

_italic text_        __bold text__
**_italics and bold together_**
```

Only '*' works **inside** words.

Use '\' for escaping:

```
3\*6 == 6\*3
```

Special characters

" and " becomes “ and ”

-- becomes – (en-dash)
--- becomes — (em-dash)

Most of this is immediately reverted by XML2RFC in plaintext mode, but it does look better in HTML.

Code spans

```
Write `code` within a line
```

Write **`code`** within a line

```
Write ``code with ` backquotes ` this way``
```

Write **`code with ` backquotes ` this way`**

(XML2RFCv2 turns these to double quoted spans in the plain text form.)

Inline links

Links:

```
[link text](link target)
```

Textless internal links replace `[](#RFC7252)`

```
{{RFC7252}}
```

Unordered lists

```
* Unnumbered list
- with *, +, -
+ mix as you want
```

- Unnumbered list
- with *, +, -
- mix as you want

Ordered lists

```
0. Numbered list
8. with any number
8. mix as you want
```

1. Numbered list
2. with any number
3. mix as you want

Nested lists

Just indent (4 spaces):

```
0. Numbered list
8. with any number
8. mix as you want
    * Unnumbered list
    - with *, +, -
    + mix as you want
3. and go on
```

(kramdown doesn't require 4 spaces, but other tools may)

1. Numbered list
2. with any number
3. mix as you want
    - Unnumbered list
    - with *, +, -
    - mix as you want
4. and go on

Definition lists

```
IP
: The Internet Protocol is defined
  in {{RFC0791}}.

TCP
: The Transmission Control Protocol
  is defined in {{RFC0793}} with
  many changes since; work is
  underway to get out a new
  definition that merges all these.
```

XML2RFC does **not** break after the term for plain text!

IP

: The Internet Protocol is defined in [RFC0791].

TCP

: The Transmission Control Protocol is defined in [RFC0793] with many changes since; work is underway to get out a new definition that merges all these.

Tables

sequence of lines with pipes:

```
| Code | Meaning |
| 2.05 | Content |
| 2.01 | Created |
| 2.04 | Changed |
```

First line is heading. Can make this more evident:

```
| Code | Meaning |
|------+---------|
| 2.05 | Content |
| 2.01 | Created |
| 2.04 | Changed |
```

Code	Meaning
2.05	Content
2.01	Created
2.04	Changed

Code blocks (ASCII Art)

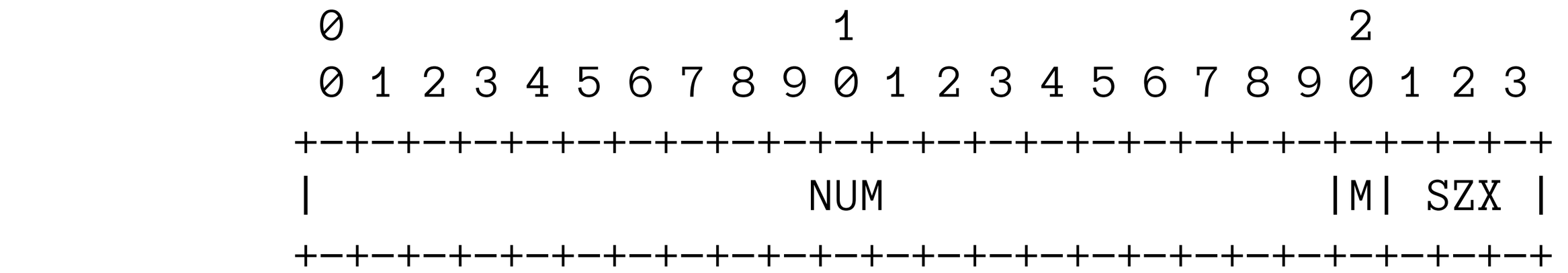
~~~~
   +Ub       +Ub
    |         |   sink 1mA
   100k       |     v
    |   LM358 |     |
    +--------|+\    |
    |        |  >--|I  2N7000
    |     +--|-/   |S  BS107
    |     |  |     |
    |     +---(-----+
----------  |     |
|LM385-1.2|---+   1k24
~~~~~~~~~~  |     |
    |       |     |
    +-------+-----+-- GND
~~~~

1. "fenced" by ~~~~ lines (above; **not** ```), or:

2. Just indented by 4+ spaces
~~~~

Attribute lists

Attach to the object being controlled:



```
{: #block title="Block option value"}
```

IAL **(Inline Attribute Lists)**

```
{: #block title="Block option value"}
```

- enclosed in **{: }**
- **#block** is a reference label ➔ usage: **{{block}}**
- **attrname="attrvalue"** is any other attribute
  - usually XML attributes (some KD specific ones)

IAL can come before or after the controlled object

ALD **(Attribute Definition Lists)**

Convenience mechanism for not having to write the same attributes again and again:

```
{:req: counter="bar" style="format R(%d)"}
```

Use ALD by just citing its name in an IAL:

```
{: req}
* Foo
* Bar
* Bax
```

Can mix ALD references with other attributes.

Advanced tables

```
| No. | C | U | N | R | Name   | Format | Length | Default |
|-----+---+---+---+---+--------+--------+--------+---------|
|  23 | C | U | - | - | Block2 | uint   |    0-3 | (none)  |
|  27 | C | U | - | - | Block1 | uint   |    0-3 | (none)  |
{: #block-option-numbers title="Block Option Numbers"
   cols="r l l l l l r l"}
```

kramdown-rfc invention: `cols` attribute.

Space-separated per column.

- l r c for alignment
- can prefix with numbers (in em) or percentage:
  cols="r 22c l" or cols="50%r 25%c l"

Cross-references

Given a table, figure, or section with an anchor:

```
| HTTP | CoAP |
| 200  | 2.05 |
{: #code-mapping}
```

We can reference this as:

```
The mapping is defined
in {{code-mapping}}.
```

HTTP	CoAP
200	2.05

The mapping is defined in Table 1.

Referencing references

References actually are internal cross-references to the References section.

```
See {{RFC7252}} for details.
```

points to the References entry created by, say:

```
informative:
  RFC7252:
```

in the YAML. Same for **I-D.ietf-core-observe**.

Referencing references for the forgetful

Labels are usually visible. Can use an internal label, prefixed by a hyphen:

```
See {{-coap}} for details.
```

Indicate this internal label in the References entry:

```
informative:
  RFC7252: coap
```

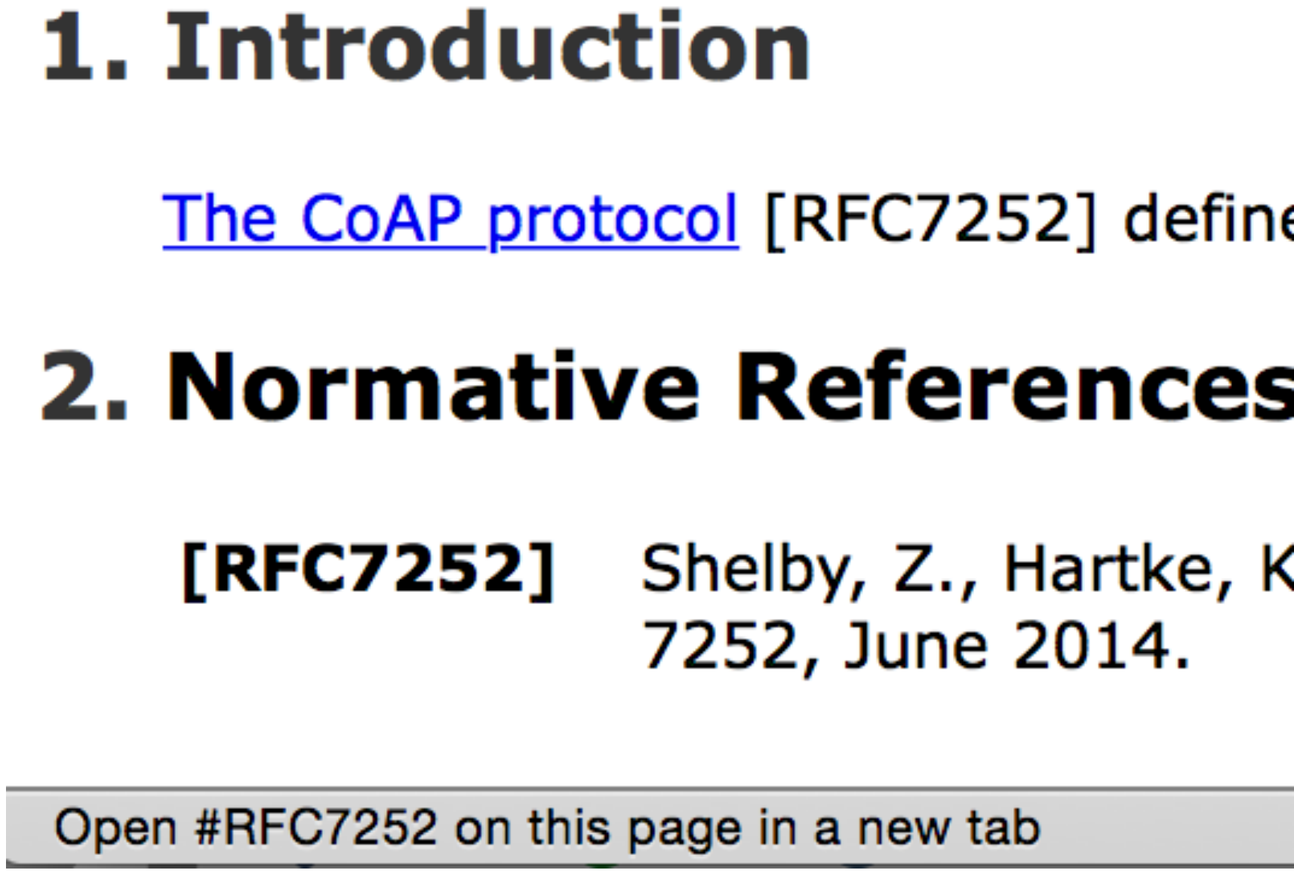
in the YAML. Same for **I-D.ietf-core-observe**.

Link contents

```
normative:
  RFC7252:
                  ...
[The CoAP protocol](#RFC7252) defines
the details.
```

Note the reference to a local fragment identifier!
(The link takes you to the References section.)

Any such link to an external URI creates the **URIs** section.



Autoreferencing

```
See {{?RFC7252}} for details.
```

automatically adds the reference as informative. Similarly,

```
See {{!RFC7252}} for the definition.
```

automatically adds it as normative.

(Does not work with link contents syntax.)

Labels for sections

```
# IANA Considerations {#iana}
                                        <!-- is the same as: -->
{: #iana}
# IANA Considerations
```

Section titles are also automatically made into labels -- lower case with hyphens will just work:

```
{{iana-considerations}}
```

- so no need for the above override, but...
  the label then of course changes with the title.

Real-world examples

```
normative:
  RFC2119:
  RFC7252: coap
  I-D.ietf-core-observe: observe
informative:
  RFC7230: http
                ....
The work on Constrained RESTful Environments (CoRE)
aims at realizing the REST architecture in a suitable
form for the most constrained nodes (such as
microcontrollers with limited RAM and ROM {{?RFC7228}})
and networks (such as 6LoWPAN, {{?RFC4944}}) {{-coap}}.
The CoAP protocol is intended to provide RESTful
{{REST}} services not unlike HTTP {{-http}}, while ...
```

Automatically include program-generated content

From the source markdown of RFC7400:

```
~~~~
{::include ../ghc/packets-new/p4.out}
~~~~
{: #example2 title="A longer RPL example"}
```

(You still have to create your own build environment to create and update these files.)

Things you probably don't need

... but are there when you do:

- Index entries ("irefs"), including auto-indexing
- Editorial comments ("crefs"; use markdown footnotes and `pi: [comment]`)

Things you'll probably need
... but aren't quite there yet:

- An upconverter (XML to markdown). Instead, send me your XML, and I'll use my experimental hack and fix that in the process.
- Colspans/Rowspans in tables. There isn't really a markdown way to do those...

Packaged workflows

TO DO

More information

https://github.com/cabo/kramdown-rfc2629

OK, OK:

http://rfc.space (RFCs from outer space?)

draft-bormann-rfc-markdown ...soon...