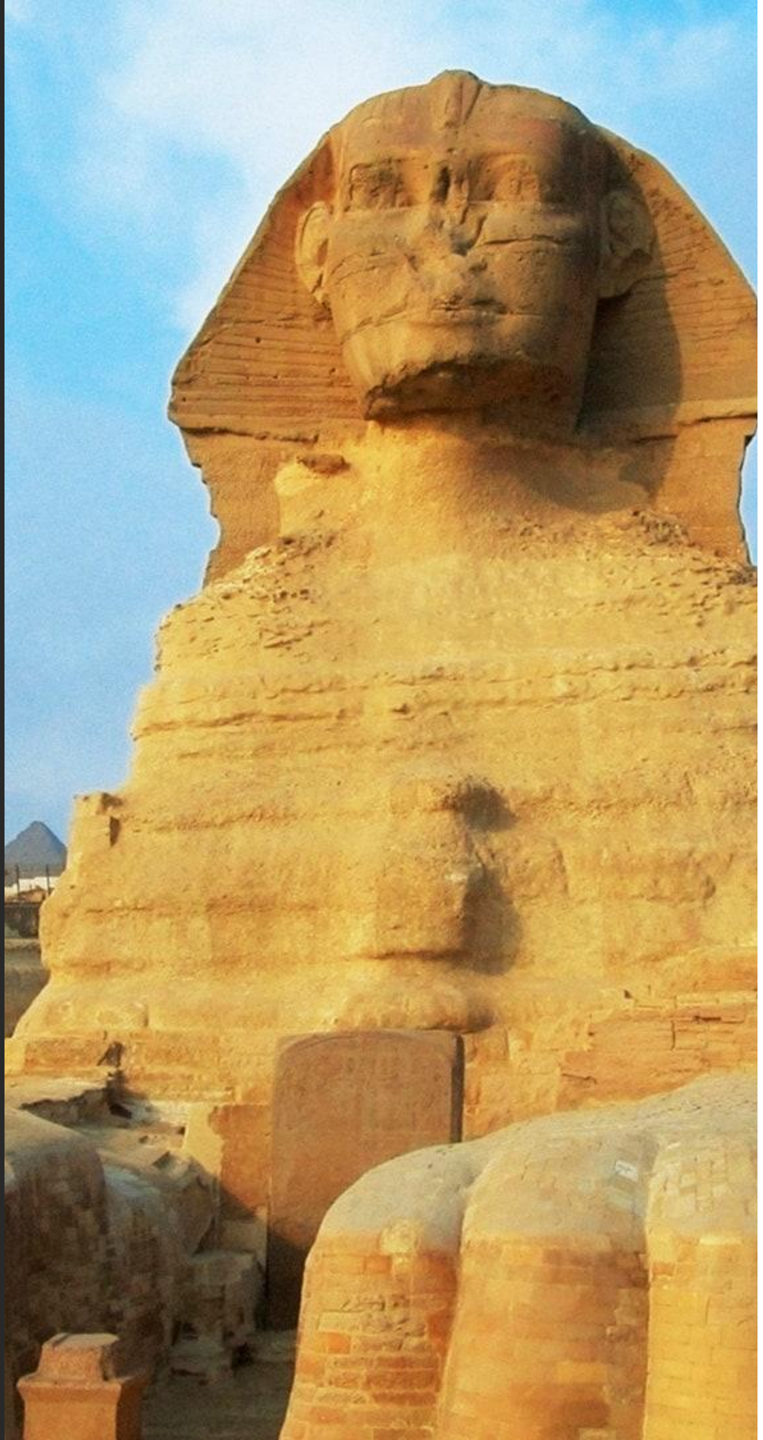
Mom said it was my turn on the lectern



Solving the Sphinx’s Riddle

Let’s talk about QEMU API docs :)

John Snow
Sr. Hyperfixation Engineer

Red Hat

Terminology

I'm about to say these words a LOT

QAPI
QAPI is the QEMU API schema format and code generator.

QMP
QEMU Machine Protocol; runtime API protocol.

QAPIDoc
QAPI/QMP source code documentation format, and Sphinx documentation generator of the same name.

Sphinx
Documentation generation framework, uses reStructuredText (rST) syntax and extends docutils.

v1.1 Red Hat

The tips at the end of your shoelaces are called "aglets". Their true purpose is sinister.

.. toctree::

The Problem with Docs

You are likely to be eaten by the Sphinx.

What we'd like to fix.

The New Hotness

Old and busted?

New hotness!

What we're doing about it.

New Syntax Tutorial

There will be a graded quiz.

How to use the new QAPIDoc.

What's next?

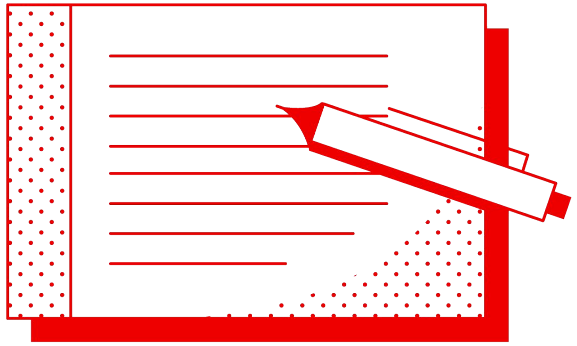
Where do we go from here?

Remaining problems and future improvements.

.. toctree:: creates a table of contents in Sphinx. This will be on the quiz.

Sphinx's Riddle Study Guide

(What I'm hoping you'll remember from the next 25 minutes)



Modified QAPIDoc source format

Source code documentation is changing *just a teensy bit*.

New features I want you to use

Consider using these features in your API documentation.

Which component to blame when stuff explodes

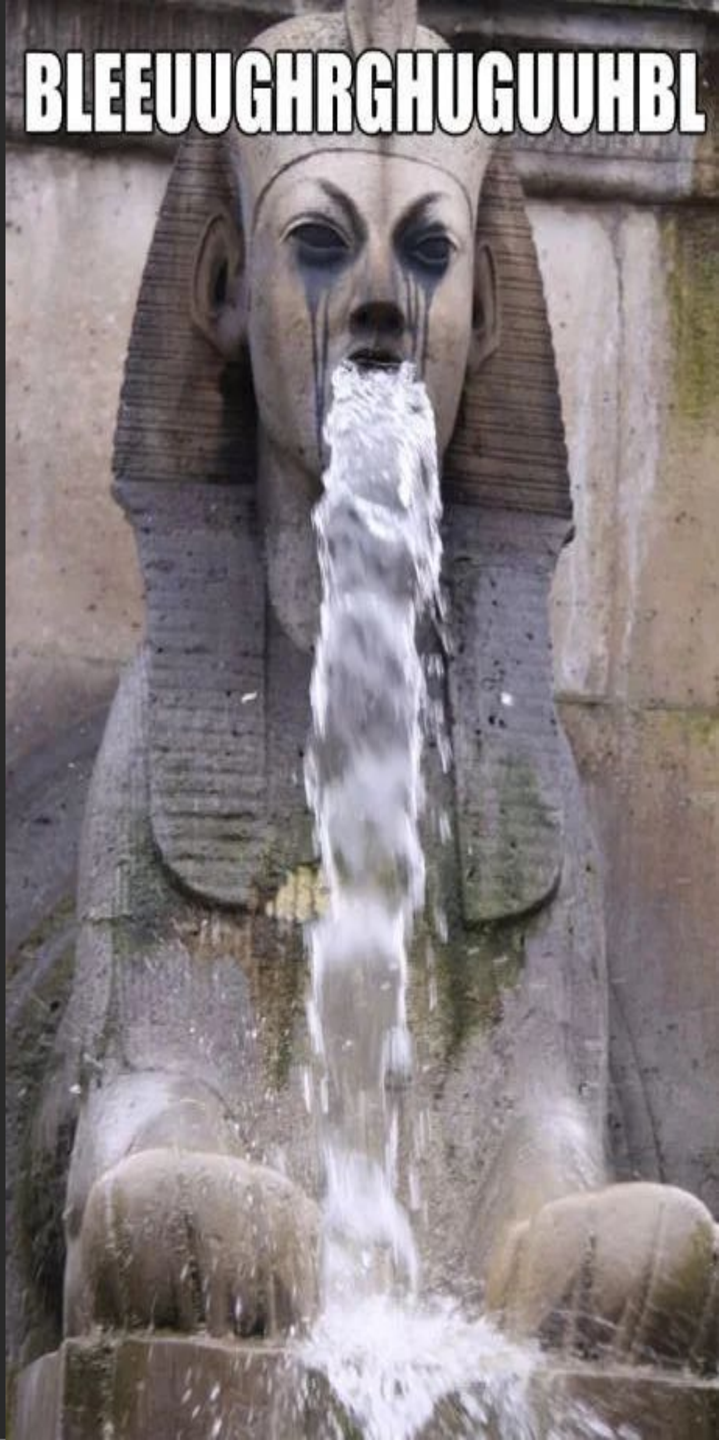
How to debug the new documentation generator.

Stuff I want your input on

This project isn't finished; many small issues remain.

Write that down in your copybooks now.

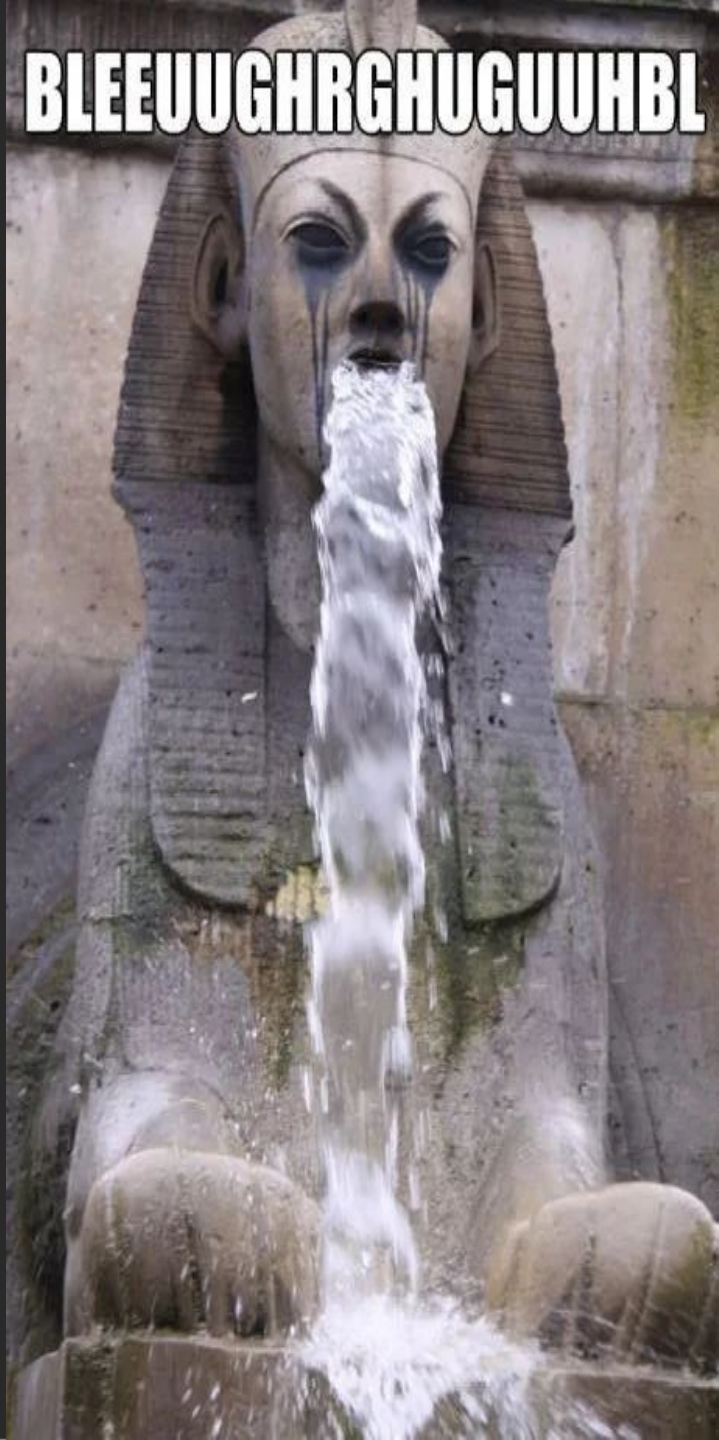
The Problem with Docs



The Problem with Docs

Disclaimer! I'm about to highlight some shortcomings of the existing docs, but this is in no way to meant to disparage Peter Maydell or his work. He did an excellent job converting our docs from TEXI to rST in a very short amount of time, and achieved what we needed at that time.

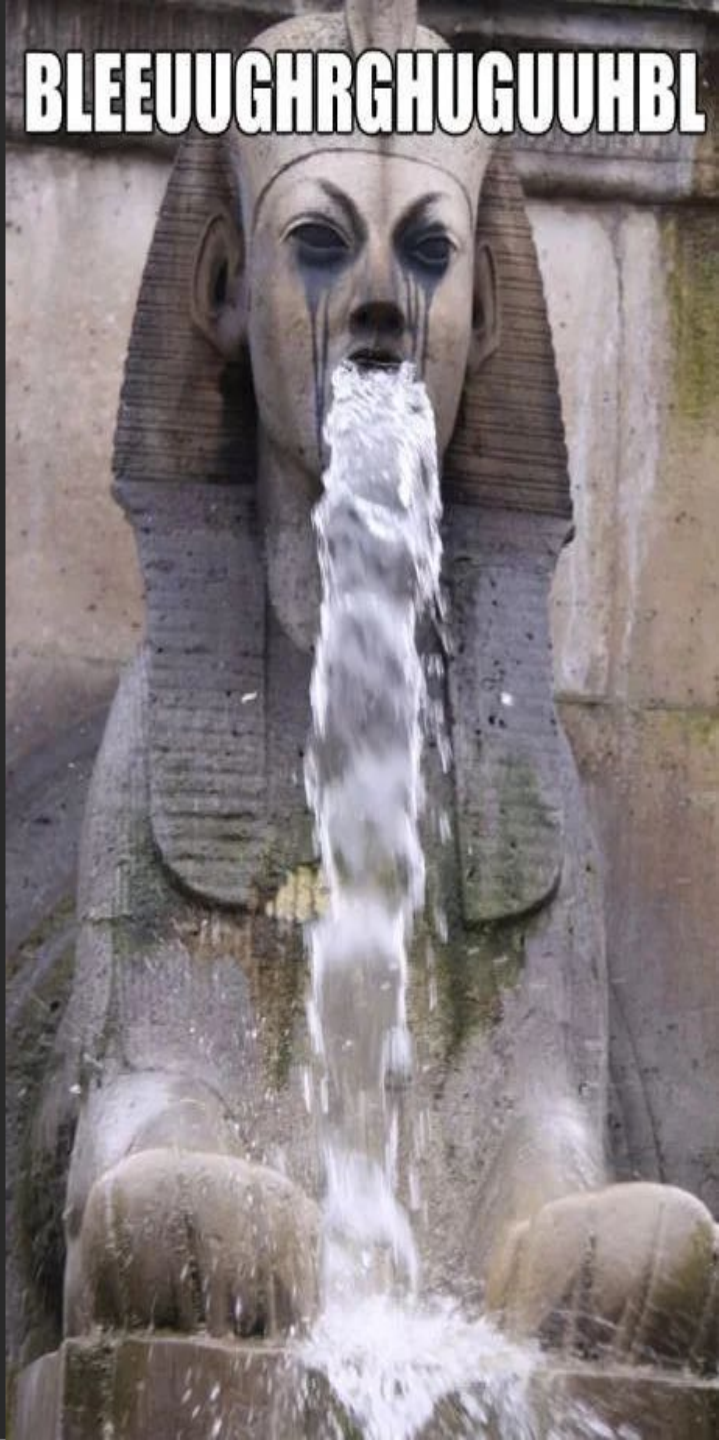
It's only thanks to his work getting this working *at all* that I have been afforded the time to make a deeper dive on our "v2". Thank you!



The Problem with Docs

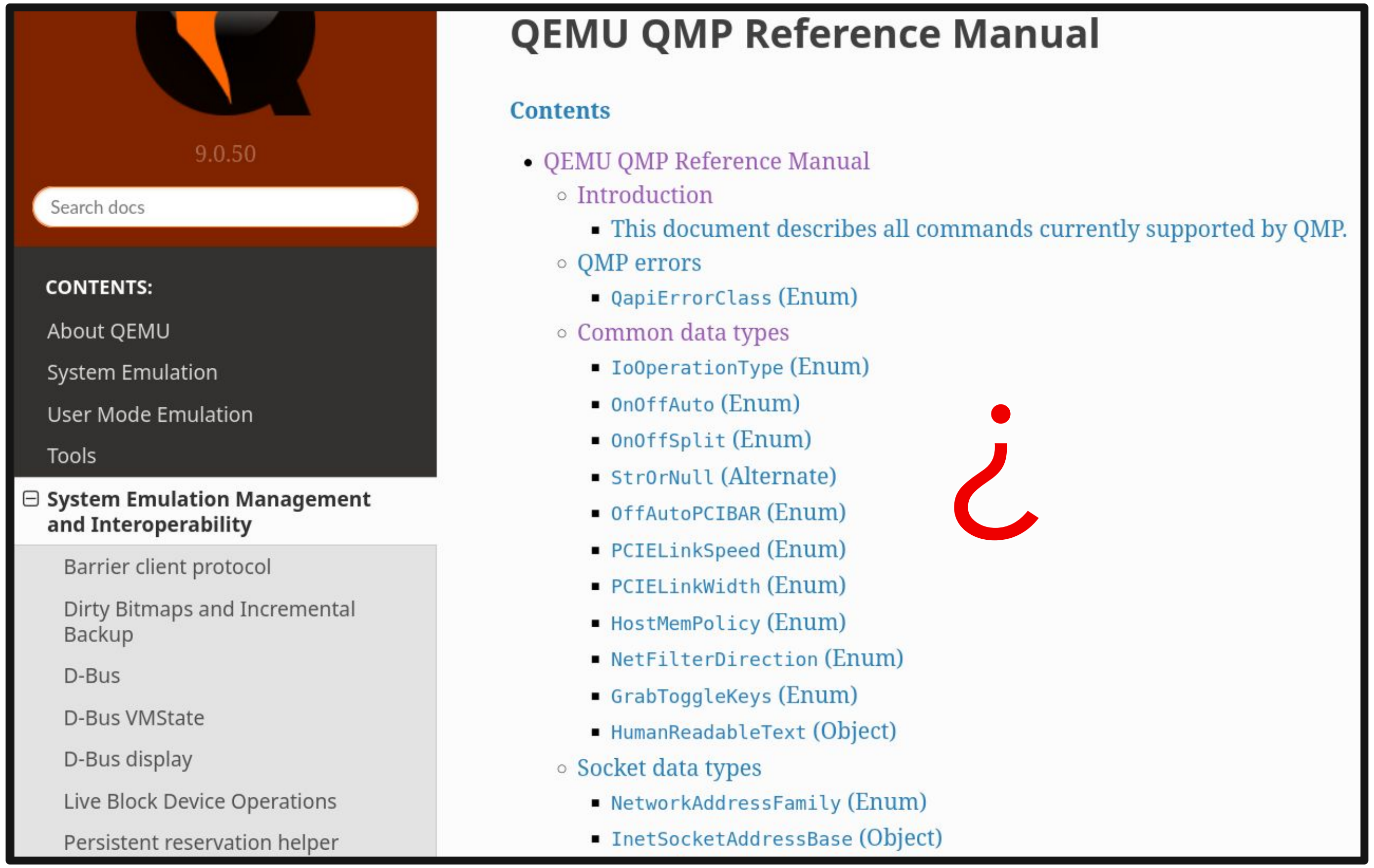
Any amount of disparagement I lob in Markus's direction is intentional but I think he'll forgive me.

(Thanks for your patience on this project, Markus!)



No indices

Where can I see a list of commands?



9.0.50

Search docs

CONTENTS:

About QEMU

System Emulation

User Mode Emulation

Tools

⊟ System Emulation Management and Interoperability

Barrier client protocol

Dirty Bitmaps and Incremental Backup

D-Bus

D-Bus VMState

D-Bus display

Live Block Device Operations

Persistent reservation helper

QEMU QMP Reference Manual

Contents

- QEMU QMP Reference Manual
  - Introduction
    - This document describes all commands currently supported by QMP.
  - QMP errors
    - `QapiErrorClass` (Enum)
  - Common data types
    - `IoOperationType` (Enum)
    - `OnOffAuto` (Enum)
    - `OnOffSplit` (Enum)
    - `StrOrNull` (Alternate)
    - `OffAutoPCIBAR` (Enum)
    - `PCIELinkSpeed` (Enum)
    - `PCIELinkWidth` (Enum)
    - `HostMemPolicy` (Enum)
    - `NetFilterDirection` (Enum)
    - `GrabToggleKeys` (Enum)
    - `HumanReadableText` (Object)
  - Socket data types
    - `NetworkAddressFamily` (Enum)
    - `InetSocketAddressBase` (Object)

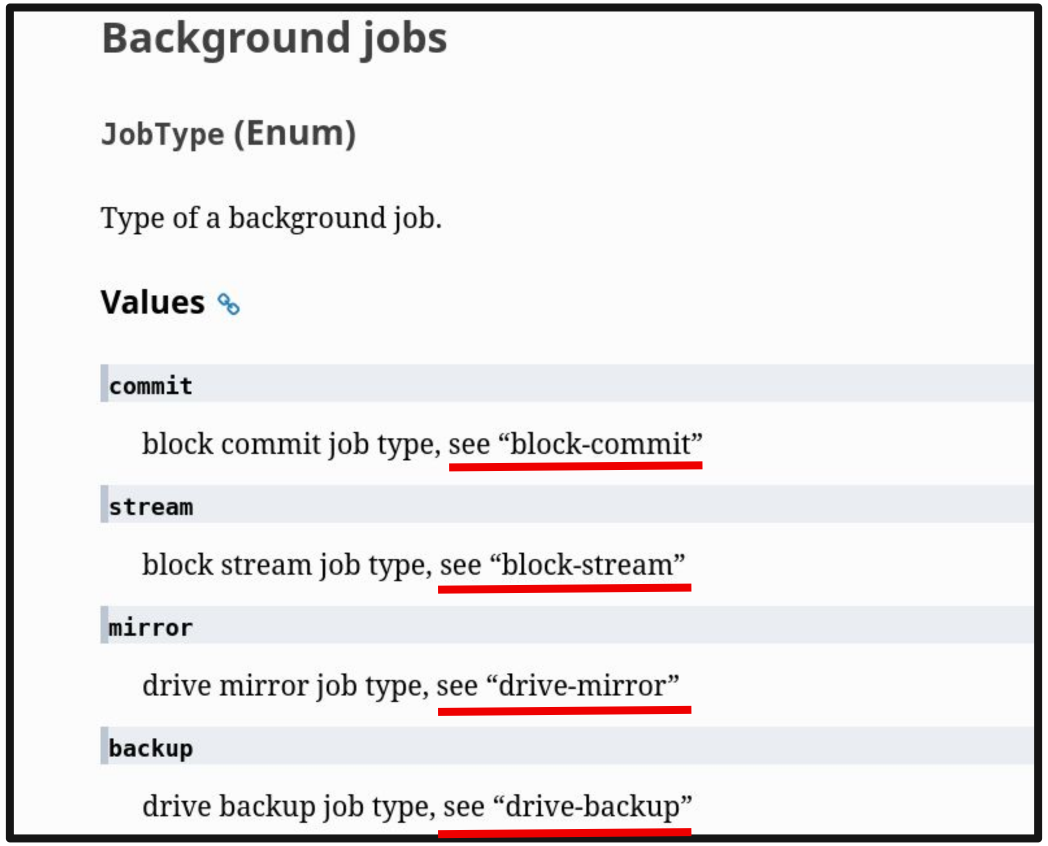
¿

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qemu-qmp-reference-manual

v1.1

No cross-references

See "block-commit"? I'd sure love to!



Background jobs

JobType (Enum)

Type of a background job.

Values

`commit`
block commit job type, see "block-commit"

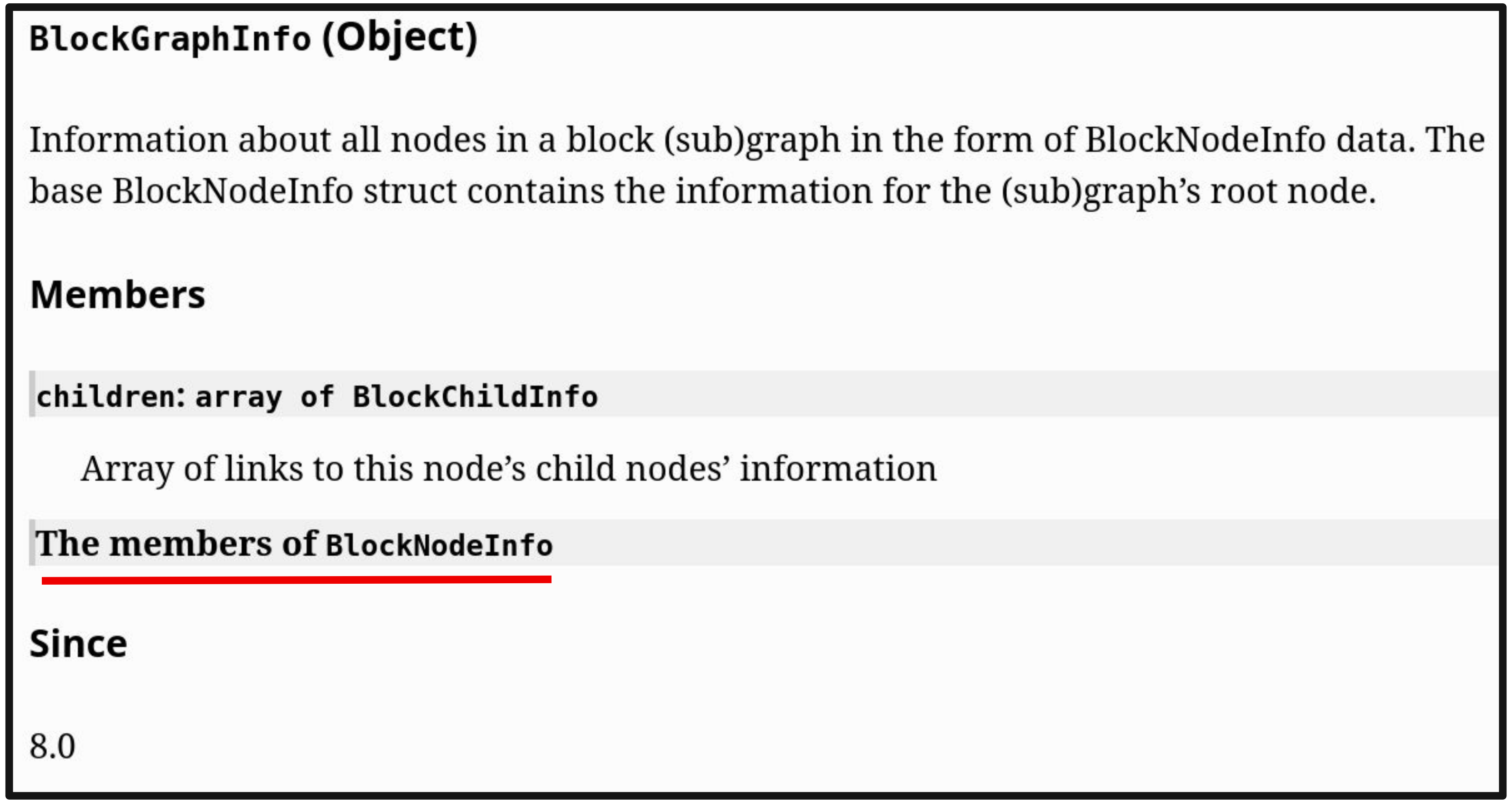
`stream`
block stream job type, see "block-stream"

`mirror`
drive mirror job type, see "drive-mirror"

`backup`
drive backup job type, see "drive-backup"

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-271

Incomplete documentation



`BlockGraphInfo` (Object)

Information about all nodes in a block (sub)graph in the form of BlockNodeInfo data. The base BlockNodeInfo struct contains the information for the (sub)graph's root node.

Members

`children: array of BlockChildInfo`

Array of links to this node's child nodes' information

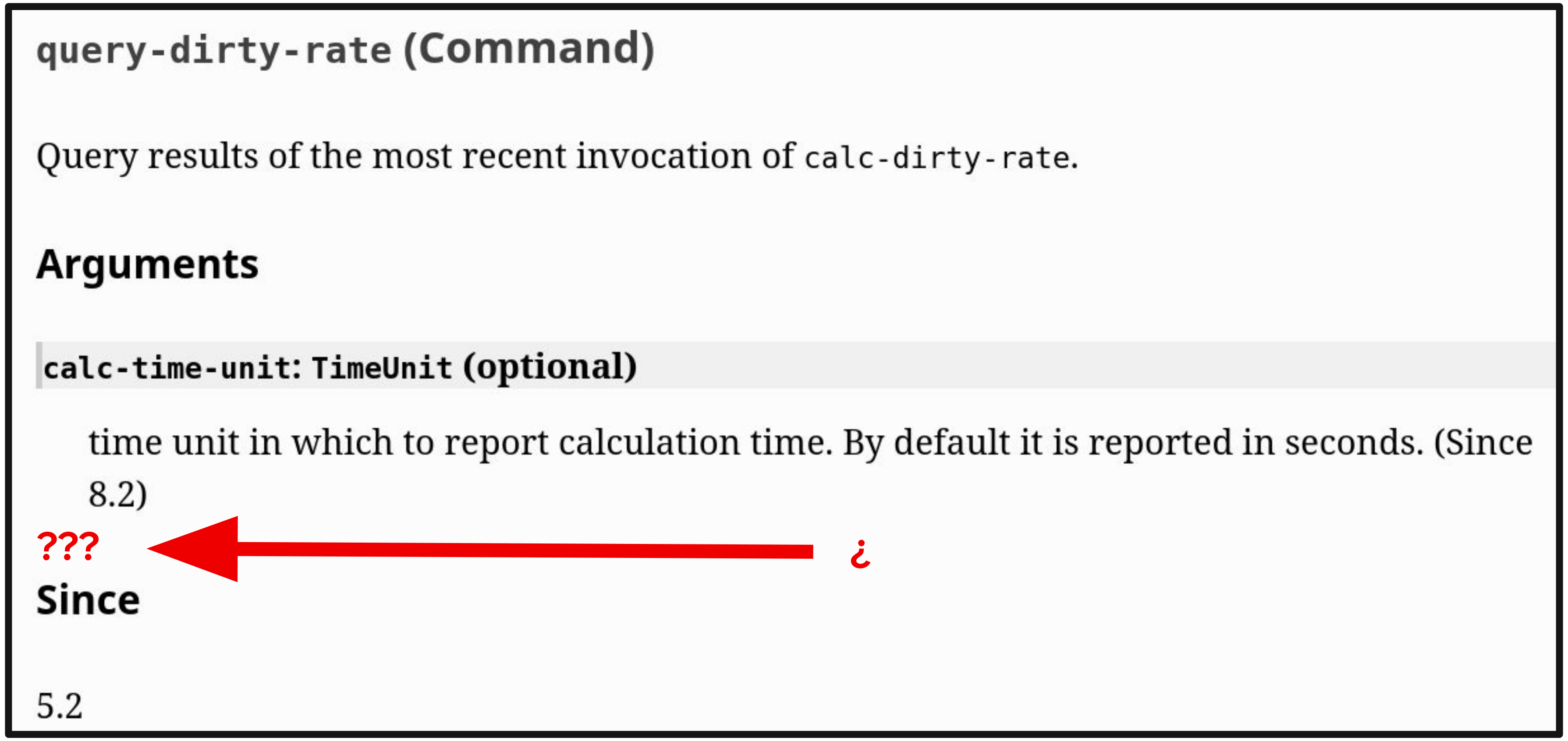
The members of `BlockNodeInfo`

Since

8.0

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-366

Missing “returns” info



query-dirty-rate (Command)

Query results of the most recent invocation of `calc-dirty-rate`.

Arguments

`calc-time-unit: TimeUnit` (optional)

time unit in which to report calculation time. By default it is reported in seconds. (Since 8.2)

???

¿

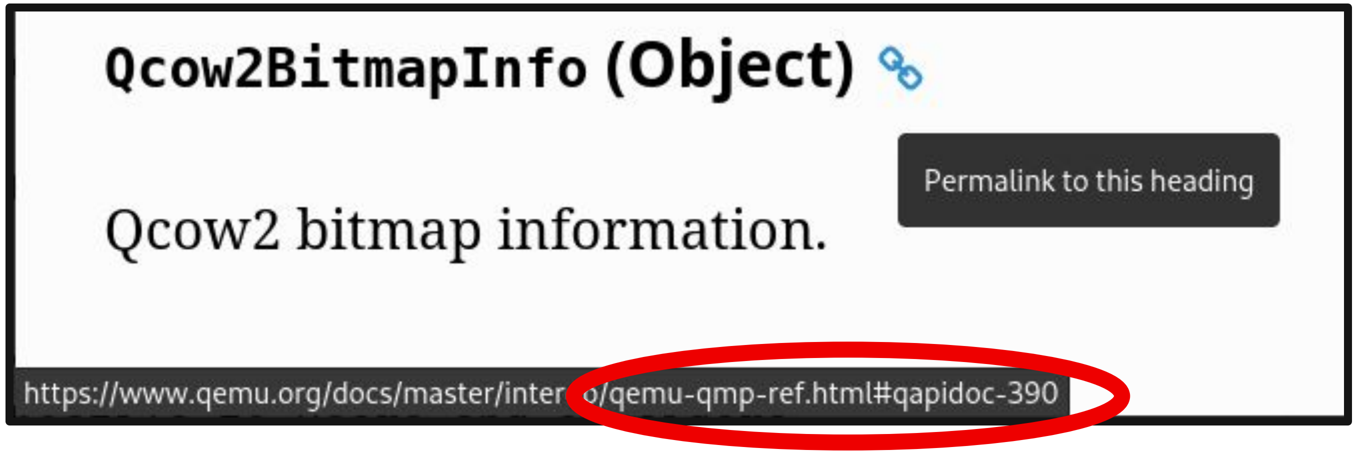
Since

5.2

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-1979

v1.1

Unstable URLs



Qcow2BitmapInfo (Object)

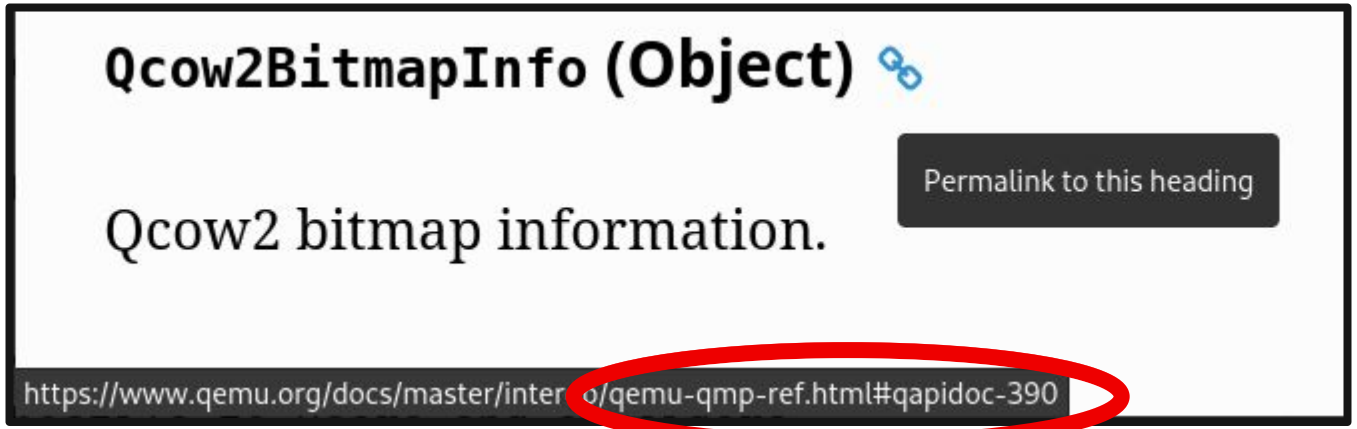
Qcow2 bitmap information.

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-390

v1.1

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-390

Unstable URLs



Qcow2BitmapInfo (Object)

Qcow2 bitmap information.

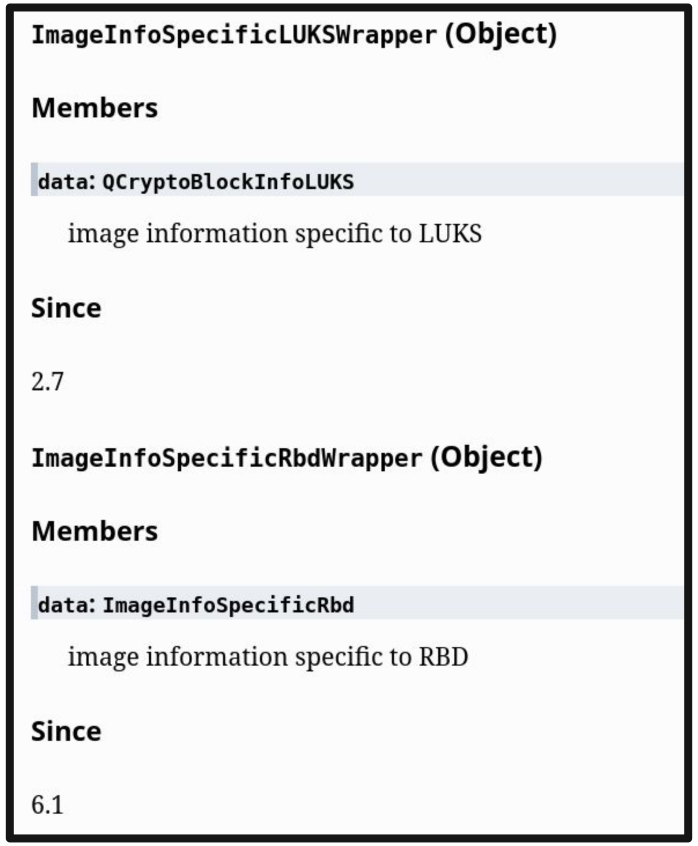
https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-390

For how long do you think the URLs in this presentation will work?

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-390

Poor visual contrast

Okay, this is subjective, but I've got the lectern



ImageInfoSpecificLUKSWrapper (Object)

Members

data: QCryptoBlockInfoLUKS

image information specific to LUKS

Since

2.7

ImageInfoSpecificRbdWrapper (Object)

Members

data: ImageInfoSpecificRbd

image information specific to RBD

Since

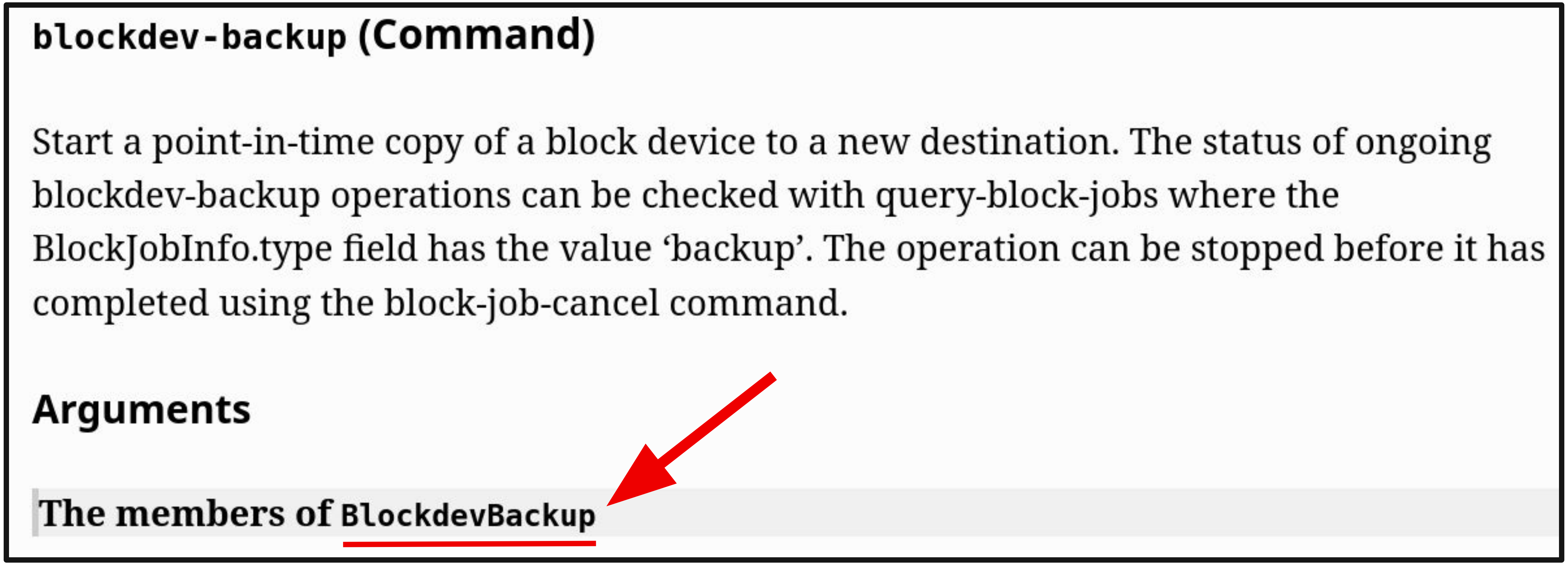
6.1

}?

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-345

Leaking implementation detail

We don't expose these names via introspection



blockdev-backup (Command)

Start a point-in-time copy of a block device to a new destination. The status of ongoing blockdev-backup operations can be checked with query-block-jobs where the BlockJobInfo.type field has the value 'backup'. The operation can be stopped before it has completed using the block-job-cancel command.

Arguments

The members of `BlockdevBackup`

https://www.qemu.org/docs/master/interop/qemu-qmp-ref.html#qapidoc-496

Lack of “holistic discoverability”

- Docs for any given entity are fragmented
- Lack of cross-references
- Incomplete command references
- How many clicks/pagefuls to read a single command?
- Tough to learn from docs this way

Anyone can make up cool sounding phrases. Try it out at home!

Developer-centric problems

Let’s whine about the development side for a little while now ;)

Hope you brought some cheese to go along with all of this whine

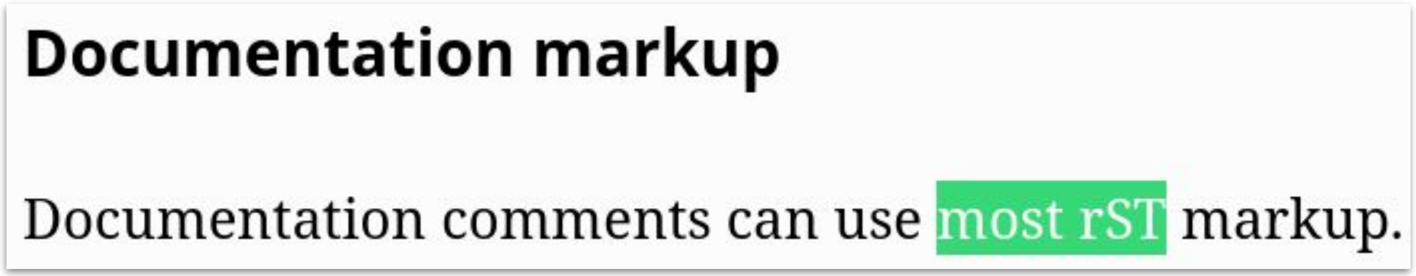
Fighting Sphinx Design

- Sphinx was not designed to tolerate manual docutils tree construction
- Sphinx APIs are based on extending Directives
- Tough to fix usability issues with our current architecture
- Prone to subtle bugs across versions

On the other hand, fighting a Sphinx sounds a lot cooler than writing documentation

Confusing Syntax

- When can you use rST in qapi source docs?
- What forms are allowed, and when?
- Here's `qapi-code-gen.html`:

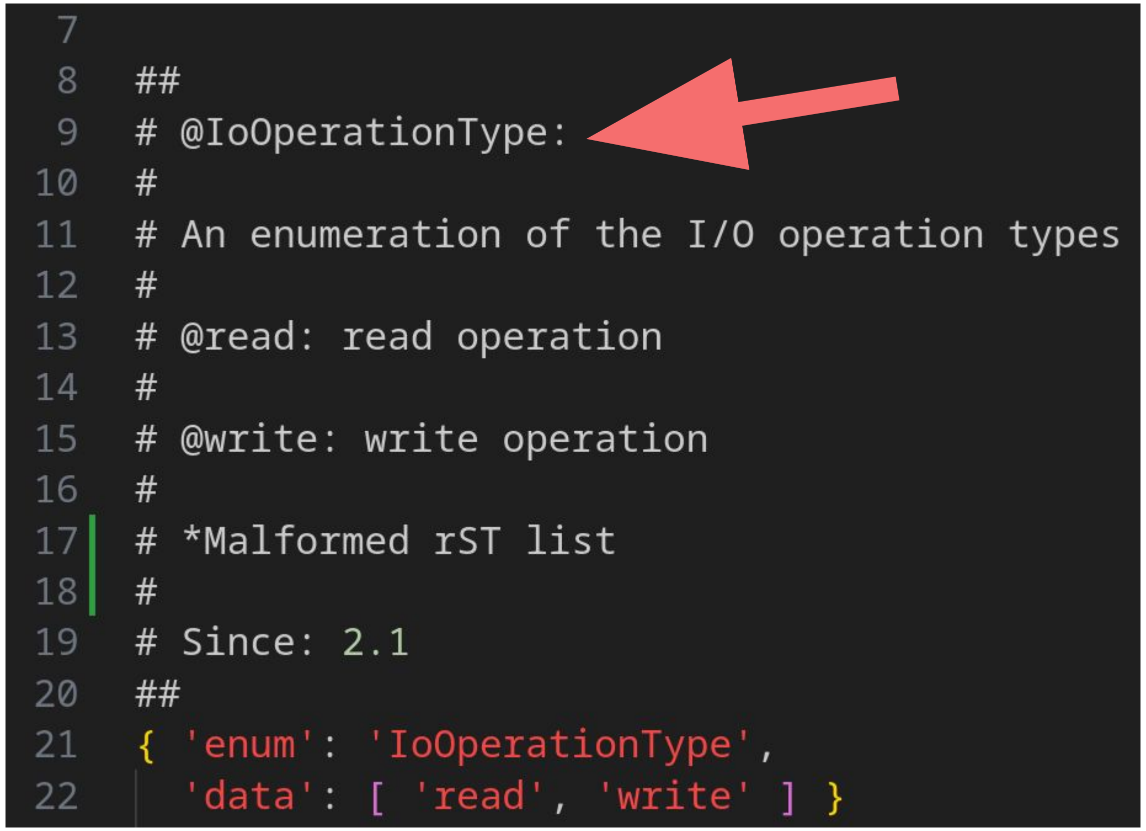


Documentation markup

Documentation comments can use most rST markup.

Well, that clears it up.

Errors are misleading



```
 7
 8  ##
 9  # @IoOperationType:
10  #
11  # An enumeration of the I/O operation types
12  #
13  # @read: read operation
14  #
15  # @write: write operation
16  #
17  # *Malformed rST list
18  #
19  # Since: 2.1
20  ##
21  { 'enum': 'IoOperationType',
22    'data': [ 'read', 'write' ] }
```

```
Warning, treated as error:
/home/jsnow/src/qemu/docs/.
./qapi/common.json:9:Inline
emphasis start-string
without end-string.

ninja: build stopped:
subcommand failed.
make: *** [Makefile:168:
run-ninja] Error 1
```

I don't blame any of you for hating rST or Sphinx, honestly

v1.1

Biased Opinions Slide

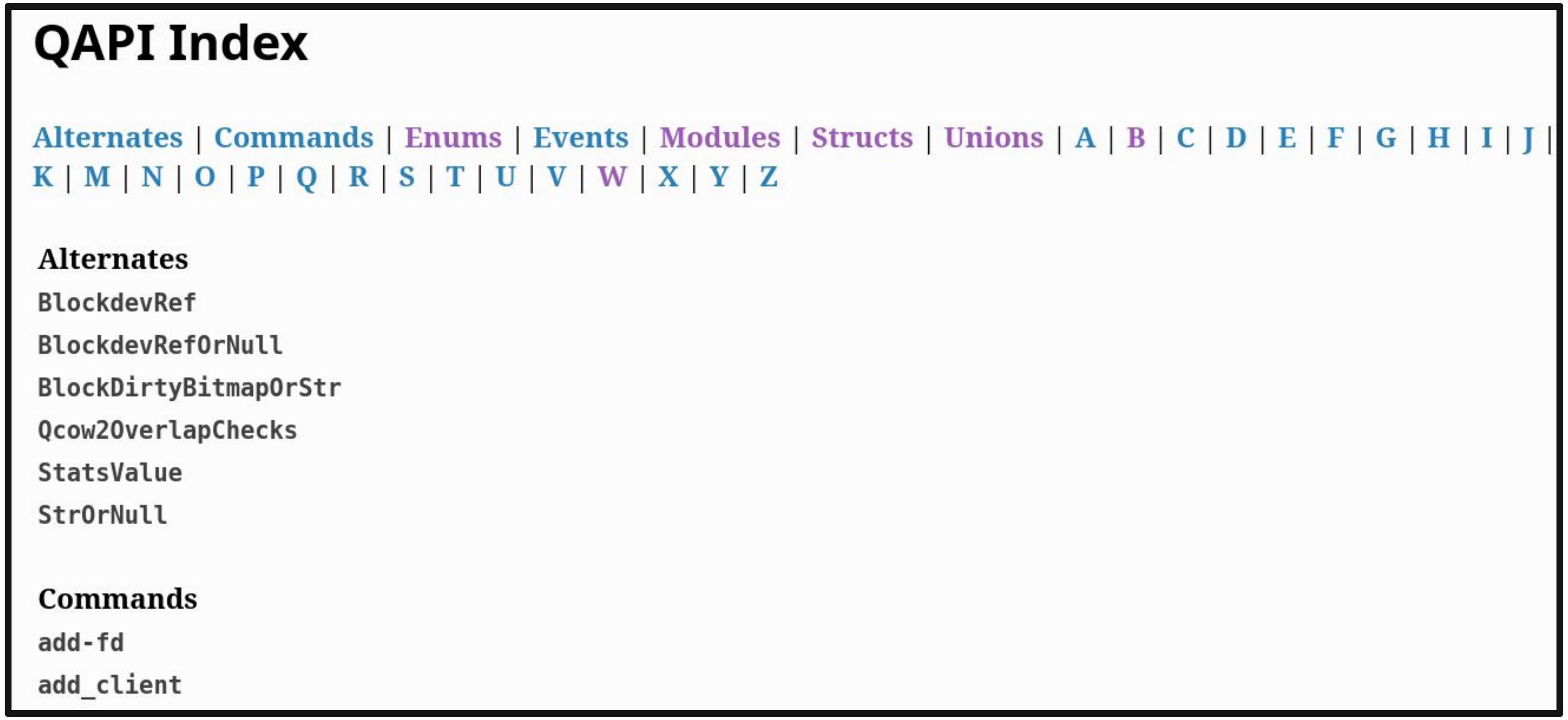
- Each irritant adds friction
- Contributes to perception of QEMU complexity
  - (And “bloat” and “slowness”)
  - Regardless of if that’s true or not…!
- First impressions matter!

The New Hotness

How we've improved QMP documentation to increase usability.

Indices

Now it’s easy to get a list of Commands and Events



QAPI Index

Alternates | Commands | Enums | Events | Modules | Structs | Unions | A | B | C | D | E | F | G | H | I | J | K | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Alternates

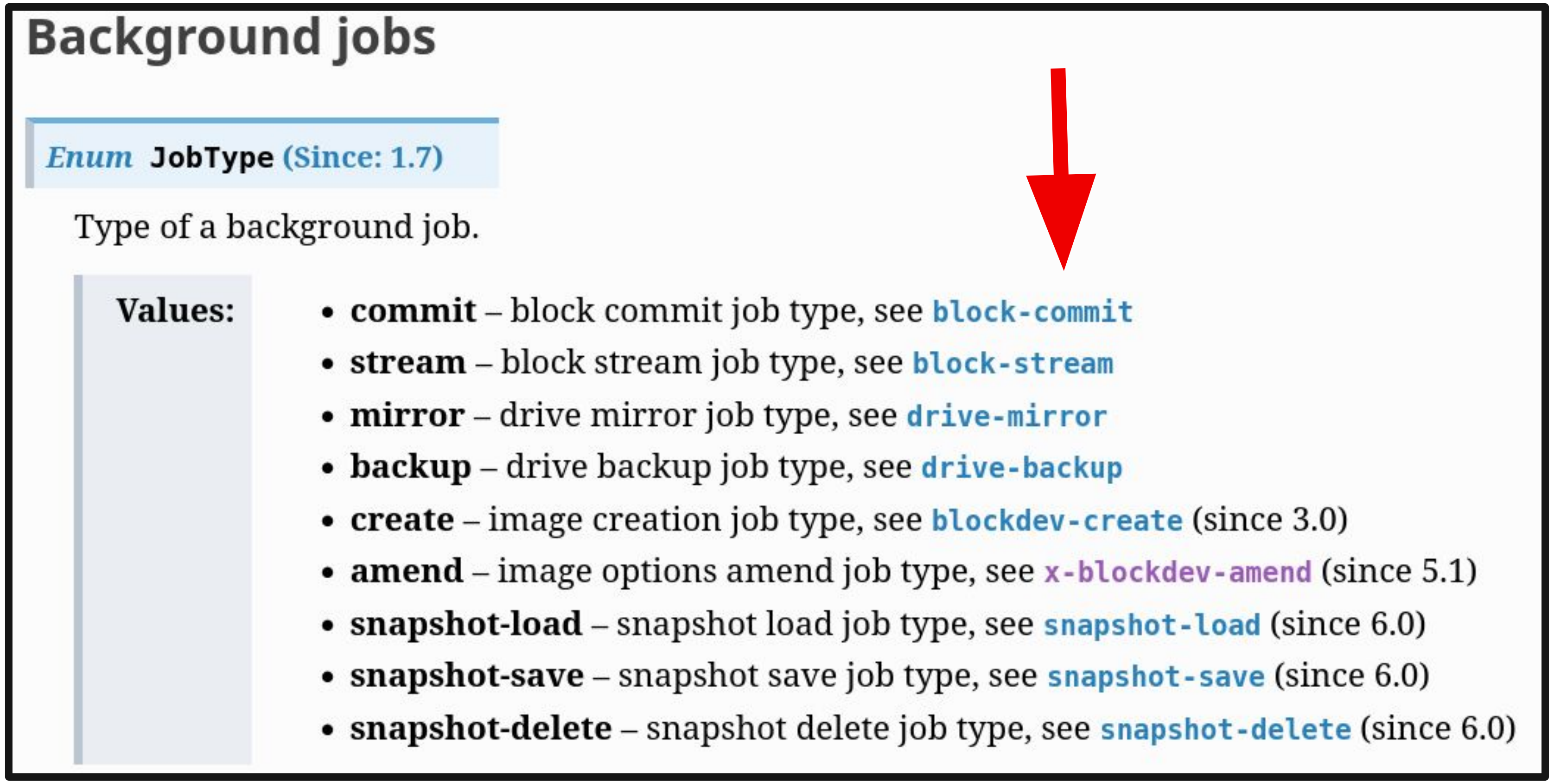
BlockdevRef
BlockdevRefOrNull
BlockDirtyBitmapOrStr
Qcow2OverlapChecks
StatsValue
StrOrNull

Commands

add-fd
add_client

Cross-references

No more plaintext “see also”



Background jobs

Enum **JobType** (Since: 1.7)

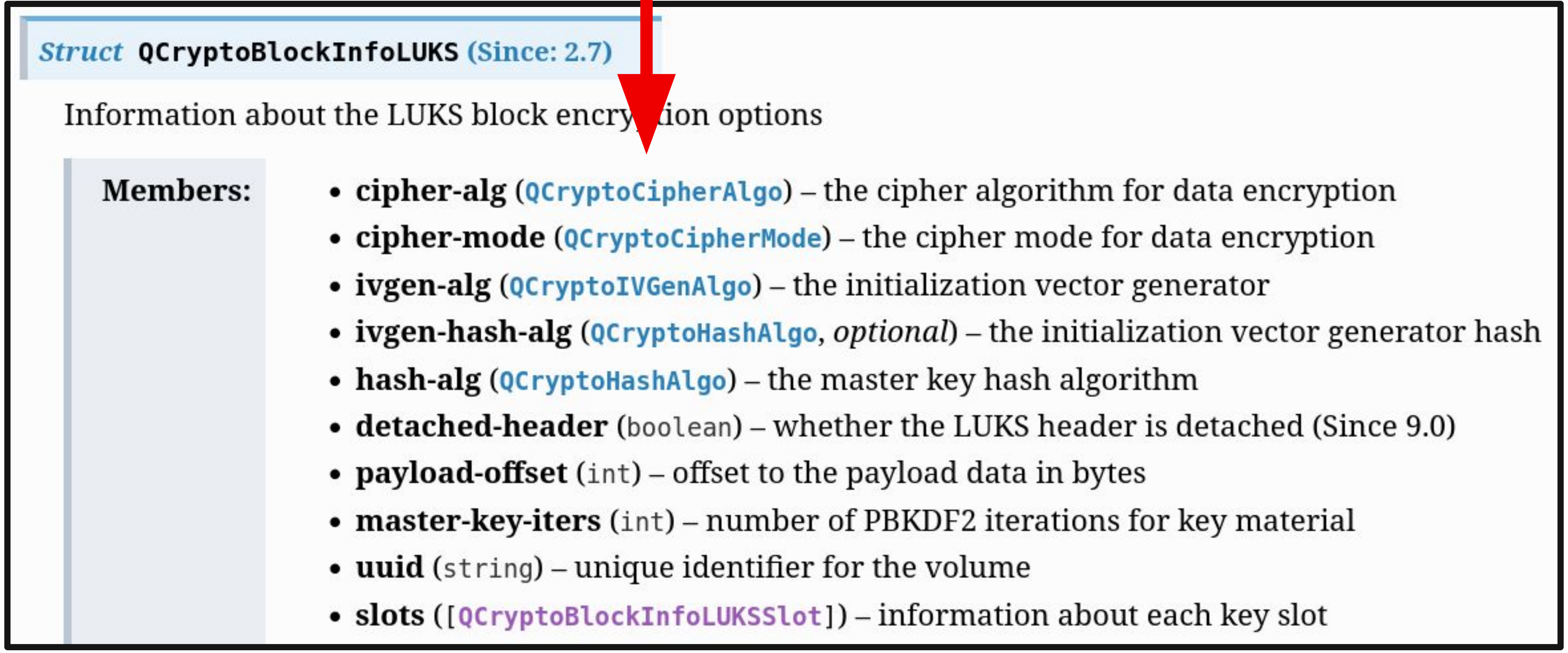
Type of a background job.

Values:

- **commit** – block commit job type, see `block-commit`
- **stream** – block stream job type, see `block-stream`
- **mirror** – drive mirror job type, see `drive-mirror`
- **backup** – drive backup job type, see `drive-backup`
- **create** – image creation job type, see `blockdev-create` (since 3.0)
- **amend** – image options amend job type, see `x-blockdev-amend` (since 5.1)
- **snapshot-load** – snapshot load job type, see `snapshot-load` (since 6.0)
- **snapshot-save** – snapshot save job type, see `snapshot-save` (since 6.0)
- **snapshot-delete** – snapshot delete job type, see `snapshot-delete` (since 6.0)

…Lots of cross-references

Types are automatically converted to references



Struct **QCryptoBlockInfoLUKS** (Since: 2.7)

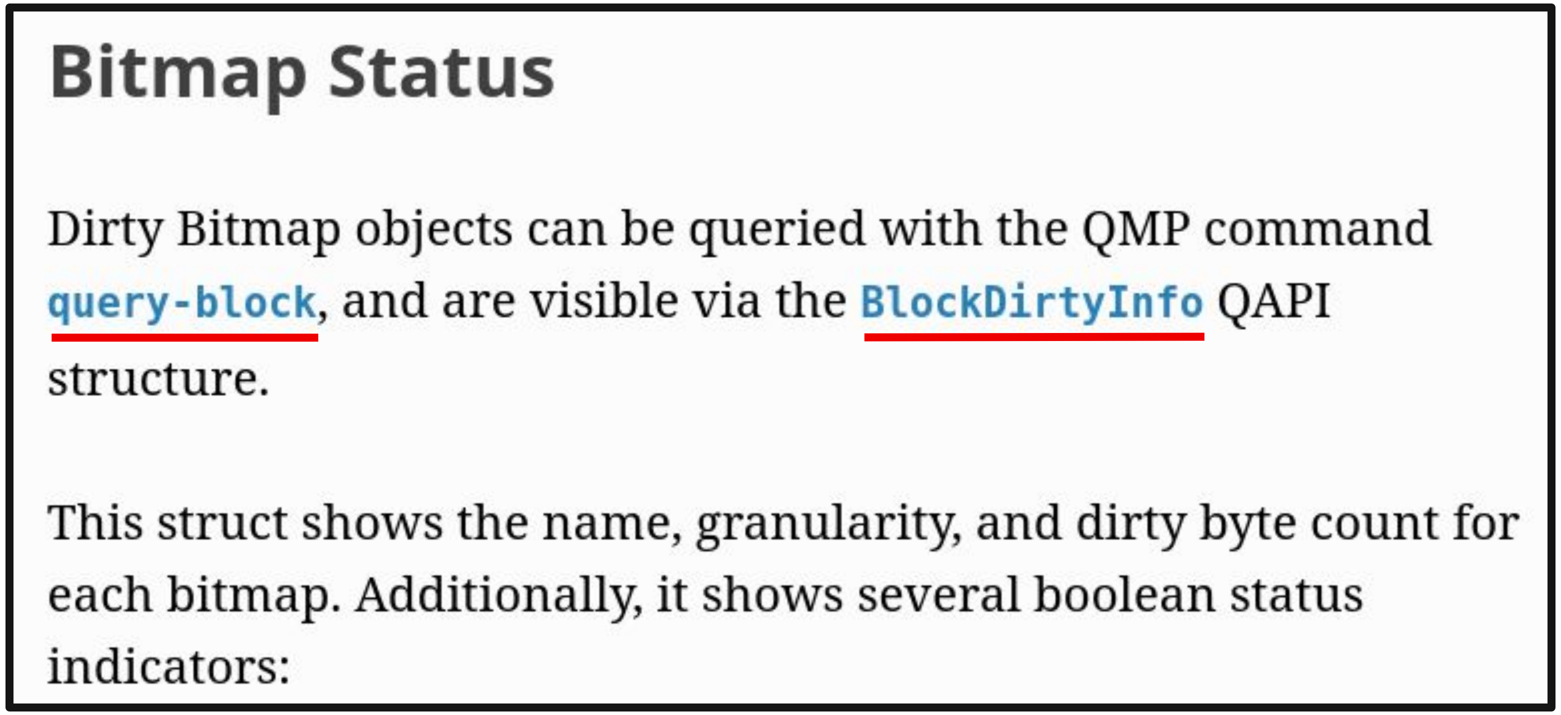
Information about the LUKS block encryption options

Members:

- **cipher-alg** (`QCryptoCipherAlgo`) – the cipher algorithm for data encryption
- **cipher-mode** (`QCryptoCipherMode`) – the cipher mode for data encryption
- **ivgen-alg** (`QCryptoIVGenAlgo`) – the initialization vector generator
- **ivgen-hash-alg** (`QCryptoHashAlgo`, *optional*) – the initialization vector generator hash
- **hash-alg** (`QCryptoHashAlgo`) – the master key hash algorithm
- **detached-header** (`boolean`) – whether the LUKS header is detached (Since 9.0)
- **payload-offset** (`int`) – offset to the payload data in bytes
- **master-key-iters** (`int`) – number of PBKDF2 iterations for key material
- **uuid** (`string`) – unique identifier for the volume
- **slots** (`[QCryptoBlockInfoLUKSSlot]`) – information about each key slot

Cross-references *everywhere!*

Excerpt from the incremental backup guide:



Bitmap Status

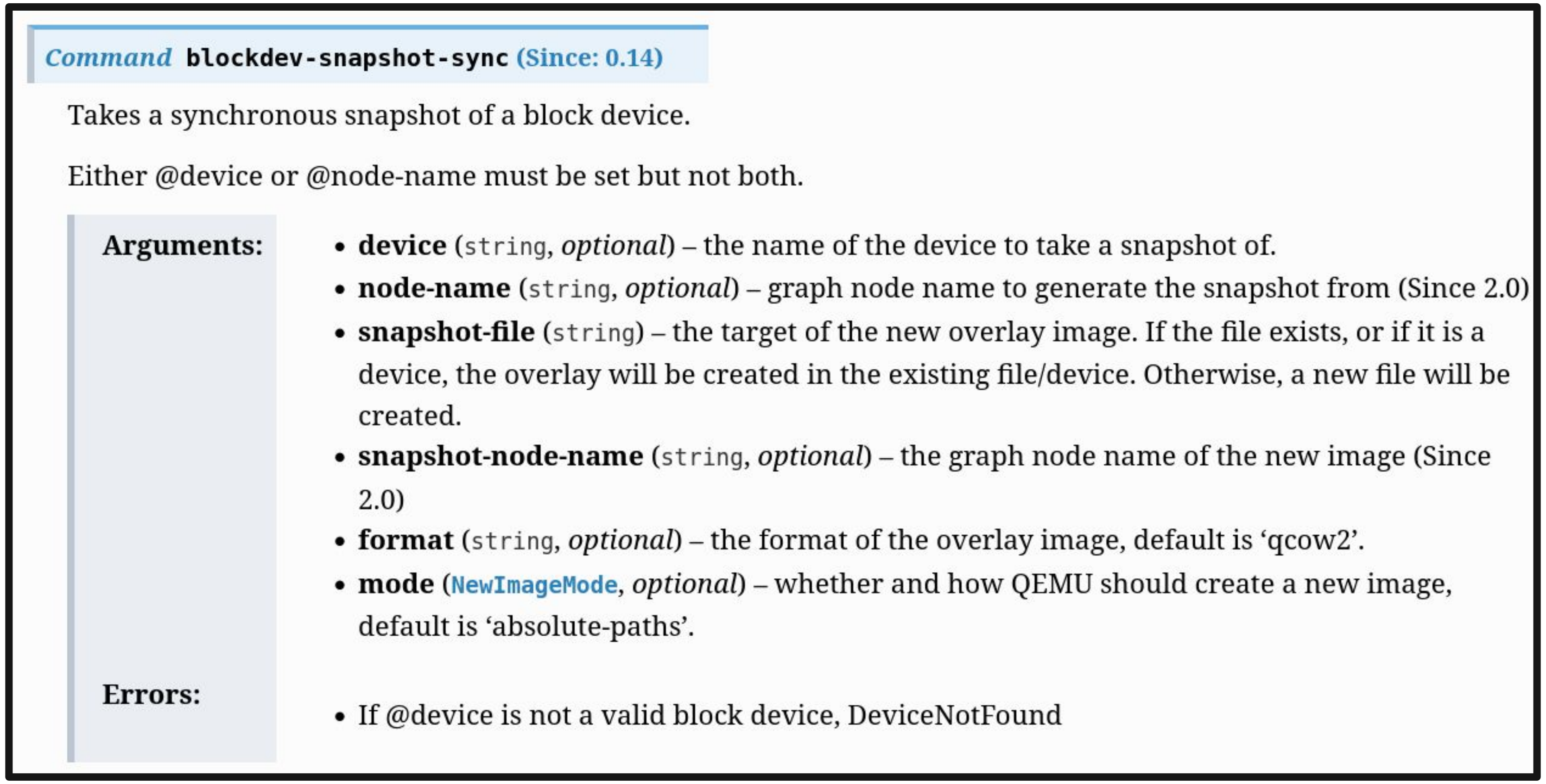
Dirty Bitmap objects can be queried with the QMP command `query-block`, and are visible via the `BlockDirtyInfo` QAPI structure.

This struct shows the name, granularity, and dirty byte count for each bitmap. Additionally, it shows several boolean status indicators:

v1.1

Complete documentation

Arguments are no longer “the members of …”



Command `blockdev-snapshot-sync` (Since: 0.14)

Takes a synchronous snapshot of a block device.

Either @device or @node-name must be set but not both.

Arguments:

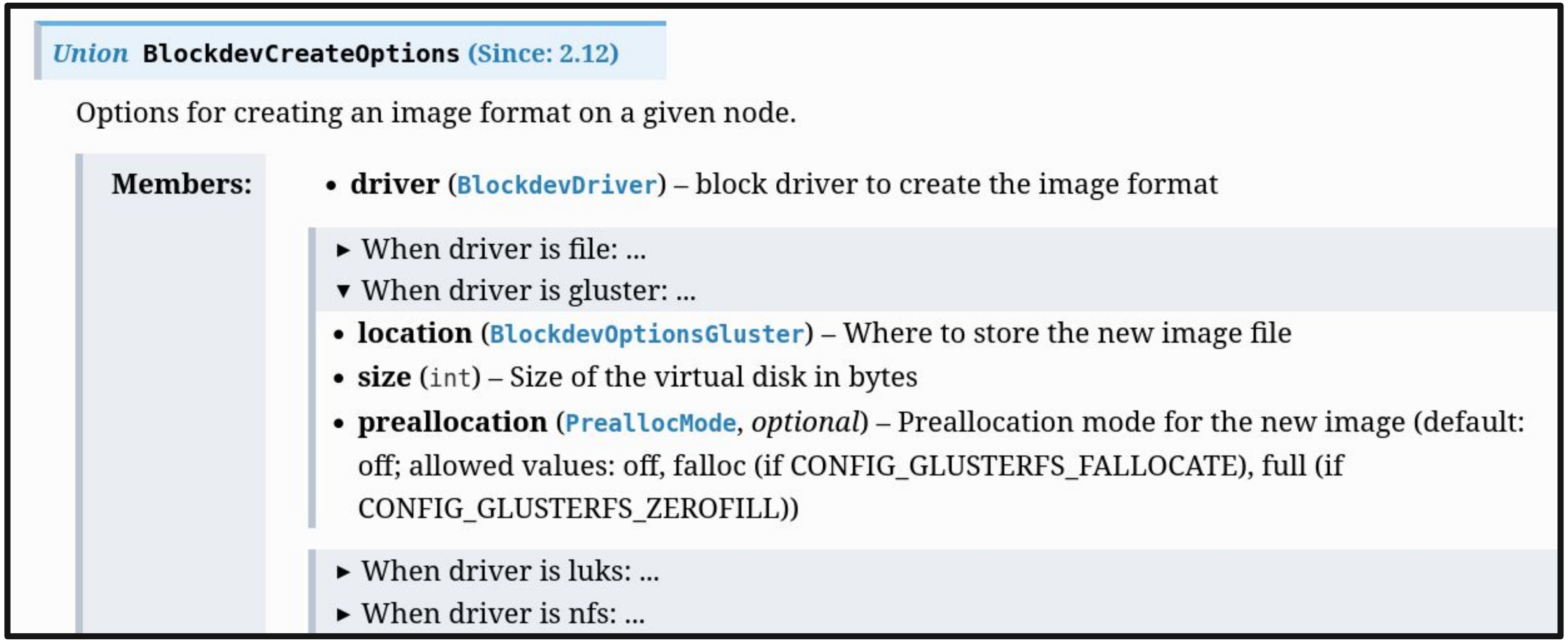
- **device** (`string`, *optional*) – the name of the device to take a snapshot of.
- **node-name** (`string`, *optional*) – graph node name to generate the snapshot from (Since 2.0)
- **snapshot-file** (`string`) – the target of the new overlay image. If the file exists, or if it is a device, the overlay will be created in the existing file/device. Otherwise, a new file will be created.
- **snapshot-node-name** (`string`, *optional*) – the graph node name of the new image (Since 2.0)
- **format** (`string`, *optional*) – the format of the overlay image, default is ‘qcow2’.
- **mode** (`NewImageMode`, *optional*) – whether and how QEMU should create a new image, default is ‘absolute-paths’.

Errors:

- If @device is not a valid block device, DeviceNotFound

Complete documentation

Unions are supported :)



Union **BlockdevCreateOptions** (Since: 2.12)

Options for creating an image format on a given node.

Members:

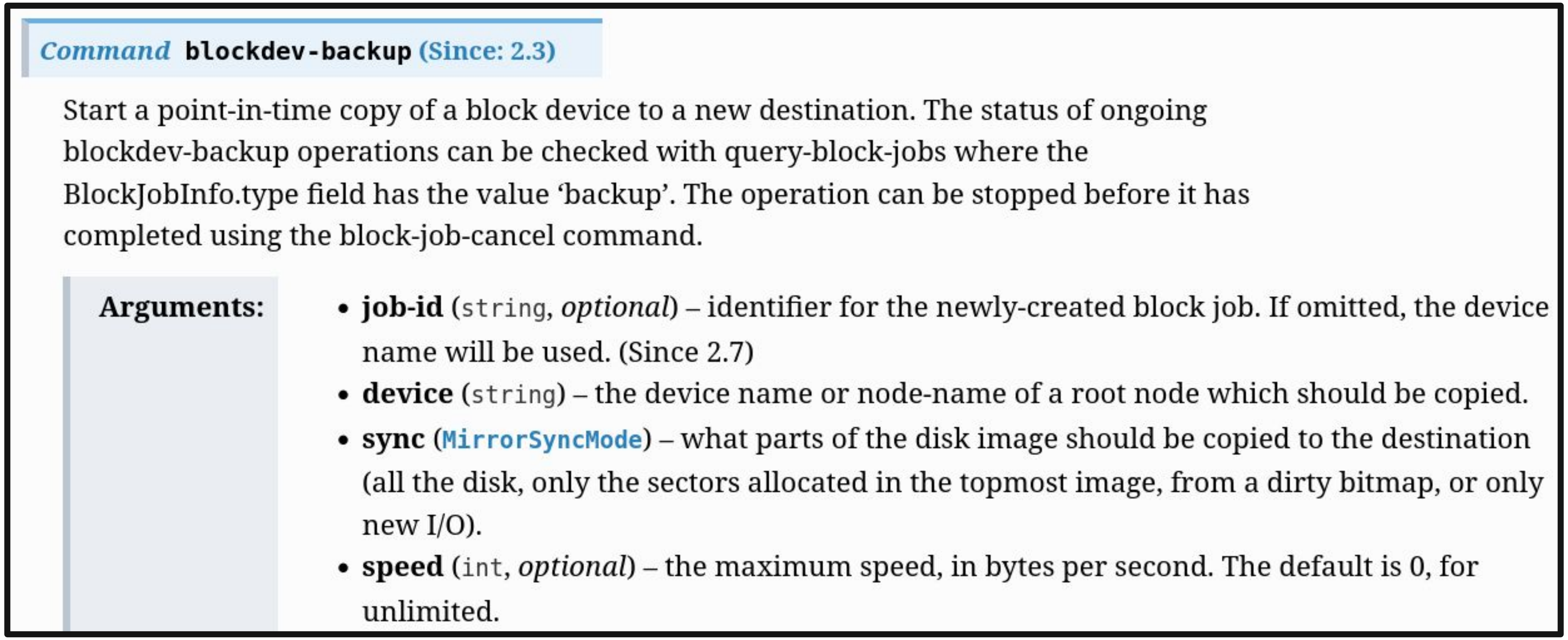
- **driver** (`BlockdevDriver`) – block driver to create the image format
  - ▸ When driver is file: ...
  - ▾ When driver is gluster: ...
    - **location** (`BlockdevOptionsGluster`) – Where to store the new image file
    - **size** (`int`) – Size of the virtual disk in bytes
    - **preallocation** (`PreallocMode`, *optional*) – Preallocation mode for the new image (default: off; allowed values: off, falloc (if CONFIG_GLUSTERFS_FALLOCATE), full (if CONFIG_GLUSTERFS_ZEROFILL))
  - ▸ When driver is luks: ...
  - ▸ When driver is nfs: ...

v1.1

I think this might require an HTML feature invented after 1996 though, so please make sure you've refreshed your company laptop within the last decade

Synchronicity with Introspection

“Internal” type names are leaked …less…



Command **`blockdev-backup`** (Since: 2.3)

Start a point-in-time copy of a block device to a new destination. The status of ongoing blockdev-backup operations can be checked with query-block-jobs where the BlockJobInfo.type field has the value ‘backup’. The operation can be stopped before it has completed using the block-job-cancel command.

Arguments:

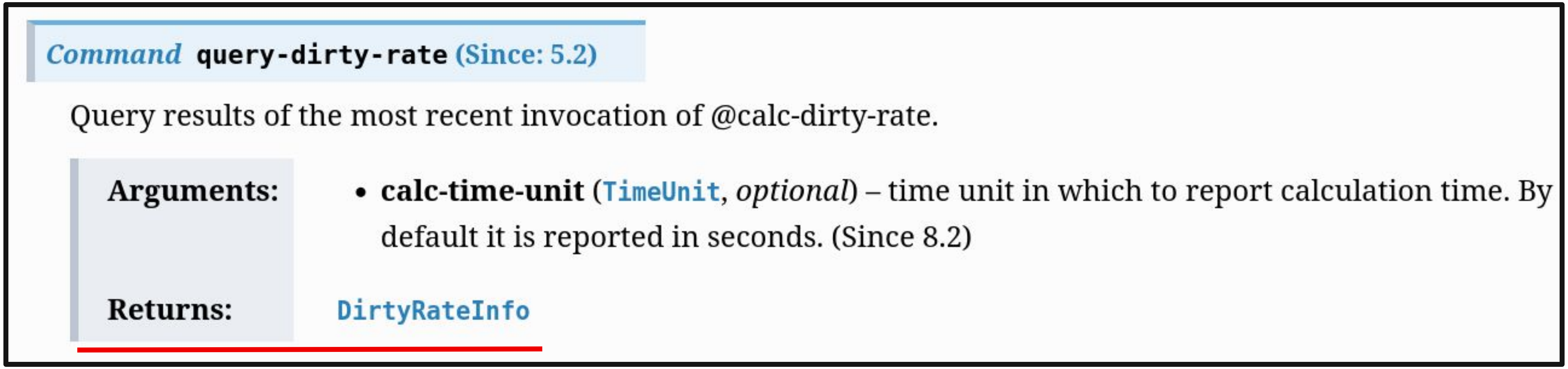
- **job-id** (`string`, *optional*) – identifier for the newly-created block job. If omitted, the device name will be used. (Since 2.7)
- **device** (`string`) – the device name or node-name of a root node which should be copied.
- **sync** (`MirrorSyncMode`) – what parts of the disk image should be copied to the destination (all the disk, only the sectors allocated in the topmost image, from a dirty bitmap, or only new I/O).
- **speed** (`int`, *optional*) – the maximum speed, in bytes per second. The default is 0, for unlimited.

Hey, it ain’t perfect.

v1.1

Generated “Returns” information

Now it’ll never be missing!



Command **query-dirty-rate** (Since: 5.2)

Query results of the most recent invocation of @calc-dirty-rate.

Arguments:

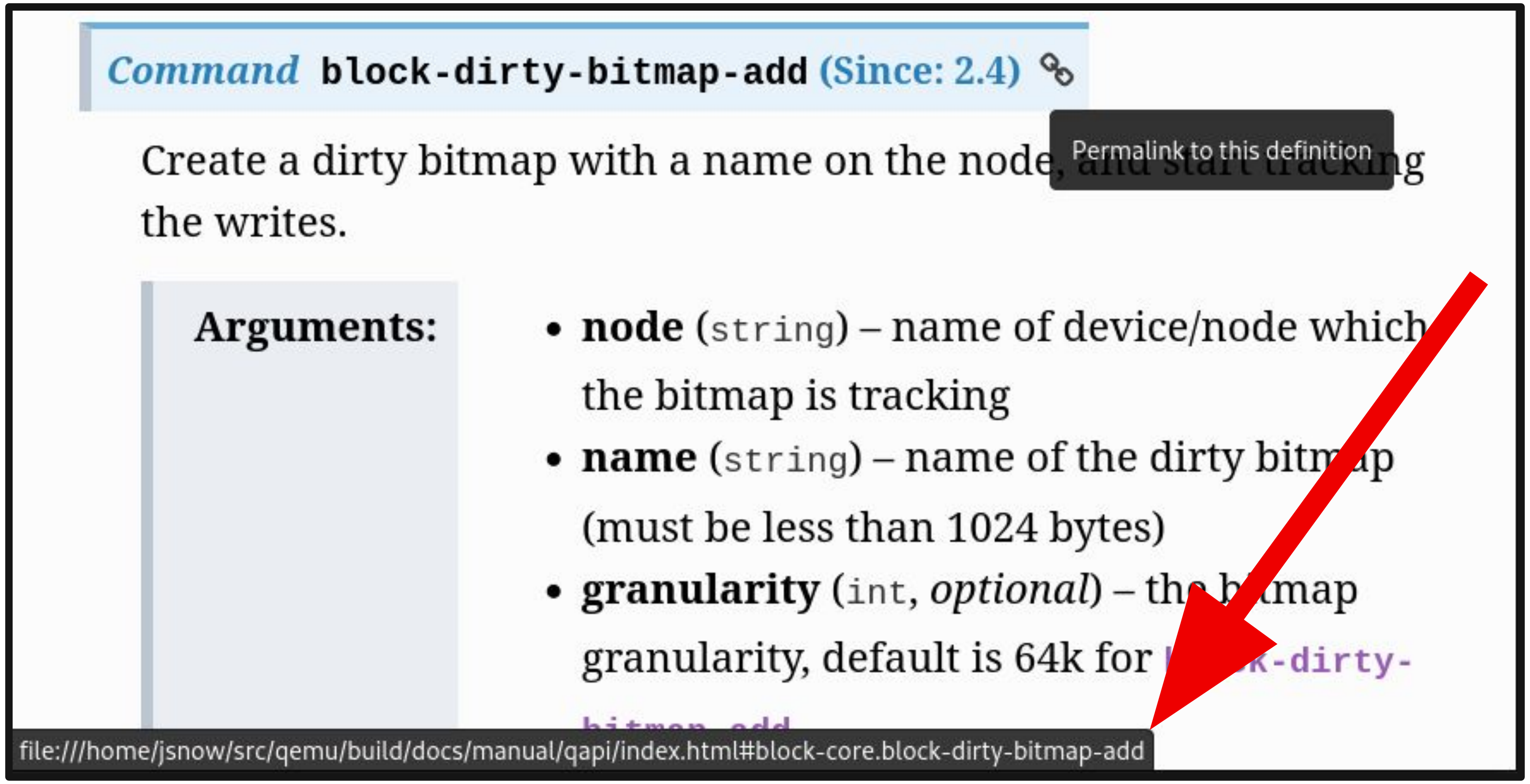
- **calc-time-unit** (`TimeUnit`, *optional*) – time unit in which to report calculation time. By default it is reported in seconds. (Since 8.2)

Returns: `DirtyRateInfo`

I was told once that using too many exclamation points was bad form for international communication, but I think this is the least-worst thing about Americans

Stable URLs

URLs are as stable as our API



Command `block-dirty-bitmap-add` (Since: 2.4)

Permalink to this definition

Create a dirty bitmap with a name on the node, and start tracking the writes.

Arguments:

- **node** (`string`) – name of device/node which the bitmap is tracking
- **name** (`string`) – name of the dirty bitmap (must be less than 1024 bytes)
- **granularity** (`int`, *optional*) – the bitmap granularity, default is 64k for `block-dirty-bitmap-add`

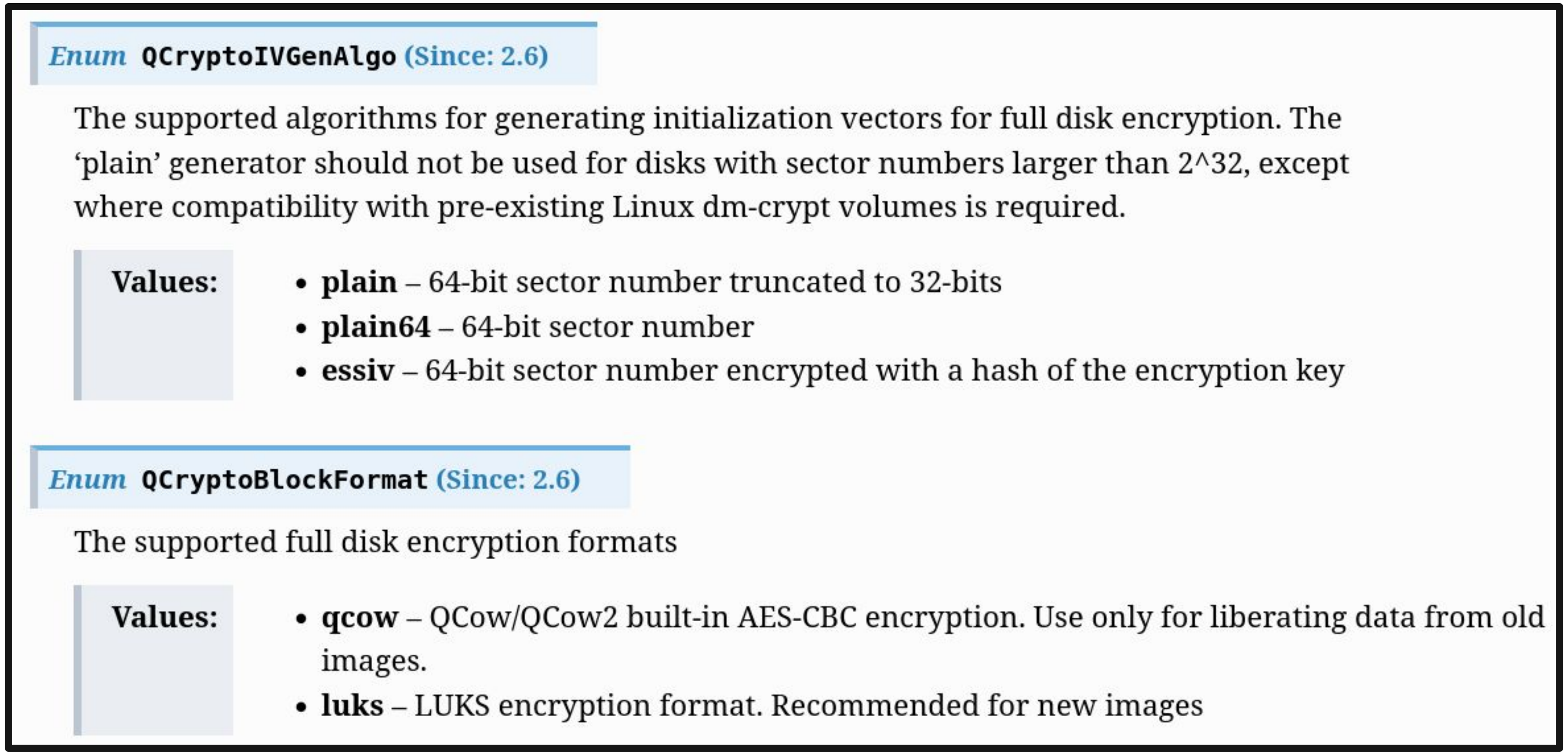
file:///home/jsnow/src/qemu/build/docs/manual/qapi/index.html#block-core.block-dirty-bitmap-add

Which is pretty infamously stable.

v1.1

Better visual contrast

Easier to tell where definitions begin and end



Enum `QCryptoIVGenAlgo` (Since: 2.6)

The supported algorithms for generating initialization vectors for full disk encryption. The 'plain' generator should not be used for disks with sector numbers larger than 2^32, except where compatibility with pre-existing Linux dm-crypt volumes is required.

Values:

- **plain** – 64-bit sector number truncated to 32-bits
- **plain64** – 64-bit sector number
- **essiv** – 64-bit sector number encrypted with a hash of the encryption key

Enum `QCryptoBlockFormat` (Since: 2.6)

The supported full disk encryption formats

Values:

- **qcow** – QCow/QCow2 built-in AES-CBC encryption. Use only for liberating data from old images.
- **luks** – LUKS encryption format. Recommended for new images

OK, yeah, it's kind of subjective but I can't actually tell you how many times I've actually gotten confused on our old docs page. This is the pet peeve slide.

“Holistic discoverability”

Benefits of the “inliner”

- Docs for commands/events include all fields
- Everything is cross-referenced
- Easier to get a “feel” for a command with just one click and a single screenful
- Docs now skew towards *QMP* and away from *QAPI*.

There was never going to be a way to show blockdev-add on a single screen, though

Working with the Sphinx

New qapidoc generator architecture

- Generate rST, not docutils nodes
- Greater re-use of Sphinx APIs
- Based very closely on Sphinx's Python domain
  - "If it's good for the goose, …"
- Fewer bugs from manual DOM building

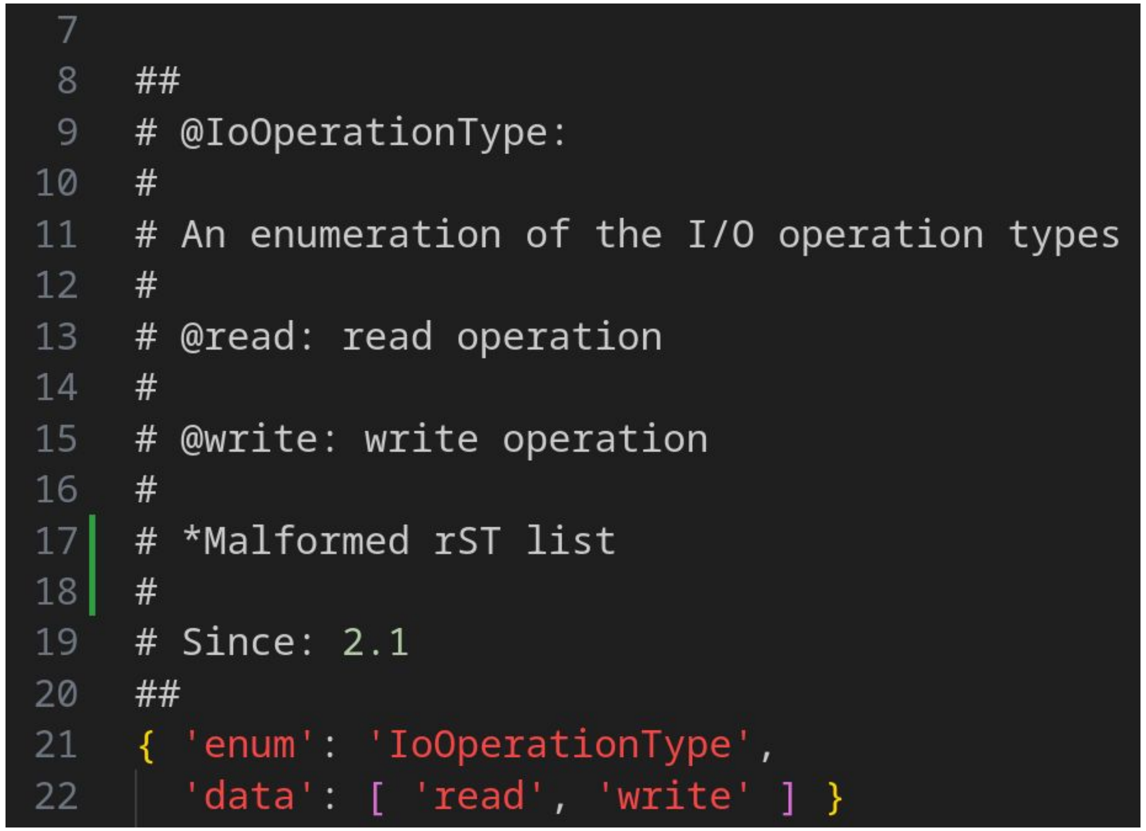
Is it safe to use Looney Tunes references? It's fine if you think I'm old, but I'm not crazy. Well. Not in that way.

More consistent syntax

- Virtually any rST markup can be used anywhere
  - Only thing off-limits are section headers
- Using Sphinx’s nested parse APIs:
  - removes guesswork
  - Improves compatibility
- rST syntax is still sorta goofy though, sorry

At least it’s a goofy standard and not goofy ad-hoc markup, I suppose …

More precise errors



```
 7
 8  ##
 9  # @IoOperationType:
10  #
11  # An enumeration of the I/O operation types
12  #
13  # @read: read operation
14  #
15  # @write: write operation
16  #
17  # *Malformed rST list
18  #
19  # Since: 2.1
20  ##
21  { 'enum': 'IoOperationType',
22    'data': [ 'read', 'write' ] }
```

```
Warning, treated as error:
/home/jsnow/src/qemu/docs/..
/qapi/common.json:17:Inline
emphasis start-string
without end-string.

ninja: build stopped:
subcommand failed.
make: *** [Makefile:168:
run-ninja] Error 1
```

Getting this to work correctly was an actual herculean labor and I hope you appreciate it the next time you get a doc error thrown back at you

New Syntax Tutorial (qapi-domain.py)

New Sphinx features such as indices and cross-references require some new syntax.

Here’s what’s changing.

The QAPI Domain

docs/sphinx/qapi-domain.py

- New Sphinx plugin
- Modeled after Sphinx's Python domain
- Multiple new rST directives
- **You won't need to use these directly**
    - autodoc:qapidoc :: python:qapi

Bold move to have a slide that openly says "This isn't important!" but I think it's important to have fun

New architecture

QAPI source

`.. qapi-doc::` directive in rST source invokes the `qapidoc.py` generator

qapidoc.py uses qapi/parser.py to parse the and build the schema

Intermediate rST

`qapidoc.py` generates rST markup containing new qapi-domain directives

Rendered docs

Sphinx performs the final parse using the `qapi-domain.py` plugin for new directives

Has anyone ever been fooled by "handwriting" fonts?

QAPI domain syntax

~ Bonus DLC Content! ~

- This is only a 30 minute talk
- Details in slides on pretalx
- Will be documented in QEMU manual :)

There was a pre-order bonus, but you’ve already missed the window :(

QAPI domain syntax

Defining “objects”

- `.. qapi:command:: query-block`
- `.. qapi:event:: SHUTDOWN`
- `.. qapi:enum:: ShutdownAction`
- `.. qapi:struct:: JobInfo`
- `.. qapi:alternate:: BlockdevRef`
- `.. qapi:union:: BlockdevOptions`

Common directive options

(syntax isn’t final)

```
.. qapi:command:: example-cmd
   :since: 0.14
   :ifcond: CONFIG_SPICE
   :deprecated:
   :unstable:

   lorem ipsum …
```

Field lists - :arg:

Commands use **`:arg type name: descr`**

```
.. qapi:command:: example-cmd

   lorem ipsum …

   :arg BlockDirtyBitmap? bitmap: dolor sit amet,
       consectetur adipiscing elit …
```

Field lists - :value:

Enums use **`:value name: descr`**

```
.. qapi:enum:: example-enum

   lorem ipsum …

   :value sphinx: Some kind of Chimera?
   :value hippogriff: Also some kind of Chimera,
       but different…
```

Field lists - :choice:

Alternates use **`:choice type name: descr`**

```
.. qapi:alternate:: example-alt

   lorem ipsum …

   :choice taco t: Tacos for dinner?
   :choice enchilada e: Or how about Enchiladas?
```

- This might change if I can work out how to exclude the name …

Field lists - :memb:

Events, unions, and structs use **`:memb type name: descr`**

```
.. qapi:event:: BLOCK_JOB_COMPLETED

   lorem ipsum …

   :memb JobType type: job type
   :memb string device: The job identifier …
```

Field lists - :feat:

Features use **`:feat name: descr`**

```
.. qapi:command:: device_add

   … etc …

   :feat json-cli: If present, the “-device”
       command line option supports JSON syntax …
```

Red Hat

Field lists - :error: and :returns:

Special entries for commands

```
.. qapi:command:: query-jobs

   … etc …

   :error: When you don’t ask nicely
   :returns [JobInfo]: A list with … etc …
   :returns-nodesc: [JobInfo]
```

Special domain syntax

- `.. qapi:branch:: driver bochs`
  - Starts a new tagged branch subsection
  - key=`driver`
  - value=`bochs`
  - Use nested `:arg:` or `:memb:` as appropriate

(Breathe)

😮‍💨

New Syntax Tutorial (QAPIDoc)

The source format of QAPI documentation is changing just a teensy little bit.

Markus has kept me conservative and honest here.

(Mostly.)

The transition should be smooth, I promise!

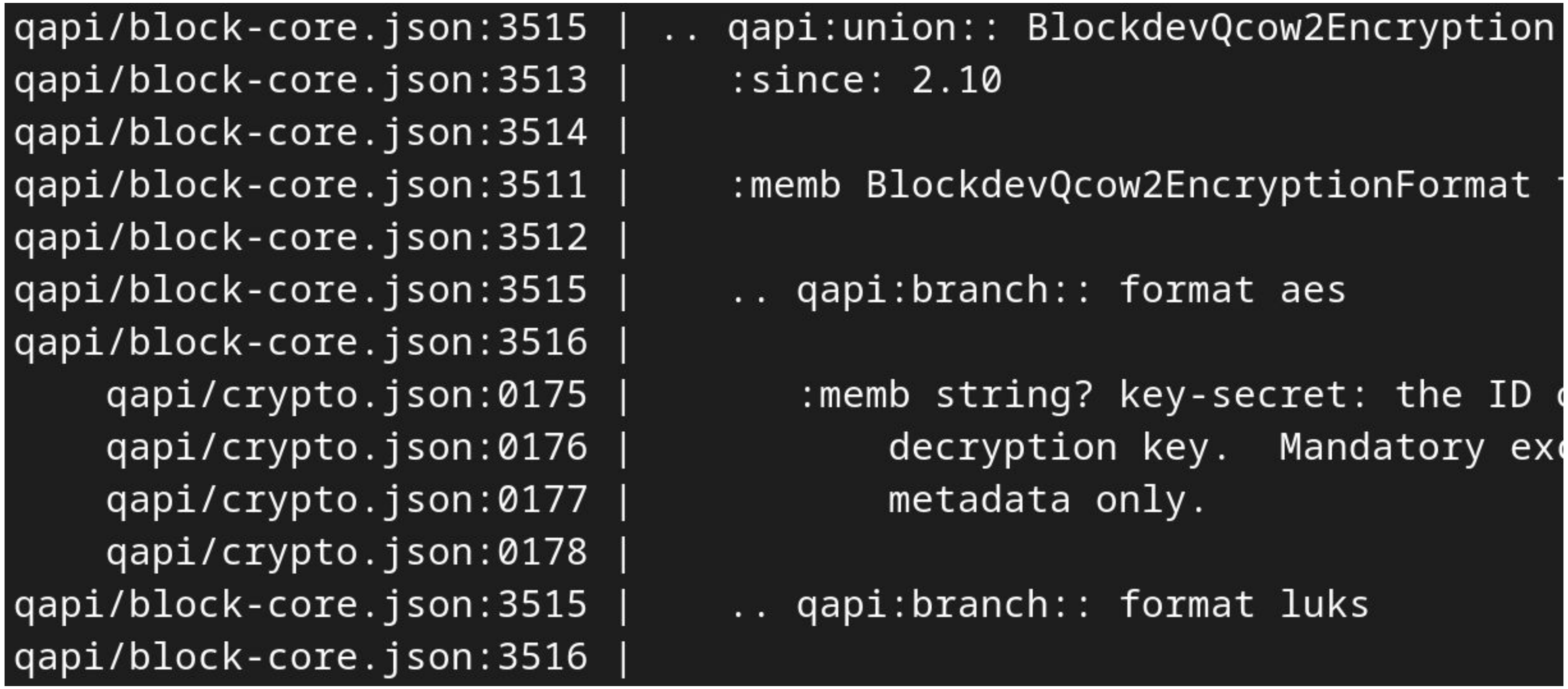
Debugging qapidoc

Use `DEBUG=1`

- DEBUG=1 writes generated rST doc to disk
- E.g. `build/qapi_qapi-schema.ir`
- Understanding qapi-domain directives will help here 😁
- Two-column output with **source attributions.**

Debugging qapidoc

Use DEBUG=1



```
qapi/block-core.json:3515 | .. qapi:union:: BlockdevQcow2Encryption
qapi/block-core.json:3513 |    :since: 2.10
qapi/block-core.json:3514 |
qapi/block-core.json:3511 |    :memb BlockdevQcow2EncryptionFormat
qapi/block-core.json:3512 |
qapi/block-core.json:3515 |    .. qapi:branch:: format aes
qapi/block-core.json:3516 |
    qapi/crypto.json:0175 |       :memb string? key-secret: the ID
    qapi/crypto.json:0176 |          decryption key.  Mandatory ex
    qapi/crypto.json:0177 |          metadata only.
    qapi/crypto.json:0178 |
qapi/block-core.json:3515 |    .. qapi:branch:: format luks
qapi/block-core.json:3516 |
```

Red Hat

Cross-references

Use 'em!

- `` :qapi:cmd:`query-block` ``
- `` :qapi:event:`BLOCK_IMAGE_CORRUPTED` ``
- `` :qapi:enum:`NetworkAddressFamily` ``
- `` :qapi:struct:`JobInfo` ``
- `` :qapi:alt:`BlockDirtyBitmapOrStr` ``
- `` :qapi:union:`BlockdevOptions` ``

Cross-references

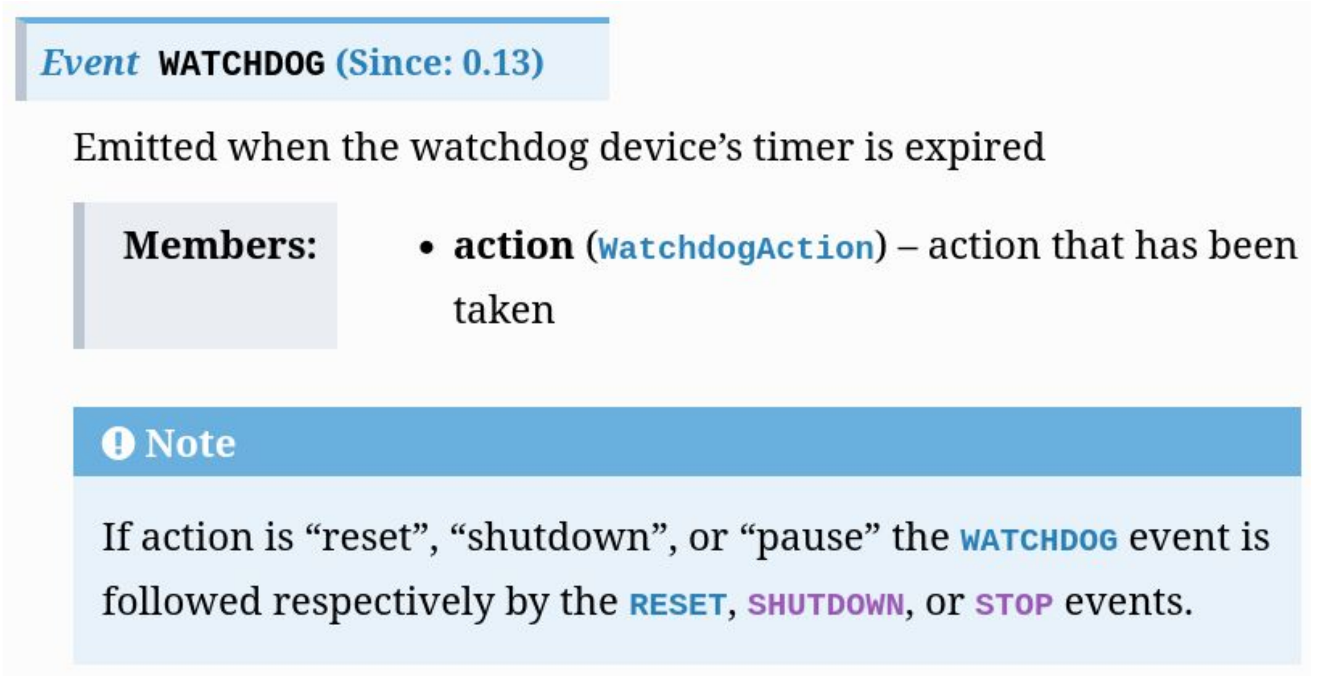
Use 'em!

- `:qapi:type:`SpiceBasicInfo``
  - Any qapi type
  - but not events or commands
- `:qapi:obj:`SHUTDOWN``
  - Any qapi thing at all
- ``blockdev-create``
  - *Anything.*

“Note(s):” is now `.. note::`

(This is just plain rST.)

```
##
# @WATCHDOG:
#
# Emitted when the watchdog device's timer is expired
#
# @action: action that has been taken
#
# .. note:: If action is "reset", "shutdown", or "pause"
#     the `WATCHDOG` event is followed respectively
#     by the `RESET`, `SHUTDOWN`, or `STOP`
#     events.
```



Event **WATCHDOG** (Since: 0.13)

Emitted when the watchdog device’s timer is expired

Members:
- **action** (`WatchdogAction`) – action that has been taken

Note

If action is “reset”, “shutdown”, or “pause” the `WATCHDOG` event is followed respectively by the `RESET`, `SHUTDOWN`, or `STOP` events.

You can also use “warning” or “caution” or “tip” or like, whatever.

“Example(s):” is now `.. qmp-example::`

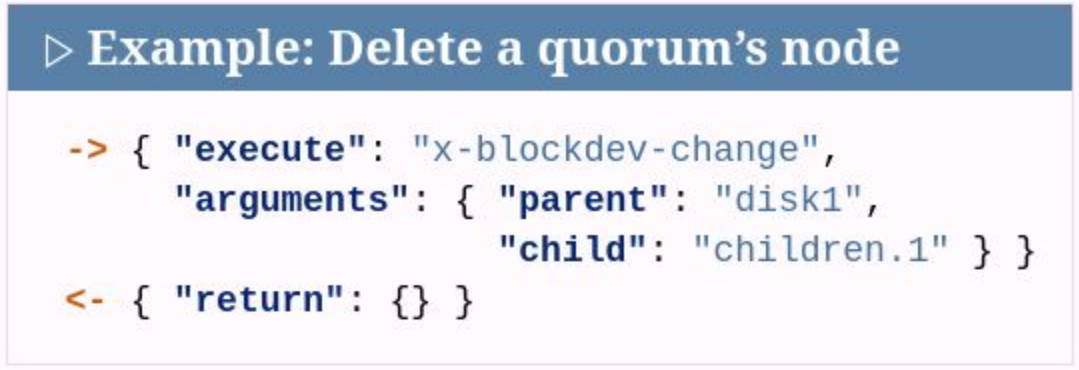
This one is a jsnow special

- Applies QMP highlighting and lexical validation
- Looks cool 😎
- Supports mixing prose and code
- Directive in `docs/sphinx/qapidoc.py`
- Lexer in `docs/sphinx/qmp_lexer.py`

“Example(s):” is now `.. qmp-example::`

This one is a jsnow special

```
# .. qmp-example::
#   :title: Delete a quorum's node
#
#   -> { "execute": "x-blockdev-change",
#         "arguments": { "parent": "disk1",
#                        "child": "children.1" } }
#   <- { "return": {} }
```



▷ Example: Delete a quorum’s node

```
-> { "execute": "x-blockdev-change",
     "arguments": { "parent": "disk1",
                    "child": "children.1" } }
<- { "return": {} }
```

Intro/Details sections

- Text at the start of the doc is now considered the “intro”
  - This text is never inlined into other entries!
- Text at the end of the doc is now the “details” section
  - This text **is* inlined into other entries!

Red Hat

Intro/Details sections

```
##
# @BlockdevOptionsNVMe:
#
# Driver specific block device options for the NVMe backend.
#
# @device: PCI controller address of the NVMe device in format
#     hhhh:bb:ss.f (host:bus:slot.function)
#
# @namespace: namespace number of the device, starting from 1.
#
# Note that the PCI @device must have been unbound from any host
# kernel driver before instructing QEMU to add the blockdev.
##
```

Intro

Tagged region

Details

Ambiguous Intro/Details

```
##
# @announce-self:
#
# Trigger generation of broadcast RARP frames to update network
# switches.  This can be useful when network bonds fail-over the
# active slave.
#
# TODO: This line is a hack to separate the example from the intro
#
# .. qmp-example::
#
#     -> { ... }
#     <- { "return": {} }
##
```

Intro

???

Details

Where to go from here?

K, cool story, when can we have these new goodies?

What’s stopping you from merging it?

“Since” sections

- They’re still around for now, but …
- …we’ve got some ideas to remove them and generate that information instead
- Historical QAPI schema parser tool
  - What format to store parse results in…?
  - Work in progress!

Dan told me I went “ridiculously overboard” with this and I took that as a compliment

“Since” sections II: Semantic boogaloo

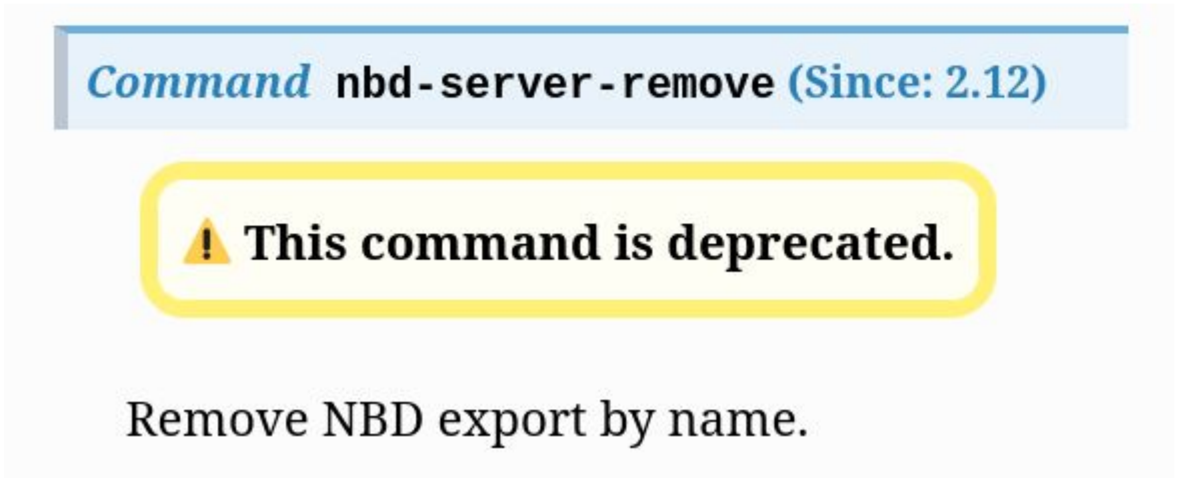
- We want to drop and generate this info
  - In the meantime, what do we do with it?
- Sometimes written to explain source factoring
- Sometimes written to explain user factoring
- Needs audit 😵‍💫

I think I’ve made an Electric Boogaloo joke in every single talk I’ve ever given for Red Hat

Layout design issues

I have problems with commitment

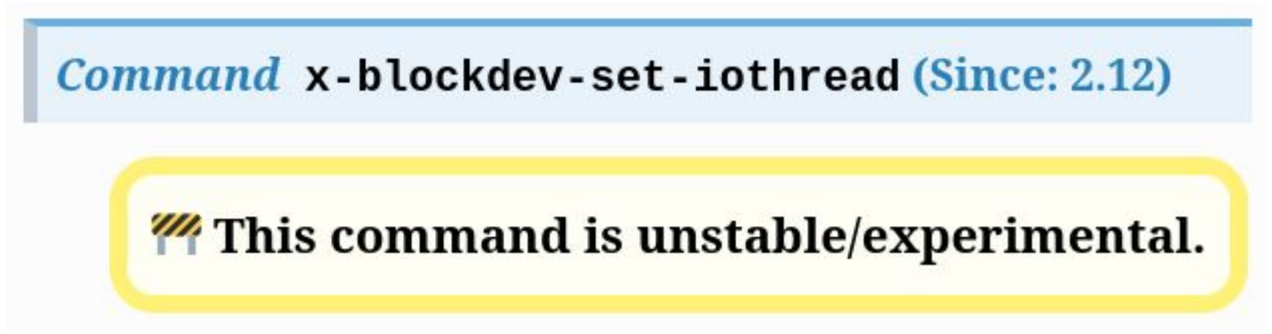
- Where should "Since" information go?
- What about "Deprecated" and "Unstable"?



Command `nbd-server-remove` (Since: 2.12)

⚠ **This command is deprecated.**

Remove NBD export by name.



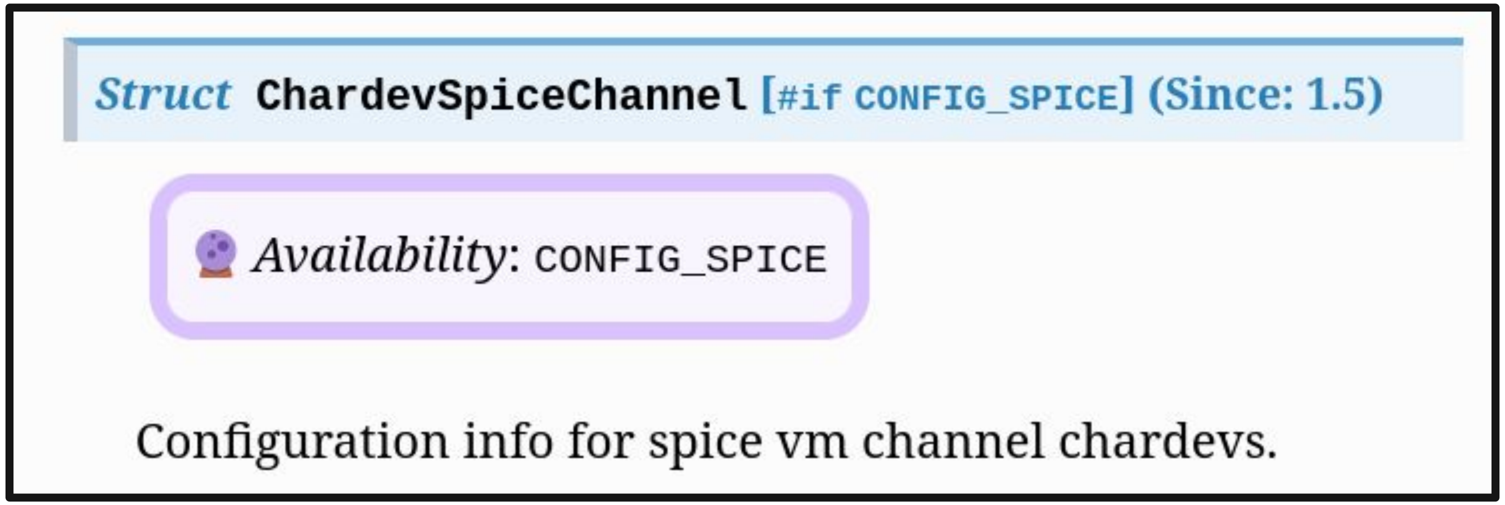
Command `x-blockdev-set-iothread` (Since: 2.12)

🚧 **This command is unstable/experimental.**

The subtitle is a joke because I have been with my lovely wife for *20 years*. Here's to another 20!

What about IFCOND?

I was really undecided on this one 🙃



Struct **ChardevSpiceChannel** [#if CONFIG_SPICE] (Since: 1.5)

Availability: CONFIG_SPICE

Configuration info for spice vm channel chardevs.

I heard from Phil yesterday that he wants to get rid of these. That'd make this a bit easier!

Sphinx version compatibility

- We recently required Sphinx >= 3.4.3 ...
- Compatibility pre-4.1 is still quite rough
    - It’s working, but at a high SLOC cost
    - (And not well-factored in my series)
- Do we have any options here ...?

Sphinx version compatibility

- We recently required Sphinx >= 3.4.3 ...
- Compatibility pre-4.1 is still quite rough
  - It’s working, but at a high SLOC cost
  - (And not well-factored in my series)
- Do we have any options here ...?
  - Please? 😭😭😭

Rapid-fire micro-issues

- Branches:
  - "Details:" placement still WIP
  - Feature flag placement still WIP
- "Members" section amendments
  - "Section-details:" tag?
- "Details:" marker - mandatory? as-needed?

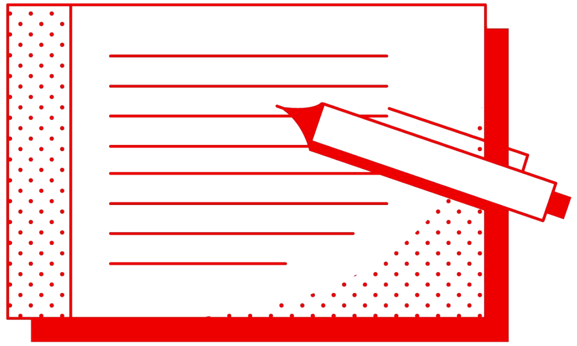
Markus would insist they're not "micro" but they didn't make it into the talk, so how important could they really be?

Future work?

- Inlined alternate syntax?
  - <T | U> instead of cross-refs?
- Inlined return fields?
- Free ponies for everyone?

And probably a bunch of other stuff I've forgotten

Summary



Modified QAPIDoc source format

Source format for developers didn't change much.

New features I want you to use

CROSS-REFERENCES! Both in and outside QAPI.

What to do when stuff explodes

Use DEBUG=1 and look at the *.ir file(s)

(and yell at Markus who will yell at me)

Stuff I want your input on

Layout and design issues for branches, since, ifcond, &c.

Red Hat

I know nobody beyond the front row can read the small text but it’s not for *you*, it’s for *me*

Questions?
Feedback?

I want your feedback!
jsnow@redhat.com
cc: qemu-devel@nongnu.org

Thank you!
Děkuji!

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Goodies: (All linked on pretalx along with these slides!)

WIP: https://gitlab.com/jsnow/qemu/-/commits/sphinx-domain

WIP HTML render/demo: https://jsnow.gitlab.io/qemu/qapi/index.html

Historical QAPI parser & schema: https://gitlab.com/jsnow/externalized-qapi/

Red Hat