



QMP Reference Manual

QEMU version 2.10.50

This is the QEMU QMP reference manual.

Copyright © 2016 The QEMU Project developers

This manual is free documentation: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.

This manual is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this manual. If not, see <http://www.gnu.org/licenses/>.

Table of Contents

1	API Reference	1
1.1	Introduction	1
1.2	Stability Considerations	1
1.3	Common data types	1
1.4	Socket data types	3
1.5	VM run state	5
1.6	Cryptography	10
1.7	Block devices	15
1.7.1	Block core (VM unrelated)	15
1.7.2	Additional block stuff (VM related)	81
1.8	Character devices	87
1.9	Net devices	95
1.10	Rocker switch device	106
1.11	TPM (trusted platform module) devices	112
1.12	Remote desktop	114
1.12.1	Spice	115
1.12.2	VNC	119
1.13	Input	125
1.14	Migration	134
1.15	Transactions	151
1.16	Tracing	153
1.17	QMP introspection	155
1.18	Miscellanea	159
	Commands and Events Index	206
	Data Types Index	209

1 API Reference

1.1 Introduction

This document describes all commands currently supported by QMP.

Most of the time their usage is exactly the same as in the user Monitor, this means that any other document which also describe commands (the manpage, QEMU's manual, etc) can and should be consulted.

QMP has two types of commands: regular and query commands. Regular commands usually change the Virtual Machine's state somehow, while query commands just return information. The sections below are divided accordingly.

It's important to observe that all communication examples are formatted in a reader-friendly way, so that they're easier to understand. However, in real protocol usage, they're emitted as a single line.

Also, the following notation is used to denote data flow:

Example:

-> data issued by the Client

<- Server data response

Please, refer to the QMP specification (docs/interop/qmp-spec.txt) for detailed information on the Server command and response formats.

1.2 Stability Considerations

The current QMP command set (described in this file) may be useful for a number of use cases, however it's limited and several commands have bad defined semantics, specially with regard to command completion.

These problems are going to be solved incrementally in the next QEMU releases and we're going to establish a deprecation policy for badly defined commands.

If you're planning to adopt QMP, please observe the following:

1. The deprecation policy will take effect and be documented soon, please check the documentation of each used command as soon as a new release of QEMU is available
2. DO NOT rely on anything which is not explicit documented
3. Errors, in special, are not documented. Applications should NOT check for specific errors classes or data (it's strongly recommended to only check for the "error" key)

1.3 Common data types

`QapiErrorClass` [Enum]

QEMU error classes

Values:

`GenericError`

this is used for errors that don't require a specific error class. This should be the default case for most errors

CommandNotFound
the requested command has not been found

DeviceNotActive
a device has failed to become active

DeviceNotFound
the requested device has not been found

KVMMissingCap
the requested operation can't be fulfilled because a required KVM capability is missing

Since: 1.2

IoOperationType [Enum]

An enumeration of the I/O operation types

Values:

read read operation

write write operation

Since: 2.1

OnOffAuto [Enum]

An enumeration of three options: on, off, and auto

Values:

auto QEMU selects the value between on and off

on Enabled

off Disabled

Since: 2.2

OnOffSplit [Enum]

An enumeration of three values: on, off, and split

Values:

on Enabled

off Disabled

split Mixed

Since: 2.6

String [Object]

A fat type wrapping 'str', to be embedded in lists.

Members:

str: string
Not documented

Since: 1.2

StrOrNull [Alternate]

This is a string value or the explicit lack of a string (null pointer in C). Intended for cases when 'optional absent' already has a different meaning.

Members:

s: `string` the string value

n: `null` no string value

Since: 2.10

1.4 Socket data types

NetworkAddressFamily [Enum]

The network address family

Values:

`ipv4` IPV4 family

`ipv6` IPV6 family

`unix` unix socket

`vsock` vsock family (since 2.8)

`unknown` otherwise

Since: 2.1

InetSocketAddressBase [Object]

Members:

host: `string`
host part of the address

port: `string`
port part of the address

InetSocketAddress [Object]

Captures a socket address or address range in the Internet namespace.

Members:

numeric: `boolean` (optional)
true if the host/port are guaranteed to be numeric, false if name resolution should be attempted. Defaults to false. (Since 2.9)

to: `int` (optional)
If present, this is range of possible addresses, with port between `port` and `to`.

ipv4: `boolean` (optional)
whether to accept IPv4 addresses, default try both IPv4 and IPv6

ipv6: `boolean` (optional)
whether to accept IPv6 addresses, default try both IPv4 and IPv6

The members of `InetSocketAddressBase`

Since: 1.3

UnixSocketAddress [Object]

Captures a socket address in the local ("Unix socket") namespace.

Members:

`path: string`
filesystem path to use

Since: 1.3

VsockSocketAddress [Object]

Captures a socket address in the vsock namespace.

Members:

`cid: string`
unique host identifier

`port: string`
port

Note: string types are used to allow for possible future hostname or service resolution support.

Since: 2.8

SocketAddressLegacy [Object]

Captures the address of a socket, which could also be a named file descriptor

Members:

`type` One of "inet", "unix", "vsock", "fd"

`data: InetSocketAddress` when type is "inet"

`data: UnixSocketAddress` when type is "unix"

`data: VsockSocketAddress` when type is "vsock"

`data: String` when type is "fd"

Note: This type is deprecated in favor of SocketAddress. The difference between SocketAddressLegacy and SocketAddress is that the latter is a flat union rather than a simple union. Flat is nicer because it avoids nesting on the wire, i.e. that form has fewer {}.

Since: 1.3

SocketAddressType [Enum]

Available SocketAddress types

Values:

`inet` Internet address

`unix` Unix domain socket

`vsock` Not documented

`fd` Not documented

Since: 2.9

SocketAddress [Object]

Captures the address of a socket, which could also be a named file descriptor

Members:

`type: SocketAddressType`
 Transport type

The members of `InetSocketAddress` when `type` is "inet"

The members of `UnixSocketAddress` when `type` is "unix"

The members of `VsockSocketAddress` when `type` is "vsock"

The members of `String` when `type` is "fd"

Since: 2.9

1.5 VM run state

RunState [Enum]

An enumeration of VM run states.

Values:

`debug` QEMU is running on a debugger

`finish-migrate`
 guest is paused to finish the migration process

`inmigrate`
 guest is paused waiting for an incoming migration. Note that this state does not tell whether the machine will start at the end of the migration. This depends on the command-line `-S` option and any invocation of 'stop' or 'cont' that has happened since QEMU was started.

`internal-error`
 An internal error that prevents further guest execution has occurred

`io-error` the last IOP has failed and the device is configured to pause on I/O errors

`paused` guest has been paused via the 'stop' command

`postmigrate`
 guest is paused following a successful 'migrate'

`prelaunch`
 QEMU was started with `-S` and guest has not started

`restore-vm`
 guest is paused to restore VM state

`running` guest is actively running

`save-vm` guest is paused to save the VM state

`shutdown` guest is shut down (and `-no-shutdown` is in use)

`suspended`
 guest is suspended (ACPI S3)

watchdog the watchdog action is configured to pause and has been triggered

guest-panicked
 guest has been panicked as a result of guest OS panic

colo guest is paused to save/restore VM state under colo checkpoint, VM can not get into this state unless colo capability is enabled for migration. (since 2.8)

StatusInfo [Object]

Information about VCPU run state

Members:

running: boolean
 true if all VCPUs are runnable, false if not runnable

singlestep: boolean
 true if VCPUs are in single-step mode

status: RunState
 the virtual machine RunState

Since: 0.14.0

Notes: `singlestep` is enabled through the GDB stub

query-status [Command]

Query the run status of all VCPUs

Returns: StatusInfo reflecting all VCPUs

Since: 0.14.0

Example:

```
-> { "execute": "query-status" }
<- { "return": { "running": true,
                 "singlestep": false,
                 "status": "running" } }
```

SHUTDOWN [Event]

Emitted when the virtual machine has shut down, indicating that qemu is about to exit.

Arguments:

guest: boolean
 If true, the shutdown was triggered by a guest request (such as a guest-initiated ACPI shutdown request or other hardware-specific action) rather than a host request (such as sending qemu a SIGINT). (since 2.10)

Note: If the command-line option `-no-shutdown` has been specified, qemu will not exit, and a STOP event will eventually follow the SHUTDOWN event

Since: 0.12.0

Example:

```
<- { "event": "SHUTDOWN", "data": { "guest": true },
      "timestamp": { "seconds": 1267040730, "microseconds": 682951 } }
```

POWERDOWN [Event]

Emitted when the virtual machine is powered down through the power control system, such as via ACPI.

Since: 0.12.0

Example:

```
<- { "event": "POWERDOWN",  
      "timestamp": { "seconds": 1267040730, "microseconds": 682951 } }
```

RESET [Event]

Emitted when the virtual machine is reset

Arguments:

guest: boolean

If true, the reset was triggered by a guest request (such as a guest-initiated ACPI reboot request or other hardware-specific action) rather than a host request (such as the QMP command `system_reset`). (since 2.10)

Since: 0.12.0

Example:

```
<- { "event": "RESET", "data": { "guest": false },  
      "timestamp": { "seconds": 1267041653, "microseconds": 9518 } }
```

STOP [Event]

Emitted when the virtual machine is stopped

Since: 0.12.0

Example:

```
<- { "event": "STOP",  
      "timestamp": { "seconds": 1267041730, "microseconds": 281295 } }
```

RESUME [Event]

Emitted when the virtual machine resumes execution

Since: 0.12.0

Example:

```
<- { "event": "RESUME",  
      "timestamp": { "seconds": 1271770767, "microseconds": 582542 } }
```

SUSPEND [Event]

Emitted when guest enters a hardware suspension state, for example, S3 state, which is sometimes called standby state

Since: 1.1

Example:

```
<- { "event": "SUSPEND",  
      "timestamp": { "seconds": 1344456160, "microseconds": 309119 } }
```

SUSPEND_DISK [Event]

Emitted when guest enters a hardware suspension state with data saved on disk, for example, S4 state, which is sometimes called hibernate state

Note: QEMU shuts down (similar to event SHUTDOWN) when entering this state

Since: 1.2

Example:

```
<- { "event": "SUSPEND_DISK",
      "timestamp": { "seconds": 1344456160, "microseconds": 309119 } }
```

WAKEUP [Event]

Emitted when the guest has woken up from suspend state and is running

Since: 1.1

Example:

```
<- { "event": "WAKEUP",
      "timestamp": { "seconds": 1344522075, "microseconds": 745528 } }
```

WATCHDOG [Event]

Emitted when the watchdog device's timer is expired

Arguments:

action: WatchdogExpirationAction
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

Note: This event is rate-limited.

Since: 0.13.0

Example:

```
<- { "event": "WATCHDOG",
      "data": { "action": "reset" },
      "timestamp": { "seconds": 1267061043, "microseconds": 959568 } }
```

WatchdogExpirationAction [Enum]

An enumeration of the actions taken when the watchdog device's timer is expired

Values:

reset	system resets
shutdown	system shutdown, note that it is similar to <code>powerdown</code> , which tries to set to system status and notify guest
poweroff	system poweroff, the emulator program exits
pause	system pauses, similar to <code>stop</code>
debug	system enters debug state
none	nothing is done

inject-nmi

a non-maskable interrupt is injected into the first VCPU (all VCPUS on x86) (since 2.4)

Since: 2.1

GUEST_PANICKED

[Event]

Emitted when guest OS panic is detected

Arguments:

action: `GuestPanicAction`

action that has been taken, currently always "pause"

info: `GuestPanicInformation` (optional)

information about a panic (since 2.9)

Since: 1.5

Example:

```
<- { "event": "GUEST_PANICKED",
      "data": { "action": "pause" } }
```

GuestPanicAction

[Enum]

An enumeration of the actions taken when guest OS panic is detected

Values:

`pause` system pauses

`poweroff` Not documented

Since: 2.1 (poweroff since 2.8)

GuestPanicInformationType

[Enum]

An enumeration of the guest panic information types

Values:

`hyper-v` Not documented

Since: 2.9

GuestPanicInformation

[Object]

Information about a guest panic

Members:

type: `GuestPanicInformationType`

Not documented

The members of `GuestPanicInformationHyperV` when type is "hyper-v"

Since: 2.9

GuestPanicInformationHyperV

[Object]

Hyper-V specific guest panic information (HV crash MSRs)

Members:

arg1: `int` Not documented

arg2: int Not documented
arg3: int Not documented
arg4: int Not documented
arg5: int Not documented
Since: 2.9

1.6 Cryptography

QCryptoTLSCredsEndpoint [Enum]

The type of network endpoint that will be using the credentials. Most types of credential require different setup / structures depending on whether they will be used in a server versus a client.

Values:

client the network endpoint is acting as the client
server the network endpoint is acting as the server

Since: 2.5

QCryptoSecretFormat [Enum]

The data format that the secret is provided in

Values:

raw raw bytes. When encoded in JSON only valid UTF-8 sequences can be used
base64 arbitrary base64 encoded binary data

Since: 2.6

QCryptoHashAlgorithm [Enum]

The supported algorithms for computing content digests

Values:

md5 MD5. Should not be used in any new code, legacy compat only
sha1 SHA-1. Should not be used in any new code, legacy compat only
sha224 SHA-224. (since 2.7)
sha256 SHA-256. Current recommended strong hash.
sha384 SHA-384. (since 2.7)
sha512 SHA-512. (since 2.7)
ripemd160 RIPEMD-160. (since 2.7)

Since: 2.6

QCryptoCipherAlgorithm [Enum]

The supported algorithms for content encryption ciphers

Values:

<code>aes-128</code>	AES with 128 bit / 16 byte keys
<code>aes-192</code>	AES with 192 bit / 24 byte keys
<code>aes-256</code>	AES with 256 bit / 32 byte keys
<code>des-rfb</code>	RFB specific variant of single DES. Do not use except in VNC.
<code>3des</code>	3DES(EDE) with 192 bit / 24 byte keys (since 2.9)
<code>cast5-128</code>	Cast5 with 128 bit / 16 byte keys
<code>serpent-128</code>	Serpent with 128 bit / 16 byte keys
<code>serpent-192</code>	Serpent with 192 bit / 24 byte keys
<code>serpent-256</code>	Serpent with 256 bit / 32 byte keys
<code>twofish-128</code>	Twofish with 128 bit / 16 byte keys
<code>twofish-192</code>	Twofish with 192 bit / 24 byte keys
<code>twofish-256</code>	Twofish with 256 bit / 32 byte keys

Since: 2.6

QCryptoCipherMode [Enum]

The supported modes for content encryption ciphers

Values:

<code>ecb</code>	Electronic Code Book
<code>cbc</code>	Cipher Block Chaining
<code>xts</code>	XEX with tweaked code book and ciphertext stealing
<code>ctr</code>	Counter (Since 2.8)

Since: 2.6

QCryptoIVGenAlgorithm [Enum]

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:

<code>plain</code>	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
Since: 2.6

QCryptoBlockFormat [Enum]

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

Since: 2.6

QCryptoBlockOptionsBase [Object]

The common options that apply to all full disk encryption formats

Members:

`format`: `QCryptoBlockFormat`
the encryption format

Since: 2.6

QCryptoBlockOptionsQCow [Object]

The options that apply to QCow/QCow2 AES-CBC encryption format

Members:

`key-secret`: `string` (optional)
the ID of a `QCryptoSecret` object providing the decryption key. Mandatory except when probing image for metadata only.

Since: 2.6

QCryptoBlockOptionsLUKS [Object]

The options that apply to LUKS encryption format

Members:

`key-secret`: `string` (optional)
the ID of a `QCryptoSecret` object providing the decryption key. Mandatory except when probing image for metadata only.

Since: 2.6

QCryptoBlockCreateOptionsLUKS [Object]

The options that apply to LUKS encryption format initialization

Members:

`cipher-alg`: `QCryptoCipherAlgorithm` (optional)
the cipher algorithm for data encryption Currently defaults to 'aes'.

`cipher-mode`: `QCryptoCipherMode` (optional)
the cipher mode for data encryption Currently defaults to 'cbc'

`ivgen-alg`: `QCryptoIVGenAlgorithm` (optional)
the initialization vector generator Currently defaults to 'essiv'

`ivgen-hash-alg`: `QCryptoHashAlgorithm` (optional)
the initialization vector generator hash Currently defaults to 'sha256'

`hash-alg`: `QCryptoHashAlgorithm` (optional)
the master key hash algorithm Currently defaults to 'sha256'

`iter-time`: `int` (optional)
number of milliseconds to spend in PBKDF passphrase processing. Currently defaults to 2000. (since 2.8)

The members of `QCryptoBlockOptionsLUKS`

Since: 2.6

`QCryptoBlockOpenOptions` [Object]

The options that are available for all encryption formats when opening an existing volume

Members:

The members of `QCryptoBlockOptionsBase`

The members of `QCryptoBlockOptionsQcow` when format is "qcow"

The members of `QCryptoBlockOptionsLUKS` when format is "luks"

Since: 2.6

`QCryptoBlockCreateOptions` [Object]

The options that are available for all encryption formats when initializing a new volume

Members:

The members of `QCryptoBlockOptionsBase`

The members of `QCryptoBlockOptionsQcow` when format is "qcow"

The members of `QCryptoBlockCreateOptionsLUKS` when format is "luks"

Since: 2.6

`QCryptoBlockInfoBase` [Object]

The common information that applies to all full disk encryption formats

Members:

`format`: `QCryptoBlockFormat`
the encryption format

Since: 2.7

`QCryptoBlockInfoLUKSSlot` [Object]

Information about the LUKS block encryption key slot options

Members:

`active`: `boolean`
whether the key slot is currently in use

key-offset: int
 offset to the key material in bytes

iters: int (optional)
 number of PBKDF2 iterations for key material

stripes: int (optional)
 number of stripes for splitting key material

Since: 2.7

QCryptoBlockInfoLUKS [Object]

Information about the LUKS block encryption options

Members:

cipher-arg: QCryptoCipherAlgorithm
 the cipher algorithm for data encryption

cipher-mode: QCryptoCipherMode
 the cipher mode for data encryption

ivgen-arg: QCryptoIVGenAlgorithm
 the initialization vector generator

ivgen-hash-arg: QCryptoHashAlgorithm (optional)
 the initialization vector generator hash

hash-arg: QCryptoHashAlgorithm
 the master key hash algorithm

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: array of QCryptoBlockInfoLUKSSlot
 information about each key slot

Since: 2.7

QCryptoBlockInfoQcow [Object]

Information about the Qcow block encryption options

Since: 2.7

QCryptoBlockInfo [Object]

Information about the block encryption options

Members:

The members of `QCryptoBlockInfoBase`
 The members of `QCryptoBlockInfoQcow` when `format` is "qcow"
 The members of `QCryptoBlockInfoLUKS` when `format` is "luks"

Since: 2.7

1.7 Block devices

1.7.1 Block core (VM unrelated)

SnapshotInfo [Object]

Members:

id: string
unique snapshot id

name: string
user chosen name

vm-state-size: int
size of the VM state

date-sec: int
UTC date of the snapshot in seconds

date-nsec: int
fractional part in nano seconds to be used with date-sec

vm-clock-sec: int
VM clock relative to boot in seconds

vm-clock-nsec: int
fractional part in nano seconds to be used with vm-clock-sec

Since: 1.3

ImageInfoSpecificQCow2EncryptionBase [Object]

Members:

format: BlockdevQcow2EncryptionFormat
The encryption format

Since: 2.10

ImageInfoSpecificQCow2Encryption [Object]

Members:

The members of `ImageInfoSpecificQCow2EncryptionBase`
The members of `QCryptoBlockInfoQCow` when `format` is "aes"
The members of `QCryptoBlockInfoLUKS` when `format` is "luks"

Since: 2.10

ImageInfoSpecificQCow2 [Object]

Members:

compat: string
compatibility level

lazy-refcounts: boolean (optional)
on or off; only valid for `compat` \geq 1.1

corrupt: `boolean` (optional)
true if the image has been marked corrupt; only valid for `compat` \geq 1.1 (since 2.2)

refcount-bits: `int`
width of a `refcount` entry in bits (since 2.3)

encrypt: `ImageInfoSpecificQCow2Encryption` (optional)
details about encryption parameters; only set if image is encrypted (since 2.10)

Since: 1.7

ImageInfoSpecificVmdk [Object]

Members:

create-type: `string`
The create type of VMDK image

cid: `int` Content id of image

parent-cid: `int`
Parent VMDK image's cid

extents: `array of ImageInfo`
List of extent files

Since: 1.7

ImageInfoSpecific [Object]

A discriminated record of image format specific information structures.

Members:

type One of "qcow2", "vmdk", "luks"

data: `ImageInfoSpecificQCow2` when `type` is "qcow2"

data: `ImageInfoSpecificVmdk` when `type` is "vmdk"

data: `QCryptoBlockInfoLUKS` when `type` is "luks"

Since: 1.7

ImageInfo [Object]

Information about a QEMU image file

Members:

filename: `string`
name of the image file

format: `string`
format of the image file

virtual-size: `int`
maximum capacity in bytes of the image

actual-size: `int` (optional)
actual size on disk in bytes of the image

dirty-flag: `boolean` (optional)
true if image is not cleanly closed

cluster-size: `int` (optional)
size of a cluster in bytes

encrypted: `boolean` (optional)
true if the image is encrypted

compressed: `boolean` (optional)
true if the image is compressed (Since 1.7)

backing-filename: `string` (optional)
name of the backing file

full-backing-filename: `string` (optional)
full path of the backing file

backing-filename-format: `string` (optional)
the format of the backing file

snapshots: array of `SnapshotInfo` (optional)
list of VM snapshots

backing-image: `ImageInfo` (optional)
info of the backing image (since 1.6)

format-specific: `ImageInfoSpecific` (optional)
structure supplying additional format-specific information (since 1.7)

Since: 1.3

ImageCheck [Object]

Information about a QEMU image file check

Members:

filename: `string`
name of the image file checked

format: `string`
format of the image file checked

check-errors: `int`
number of unexpected errors occurred during check

image-end-offset: `int` (optional)
offset (in bytes) where the image ends, this field is present if the driver for the image format supports it

corruptions: `int` (optional)
number of corruptions found during the check if any

leaks: `int` (optional)
number of leaks found during the check if any

corruptions-fixed: `int` (optional)
number of corruptions fixed during the check if any

leaks-fixed: `int` (optional)
 number of leaks fixed during the check if any

total-clusters: `int` (optional)
 total number of clusters, this field is present if the driver for the image format supports it

allocated-clusters: `int` (optional)
 total number of allocated clusters, this field is present if the driver for the image format supports it

fragmented-clusters: `int` (optional)
 total number of fragmented clusters, this field is present if the driver for the image format supports it

compressed-clusters: `int` (optional)
 total number of compressed clusters, this field is present if the driver for the image format supports it

Since: 1.4

MapEntry [Object]

Mapping information from a virtual block range to a host file range

Members:

start: `int`
 the start byte of the mapped virtual range

length: `int`
 the number of bytes of the mapped virtual range

data: `boolean`
 whether the mapped range has data

zero: `boolean`
 whether the virtual blocks are zeroed

depth: `int`
 the depth of the mapping

offset: `int` (optional)
 the offset in file that the virtual sectors are mapped to

filename: `string` (optional)
 filename that is referred to by `offset`

Since: 2.6

BlockdevCacheInfo [Object]

Cache mode information for a block device

Members:

writeback: `boolean`
 true if writeback mode is enabled

direct: boolean
true if the host page cache is bypassed (O_DIRECT)

no-flush: boolean
true if flush requests are ignored for the device

Since: 2.3

BlockDeviceInfo [Object]

Information about the backing device for a block device.

Members:

file: string
the filename of the backing device

node-name: string (optional)
the name of the block driver node (Since 2.0)

ro: boolean
true if the backing device was open read-only

drv: string
the name of the block format used to open the backing device. As of 0.14.0 this can be: 'blkdebug', 'bochs', 'cloop', 'cow', 'dmg', 'file', 'file', 'ftp', 'ftps', 'host_cdrom', 'host_device', 'http', 'https', 'luks', 'nbd', 'parallels', 'qcow', 'qcow2', 'raw', 'vdi', 'vmdk', 'vpc', 'vvfat' 2.2: 'archipelago' added, 'cow' dropped 2.3: 'host_floppy' deprecated 2.5: 'host_floppy' dropped 2.6: 'luks' added 2.8: 'replication' added, 'tftp' dropped 2.9: 'archipelago' dropped

backing_file: string (optional)
the name of the backing file (for copy-on-write)

backing_file_depth: int
number of files in the backing file chain (since: 1.2)

encrypted: boolean
true if the backing device is encrypted

encryption_key_missing: boolean
Deprecated; always false

detect_zeroes: BlockdevDetectZeroesOptions
detect and optimize zero writes (Since 2.1)

bps: int total throughput limit in bytes per second is specified

bps_rd: int
read throughput limit in bytes per second is specified

bps_wr: int
write throughput limit in bytes per second is specified

iops: int total I/O operations per second is specified

iops_rd: int
read I/O operations per second is specified

`iops_wr`: `int`
write I/O operations per second is specified

`image`: `ImageInfo`
the info of image used (since: 1.6)

`bps_max`: `int` (optional)
total throughput limit during bursts, in bytes (Since 1.7)

`bps_rd_max`: `int` (optional)
read throughput limit during bursts, in bytes (Since 1.7)

`bps_wr_max`: `int` (optional)
write throughput limit during bursts, in bytes (Since 1.7)

`iops_max`: `int` (optional)
total I/O operations per second during bursts, in bytes (Since 1.7)

`iops_rd_max`: `int` (optional)
read I/O operations per second during bursts, in bytes (Since 1.7)

`iops_wr_max`: `int` (optional)
write I/O operations per second during bursts, in bytes (Since 1.7)

`bps_max_length`: `int` (optional)
maximum length of the `bps_max` burst period, in seconds. (Since 2.6)

`bps_rd_max_length`: `int` (optional)
maximum length of the `bps_rd_max` burst period, in seconds. (Since 2.6)

`bps_wr_max_length`: `int` (optional)
maximum length of the `bps_wr_max` burst period, in seconds. (Since 2.6)

`iops_max_length`: `int` (optional)
maximum length of the `iops` burst period, in seconds. (Since 2.6)

`iops_rd_max_length`: `int` (optional)
maximum length of the `iops_rd_max` burst period, in seconds. (Since 2.6)

`iops_wr_max_length`: `int` (optional)
maximum length of the `iops_wr_max` burst period, in seconds. (Since 2.6)

`iops_size`: `int` (optional)
an I/O size in bytes (Since 1.7)

`group`: `string` (optional)
throttle group name (Since 2.4)

`cache`: `BlockdevCacheInfo`
the cache mode used for the block device (since: 2.3)

`write_threshold`: `int`
configured write threshold for the device. 0 if disabled. (Since 2.3)

Since: 0.14.0

BlockDeviceIoStatus [Enum]

An enumeration of block device I/O status.

Values:

- ok** The last I/O operation has succeeded
- failed** The last I/O operation has failed
- nospace** The last I/O operation has failed due to a no-space condition

Since: 1.0

BlockDeviceMapEntry [Object]

Entry in the metadata map of the device (returned by "qemu-img map")

Members:

- start: int**
Offset in the image of the first byte described by this entry (in bytes)
- length: int**
Length of the range described by this entry (in bytes)
- depth: int**
Number of layers (0 = top image, 1 = top image's backing file, etc.) before reaching one for which the range is allocated. The value is in the range 0 to the depth of the image chain - 1.
- zero: boolean**
the sectors in this range read as zeros
- data: boolean**
reading the image will actually read data from a file (in particular, if **offset** is present this means that the sectors are not simply preallocated, but contain actual data in raw format)
- offset: int (optional)**
if present, the image file stores the data for this range in raw format at the given offset.

Since: 1.7

DirtyBitmapStatus [Enum]

An enumeration of possible states that a dirty bitmap can report to the user.

Values:

- frozen** The bitmap is currently in-use by a backup operation or block job, and is immutable.
- disabled** The bitmap is currently in-use by an internal operation and is read-only. It can still be deleted.
- active** The bitmap is actively monitoring for new writes, and can be cleared, deleted, or used for backup operations.

Since: 2.4

BlockDirtyInfo [Object]

Block dirty bitmap information.

Members:

name: string (optional)
the name of the dirty bitmap (Since 2.4)

count: int
number of dirty bytes according to the dirty bitmap

granularity: int
granularity of the dirty bitmap in bytes (since 1.4)

status: DirtyBitmapStatus
current status of the dirty bitmap (since 2.4)

Since: 1.3

BlockInfo [Object]

Block device information. This structure describes a virtual device and the backing device associated with it.

Members:

device: string
The device name associated with the virtual device.

qdev: string (optional)
The qdev ID, or if no ID is assigned, the QOM path of the block device. (since 2.10)

type: string
This field is returned only for compatibility reasons, it should not be used (always returns 'unknown')

removable: boolean
True if the device supports removable media.

locked: boolean
True if the guest has locked this device from having its media removed

tray_open: boolean (optional)
True if the device's tray is open (only present if it has a tray)

dirty-bitmaps: array of BlockDirtyInfo (optional)
dirty bitmaps information (only present if the driver has one or more dirty bitmaps) (Since 2.0)

io-status: BlockDeviceIoStatus (optional)
BlockDeviceIoStatus. Only present if the device supports it and the VM is configured to stop on errors (supported device models: virtio-blk, IDE, SCSI except scsi-generic)

inserted: BlockDeviceInfo (optional)
BlockDeviceInfo describing the device if media is present

Since: 0.14.0

BlockMeasureInfo [Object]

Image file size calculation information. This structure describes the size requirements for creating a new image file.

The size requirements depend on the new image file format. File size always equals virtual disk size for the 'raw' format, even for sparse POSIX files. Compact formats such as 'qcow2' represent unallocated and zero regions efficiently so file size may be smaller than virtual disk size.

The values are upper bounds that are guaranteed to fit the new image file. Subsequent modification, such as internal snapshot or bitmap creation, may require additional space and is not covered here.

Members:

required: int

Size required for a new image file, in bytes.

fully-allocated: int

Image file size, in bytes, once data has been written to all sectors.

Since: 2.10

query-block [Command]

Get a list of BlockInfo for all virtual block devices.

Returns: a list of BlockInfo describing each virtual block device. Filter nodes that were created implicitly are skipped over.

Since: 0.14.0

Example:

```
-> { "execute": "query-block" }
<- {
  "return": [
    {
      "io-status": "ok",
      "device": "ide0-hd0",
      "locked": false,
      "removable": false,
      "inserted": {
        "ro": false,
        "drv": "qcow2",
        "encrypted": false,
        "file": "disks/test.qcow2",
        "backing_file_depth": 1,
        "bps": 1000000,
        "bps_rd": 0,
        "bps_wr": 0,
        "iops": 1000000,
        "iops_rd": 0,
        "iops_wr": 0,
        "bps_max": 8000000,
      }
    }
  ]
}
```

```

    "bps_rd_max": 0,
    "bps_wr_max": 0,
    "iops_max": 0,
    "iops_rd_max": 0,
    "iops_wr_max": 0,
    "iops_size": 0,
    "detect_zeroes": "on",
    "write_threshold": 0,
    "image":{
      "filename":"disks/test.qcow2",
      "format":"qcow2",
      "virtual-size":2048000,
      "backing_file":"base.qcow2",
      "full-backing-filename":"disks/base.qcow2",
      "backing-filename-format":"qcow2",
      "snapshots":[
        {
          "id": "1",
          "name": "snapshot1",
          "vm-state-size": 0,
          "date-sec": 10000200,
          "date-nsec": 12,
          "vm-clock-sec": 206,
          "vm-clock-nsec": 30
        }
      ],
      "backing-image":{
        "filename":"disks/base.qcow2",
        "format":"qcow2",
        "virtual-size":2048000
      }
    }
  },
  "qdev": "ide_disk",
  "type":"unknown"
},
{
  "io-status": "ok",
  "device":"ide1-cd0",
  "locked":false,
  "removable":true,
  "qdev": "/machine/unattached/device[23]",
  "tray_open": false,
  "type":"unknown"
},
{
  "device":"floppy0",

```

```

        "locked":false,
        "removable":true,
        "qdev": "/machine/unattached/device[20]",
        "type":"unknown"
    },
    {
        "device":"sd0",
        "locked":false,
        "removable":true,
        "type":"unknown"
    }
]
}

```

BlockDeviceTimedStats

[Object]

Statistics of a block device during a given interval of time.

Members:

interval_length: int

Interval used for calculating the statistics, in seconds.

min_rd_latency_ns: int

Minimum latency of read operations in the defined interval, in nanoseconds.

min_wr_latency_ns: int

Minimum latency of write operations in the defined interval, in nanoseconds.

min_flush_latency_ns: int

Minimum latency of flush operations in the defined interval, in nanoseconds.

max_rd_latency_ns: int

Maximum latency of read operations in the defined interval, in nanoseconds.

max_wr_latency_ns: int

Maximum latency of write operations in the defined interval, in nanoseconds.

max_flush_latency_ns: int

Maximum latency of flush operations in the defined interval, in nanoseconds.

avg_rd_latency_ns: int

Average latency of read operations in the defined interval, in nanoseconds.

avg_wr_latency_ns: int

Average latency of write operations in the defined interval, in nanoseconds.

avg_flush_latency_ns: int
Average latency of flush operations in the defined interval, in nanoseconds.

avg_rd_queue_depth: number
Average number of pending read operations in the defined interval.

avg_wr_queue_depth: number
Average number of pending write operations in the defined interval.

Since: 2.5

BlockDeviceStats [Object]

Statistics of a virtual block device or a block backing device.

Members:

rd_bytes: int
The number of bytes read by the device.

wr_bytes: int
The number of bytes written by the device.

rd_operations: int
The number of read operations performed by the device.

wr_operations: int
The number of write operations performed by the device.

flush_operations: int
The number of cache flush operations performed by the device (since 0.15.0)

flush_total_time_ns: int
Total time spend on cache flushes in nano-seconds (since 0.15.0).

wr_total_time_ns: int
Total time spend on writes in nano-seconds (since 0.15.0).

rd_total_time_ns: int
Total time spend on reads in nano-seconds (since 0.15.0).

wr_highest_offset: int
The offset after the greatest byte written to the device. The intended use of this information is for growable sparse files (like qcow2) that are used on top of a physical device.

rd_merged: int
Number of read requests that have been merged into another request (Since 2.3).

wr_merged: int
Number of write requests that have been merged into another request (Since 2.3).

idle_time_ns: `int` (optional)
Time since the last I/O operation, in nanoseconds. If the field is absent it means that there haven't been any operations yet (Since 2.5).

failed_rd_operations: `int`
The number of failed read operations performed by the device (Since 2.5)

failed_wr_operations: `int`
The number of failed write operations performed by the device (Since 2.5)

failed_flush_operations: `int`
The number of failed flush operations performed by the device (Since 2.5)

invalid_rd_operations: `int`
The number of invalid read operations performed by the device (Since 2.5)

invalid_wr_operations: `int`
The number of invalid write operations performed by the device (Since 2.5)

invalid_flush_operations: `int`
The number of invalid flush operations performed by the device (Since 2.5)

account_invalid: `boolean`
Whether invalid operations are included in the last access statistics (Since 2.5)

account_failed: `boolean`
Whether failed operations are included in the latency and last access statistics (Since 2.5)

timed_stats: `array of BlockDeviceTimedStats`
Statistics specific to the set of previously defined intervals of time (Since 2.5)

Since: 0.14.0

BlockStats [Object]

Statistics of a virtual block device or a block backing device.

Members:

device: `string` (optional)
If the stats are for a virtual block device, the name corresponding to the virtual block device.

node-name: `string` (optional)
The node name of the device. (Since 2.3)

stats: `BlockDeviceStats`
A `BlockDeviceStats` for the device.

parent: `BlockStats` (optional)

This describes the file block device if it has one. Contains recursively the statistics of the underlying protocol (e.g. the host file for a qcow2 image). If there is no underlying protocol, this field is omitted

backing: `BlockStats` (optional)

This describes the backing block device if it has one. (Since 2.0)

Since: 0.14.0

`query-blockstats`

[Command]

Query the `BlockStats` for all virtual block devices.

Arguments:

query-nodes: `boolean` (optional)

If true, the command will query all the block nodes that have a node name, in a list which will include "parent" information, but not "backing". If false or omitted, the behavior is as before - query all the device backends, recursively including their "parent" and "backing". Filter nodes that were created implicitly are skipped over in this mode. (Since 2.3)

Returns: A list of `BlockStats` for each virtual block devices.

Since: 0.14.0

Example:

```
-> { "execute": "query-blockstats" }
<- {
  "return": [
    {
      "device": "ide0-hd0",
      "parent": {
        "stats": {
          "wr_highest_offset": 3686448128,
          "wr_bytes": 9786368,
          "wr_operations": 751,
          "rd_bytes": 122567168,
          "rd_operations": 36772,
          "wr_total_times_ns": 313253456,
          "rd_total_times_ns": 3465673657,
          "flush_total_times_ns": 49653,
          "flush_operations": 61,
          "rd_merged": 0,
          "wr_merged": 0,
          "idle_time_ns": 2953431879,
          "account_invalid": true,
          "account_failed": false
        }
      }
    }
  ],
}
```

```
    "stats":{
      "wr_highest_offset":2821110784,
      "wr_bytes":9786368,
      "wr_operations":692,
      "rd_bytes":122739200,
      "rd_operations":36604
      "flush_operations":51,
      "wr_total_times_ns":313253456
      "rd_total_times_ns":3465673657
      "flush_total_times_ns":49653,
      "rd_merged":0,
      "wr_merged":0,
      "idle_time_ns":2953431879,
      "account_invalid":true,
      "account_failed":false
    }
  },
  {
    "device":"ide1-cd0",
    "stats":{
      "wr_highest_offset":0,
      "wr_bytes":0,
      "wr_operations":0,
      "rd_bytes":0,
      "rd_operations":0
      "flush_operations":0,
      "wr_total_times_ns":0
      "rd_total_times_ns":0
      "flush_total_times_ns":0,
      "rd_merged":0,
      "wr_merged":0,
      "account_invalid":false,
      "account_failed":false
    }
  },
  {
    "device":"floppy0",
    "stats":{
      "wr_highest_offset":0,
      "wr_bytes":0,
      "wr_operations":0,
      "rd_bytes":0,
      "rd_operations":0
      "flush_operations":0,
      "wr_total_times_ns":0
      "rd_total_times_ns":0
      "flush_total_times_ns":0,
```



```

        "rd_merged":0,
        "wr_merged":0,
        "account_invalid":false,
        "account_failed":false
    }
},
{
    "device":"sd0",
    "stats":{
        "wr_highest_offset":0,
        "wr_bytes":0,
        "wr_operations":0,
        "rd_bytes":0,
        "rd_operations":0
        "flush_operations":0,
        "wr_total_times_ns":0
        "rd_total_times_ns":0
        "flush_total_times_ns":0,
        "rd_merged":0,
        "wr_merged":0,
        "account_invalid":false,
        "account_failed":false
    }
}
]
}

```

BlockdevOnError [Enum]

An enumeration of possible behaviors for errors on I/O operations. The exact meaning depends on whether the I/O was initiated by a guest or by a block job

Values:

- report** for guest operations, report the error to the guest; for jobs, cancel the job
- ignore** ignore the error, only report a QMP event (BLOCK_IO_ERROR or BLOCK_JOB_ERROR)
- enospc** same as **stop** on ENOSPC, same as **report** otherwise.
- stop** for guest operations, stop the virtual machine; for jobs, pause the job
- auto** inherit the error handling policy of the backend (since: 2.7)

Since: 1.3

MirrorSyncMode [Enum]

An enumeration of possible behaviors for the initial synchronization phase of storage mirroring.

Values:

- top** copies data in the topmost image to the destination

full copies data from all images to the destination
none only copy data written from now on
incremental only copy data described by the dirty bitmap. Since: 2.4

Since: 1.3

BlockJobType [Enum]

Type of a block 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"

Since: 1.7

BlockJobInfo [Object]

Information about a long-running block device operation.

Members:

type: `string`
the job type ('stream' for image streaming)

device: `string`
The job identifier. Originally the device name but other values are allowed since QEMU 2.7

len: `int` the maximum progress value

busy: `boolean`
false if the job is known to be in a quiescent state, with no pending I/O. Since 1.3.

paused: `boolean`
whether the job is paused or, if `busy` is true, will pause itself as soon as possible. Since 1.3.

offset: `int`
the current progress value

speed: `int`
the rate limit, bytes per second

io-status: `BlockDeviceIoStatus`
the status of the job (since 1.3)

ready: `boolean`
true if the job may be completed (since 2.2)

Since: 1.1

`query-block-jobs` [Command]

Return information about long-running block device operations.

Returns: a list of `BlockJobInfo` for each active block job

Since: 1.1

`block_passwd` [Command]

This command sets the password of a block device that has not been open with a password and requires one.

This command is now obsolete and will always return an error since 2.10

Arguments:

`device: string` (optional)
Not documented

`node-name: string` (optional)
Not documented

`password: string`
Not documented

`block_resize` [Command]

Resize a block image while a guest is running.

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

Arguments:

`device: string` (optional)
the name of the device to get the image resized

`node-name: string` (optional)
graph node name to get the image resized (Since 2.0)

`size: int` new image size in bytes

Returns: nothing on success If `device` is not a valid block device, `DeviceNotFound`

Since: 0.14.0

Example:

```
-> { "execute": "block_resize",
      "arguments": { "device": "scratch", "size": 1073741824 } }
<- { "return": {} }
```

`NewImageMode` [Enum]

An enumeration that tells QEMU how to set the backing file path in a new image file.

Values:

`existing` QEMU should look for an existing image file.

`absolute-paths`
QEMU should create a new image with absolute paths for the backing file. If there is no backing file available, the new image will not be backed either.

Since: 1.1

BlockdevSnapshotSync [Object]

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

Members:

`device: string` (optional)

the name of the device to generate the snapshot from.

`node-name: string` (optional)

graph node name to generate the snapshot from (Since 2.0)

`snapshot-file: string`

the target of the new image. If the file exists, or if it is a device, the snapshot 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 snapshot image, default is 'qcow2'.

`mode: NewImageMode` (optional)

whether and how QEMU should create a new image, default is 'absolute-paths'.

BlockdevSnapshot [Object]

Members:

`node: string`

device or node name that will have a snapshot created.

`overlay: string`

reference to the existing block device that will become the overlay of `node`, as part of creating the snapshot. It must not have a current backing file (this can be achieved by passing "backing": "" to `blockdev-add`).

Since: 2.5

DriveBackup [Object]

Members:

`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.

`target: string`

the target of the new image. If the file exists, or if it is a device, the existing file/device will be used as the new destination. If it does not exist, a new file will be created.

`format: string` (optional)

the format of the new destination, default is to probe if `mode` is 'existing', else the format of the source

- 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).
- mode:** `NewImageMode` (optional)
 whether and how QEMU should create a new image, default is 'absolute-paths'.
- speed:** `int` (optional)
 the maximum speed, in bytes per second
- bitmap:** `string` (optional)
 the name of dirty bitmap if sync is "incremental". Must be present if sync is "incremental", must NOT be present otherwise. (Since 2.4)
- compress:** `boolean` (optional)
 true to compress data, if the target format supports it. (default: false) (since 2.8)
- on-source-error:** `BlockdevOnError` (optional)
 the action to take on an error on the source, default 'report'. 'stop' and 'enospc' can only be used if the block device supports io-status (see `BlockInfo`).
- on-target-error:** `BlockdevOnError` (optional)
 the action to take on an error on the target, default 'report' (no limitations, since this applies to a different block device than `device`).

Note: `on-source-error` and `on-target-error` only affect background I/O. If an error occurs during a guest write request, the device's `rerror/werror` actions will be used.

Since: 1.6

`BlockdevBackup` [Object]

Members:

- 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.
- target:** `string`
 the device name or node-name of the backup target node.
- 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, or only new I/O).
- speed:** `int` (optional)
 the maximum speed, in bytes per second. The default is 0, for unlimited.

compress: `boolean` (optional)
 true to compress data, if the target format supports it. (default: `false`)
 (since 2.8)

on-source-error: `BlockdevOnError` (optional)
 the action to take on an error on the source, default `'report'`. `'stop'`
 and `'enospc'` can only be used if the block device supports `io-status` (see
`BlockInfo`).

on-target-error: `BlockdevOnError` (optional)
 the action to take on an error on the target, default `'report'` (no limita-
 tions, since this applies to a different block device than `device`).

Note: `on-source-error` and `on-target-error` only affect background I/O. If an
 error occurs during a guest write request, the device's `rerror/werror` actions will be
 used.

Since: 2.3

`blockdev-snapshot-sync` [Command]

Generates a synchronous snapshot of a block device.

For the arguments, see the documentation of `BlockdevSnapshotSync`.

Returns: nothing on success If `device` is not a valid block device, `DeviceNotFound`

Since: 0.14.0

Example:

```
-> { "execute": "blockdev-snapshot-sync",
      "arguments": { "device": "ide-hd0",
                    "snapshot-file":
                      "/some/place/my-image",
                    "format": "qcow2" } }

<- { "return": {} }
```

`blockdev-snapshot` [Command]

Generates a snapshot of a block device.

Create a snapshot, by installing `'node'` as the backing image of `'overlay'`. Additionally,
 if `'node'` is associated with a block device, the block device changes to using `'overlay'`
 as its new active image.

For the arguments, see the documentation of `BlockdevSnapshot`.

Since: 2.5

Example:

```
-> { "execute": "blockdev-add",
      "arguments": { "driver": "qcow2",
                    "node-name": "node1534",
                    "file": { "driver": "file",
                              "filename": "hd1.qcow2" },
                    "backing": "" } }

<- { "return": {} }
```

```
-> { "execute": "blockdev-snapshot",
      "arguments": { "node": "ide-hd0",
                    "overlay": "node1534" } }
<- { "return": {} }
```

change-backing-file [Command]

Change the backing file in the image file metadata. This does not cause QEMU to reopen the image file to reparse the backing filename (it may, however, perform a reopen to change permissions from r/o -> r/w -> r/o, if needed). The new backing file string is written into the image file metadata, and the QEMU internal strings are updated.

Arguments:

image-node-name: string

The name of the block driver state node of the image to modify. The "device" argument is used to verify "image-node-name" is in the chain described by "device".

device: string

The device name or node-name of the root node that owns image-node-name.

backing-file: string

The string to write as the backing file. This string is not validated, so care should be taken when specifying the string or the image chain may not be able to be reopened again.

Returns: Nothing on success

If "device" does not exist or cannot be determined, DeviceNotFound

Since: 2.1

block-commit [Command]

Live commit of data from overlay image nodes into backing nodes - i.e., writes data between 'top' and 'base' into 'base'.

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

base: string (optional)

The file name of the backing image to write data into. If not specified, this is the deepest backing image.

top: string (optional)

The file name of the backing image within the image chain, which contains the topmost data to be committed down. If not specified, this is the active layer.

backing-file: string (optional)

The backing file string to write into the overlay image of 'top'. If 'top' is the active layer, specifying a backing file string is an error. This filename is not validated.

If a pathname string is such that it cannot be resolved by QEMU, that means that subsequent QMP or HMP commands must use node-names for the image in question, as filename lookup methods will fail.

If not specified, QEMU will automatically determine the backing file string to use, or error out if there is no obvious choice. Care should be taken when specifying the string, to specify a valid filename or protocol. (Since 2.1)

If top == base, that is an error. If top == active, the job will not be completed by itself, user needs to complete the job with the block-job-complete command after getting the ready event. (Since 2.0)

If the base image is smaller than top, then the base image will be resized to be the same size as top. If top is smaller than the base image, the base will not be truncated. If you want the base image size to match the size of the smaller top, you can safely truncate it yourself once the commit operation successfully completes.

speed: int (optional)

the maximum speed, in bytes per second

filter-node-name: string (optional)

the node name that should be assigned to the filter driver that the commit job inserts into the graph above top. If this option is not given, a node name is autogenerated. (Since: 2.9)

Returns: Nothing on success If commit or stream is already active on this device, DeviceInUse If device does not exist, DeviceNotFound If image commit is not supported by this device, NotSupported If base or top is invalid, a generic error is returned If speed is invalid, InvalidParameter

Since: 1.3

Example:

```
-> { "execute": "block-commit",
      "arguments": { "device": "virtio0",
                    "top": "/tmp/snap1.qcow2" } }
<- { "return": {} }
```

drive-backup

[Command]

Start a point-in-time copy of a block device to a new destination. The status of ongoing drive-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 DriveBackup

Returns: nothing on success If device is not a valid block device, GenericError

Since: 1.6

Example:

```
-> { "execute": "drive-backup",
      "arguments": { "device": "drive0",
                    "sync": "full",
                    "target": "backup.img" } }
<- { "return": {} }
```

`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`

Returns: nothing on success If device is not a valid block device, `DeviceNotFound`

Since: 2.3

Example:

```
-> { "execute": "blockdev-backup",
      "arguments": { "device": "src-id",
                    "sync": "full",
                    "target": "tgt-id" } }
<- { "return": {} }
```

`query-named-block-nodes` [Command]

Get the named block driver list

Returns: the list of `BlockDeviceInfo`

Since: 2.0

Example:

```
-> { "execute": "query-named-block-nodes" }
<- { "return": [ { "ro":false,
                  "drv":"qcow2",
                  "encrypted":false,
                  "file":"disks/test.qcow2",
                  "node-name": "my-node",
                  "backing_file_depth":1,
                  "bps":1000000,
                  "bps_rd":0,
                  "bps_wr":0,
                  "iops":1000000,
                  "iops_rd":0,
                  "iops_wr":0,
                  "bps_max": 8000000,
                  "bps_rd_max": 0,
                  "bps_wr_max": 0,
                  "iops_max": 0,
                  "iops_rd_max": 0,
```

```

    "iops_wr_max": 0,
    "iops_size": 0,
    "write_threshold": 0,
    "image":{
      "filename":"disks/test.qcow2",
      "format":"qcow2",
      "virtual-size":2048000,
      "backing_file":"base.qcow2",
      "full-backing-filename":"disks/base.qcow2",
      "backing-filename-format":"qcow2",
      "snapshots":[
        {
          "id": "1",
          "name": "snapshot1",
          "vm-state-size": 0,
          "date-sec": 10000200,
          "date-nsec": 12,
          "vm-clock-sec": 206,
          "vm-clock-nsec": 30
        }
      ],
      "backing-image":{
        "filename":"disks/base.qcow2",
        "format":"qcow2",
        "virtual-size":2048000
      }
    }
  } ] }

```

drive-mirror [Command]

Start mirroring a block device's writes to a new destination. `target` specifies the target of the new image. If the file exists, or if it is a device, it will be used as the new destination for writes. If it does not exist, a new file will be created. `format` specifies the format of the mirror image, default is to probe if mode='existing', else the format of the source.

Arguments: the members of `DriveMirror`

Returns: nothing on success If `device` is not a valid block device, `GenericError`

Since: 1.3

Example:

```

-> { "execute": "drive-mirror",
    "arguments": { "device": "ide-hd0",
                  "target": "/some/place/my-image",
                  "sync": "full",
                  "format": "qcow2" } }

<- { "return": {} }

```

DriveMirror [Object]

A set of parameters describing drive mirror setup.

Members:

- 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 whose writes should be mirrored.
- target:** `string`
the target of the new image. If the file exists, or if it is a device, the existing file/device will be used as the new destination. If it does not exist, a new file will be created.
- format:** `string` (optional)
the format of the new destination, default is to probe if `mode` is 'existing', else the format of the source
- node-name:** `string` (optional)
the new block driver state node name in the graph (Since 2.1)
- replaces:** `string` (optional)
with `sync=full` graph node name to be replaced by the new image when a whole image copy is done. This can be used to repair broken Quorum files. (Since 2.1)
- mode:** `NewImageMode` (optional)
whether and how QEMU should create a new image, default is 'absolute-paths'.
- speed:** `int` (optional)
the maximum speed, in bytes per second
- 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, or only new I/O).
- granularity:** `int` (optional)
granularity of the dirty bitmap, default is 64K if the image format doesn't have clusters, 4K if the clusters are smaller than that, else the cluster size. Must be a power of 2 between 512 and 64M (since 1.4).
- buf-size:** `int` (optional)
maximum amount of data in flight from source to target (since 1.4).
- on-source-error:** `BlockdevOnError` (optional)
the action to take on an error on the source, default 'report'. 'stop' and 'enospc' can only be used if the block device supports io-status (see `BlockInfo`).
- on-target-error:** `BlockdevOnError` (optional)
the action to take on an error on the target, default 'report' (no limitations, since this applies to a different block device than `device`).

unmap: `boolean` (optional)

Whether to try to unmap target sectors where source has only zero. If true, and target unallocated sectors will read as zero, target image sectors will be unmapped; otherwise, zeroes will be written. Both will result in identical contents. Default is true. (Since 2.4)

Since: 1.3

BlockDirtyBitmap [Object]

Members:

node: `string`

name of device/node which the bitmap is tracking

name: `string`

name of the dirty bitmap

Since: 2.4

BlockDirtyBitmapAdd [Object]

Members:

node: `string`

name of device/node which the bitmap is tracking

name: `string`

name of the dirty bitmap

granularity: `int` (optional)

the bitmap granularity, default is 64k for block-dirty-bitmap-add

persistent: `boolean` (optional)

the bitmap is persistent, i.e. it will be saved to the corresponding block device image file on its close. For now only Qcow2 disks support persistent bitmaps. Default is false for block-dirty-bitmap-add. (Since: 2.10)

autoload: `boolean` (optional)

the bitmap will be automatically loaded when the image it is stored in is opened. This flag may only be specified for persistent bitmaps. Default is false for block-dirty-bitmap-add. (Since: 2.10)

Since: 2.4

block-dirty-bitmap-add [Command]

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

Returns: nothing on success If `node` is not a valid block device or node, `DeviceNotFound` If `name` is already taken, `GenericError` with an explanation

Since: 2.4

Example:

```
-> { "execute": "block-dirty-bitmap-add",
      "arguments": { "node": "drive0", "name": "bitmap0" } }
<- { "return": {} }
```

block-dirty-bitmap-remove [Command]

Stop write tracking and remove the dirty bitmap that was created with `block-dirty-bitmap-add`. If the bitmap is persistent, remove it from its storage too.

Returns: nothing on success If `node` is not a valid block device or node, `DeviceNotFound` If `name` is not found, `GenericError` with an explanation if `name` is frozen by an operation, `GenericError`

Since: 2.4

Example:

```
-> { "execute": "block-dirty-bitmap-remove",
      "arguments": { "node": "drive0", "name": "bitmap0" } }
<- { "return": {} }
```

block-dirty-bitmap-clear [Command]

Clear (reset) a dirty bitmap on the device, so that an incremental backup from this point in time forward will only backup clusters modified after this clear operation.

Returns: nothing on success If `node` is not a valid block device, `DeviceNotFound` If `name` is not found, `GenericError` with an explanation

Since: 2.4

Example:

```
-> { "execute": "block-dirty-bitmap-clear",
      "arguments": { "node": "drive0", "name": "bitmap0" } }
<- { "return": {} }
```

BlockDirtyBitmapSha256 [Object]

SHA256 hash of dirty bitmap data

Members:

`sha256`: `string`
ASCII representation of SHA256 bitmap hash

Since: 2.10

x-debug-block-dirty-bitmap-sha256 [Command]

Get bitmap SHA256

Returns: `BlockDirtyBitmapSha256` on success If `node` is not a valid block device, `DeviceNotFound` If `name` is not found or if hashing has failed, `GenericError` with an explanation

Since: 2.10

blockdev-mirror [Command]

Start mirroring a block device's writes to a new destination.

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 whose writes should be mirrored.

target: `string`
 the id or node-name of the block device to mirror to. This mustn't be attached to guest.

replaces: `string` (optional)
 with `sync=full` graph node name to be replaced by the new image when a whole image copy is done. This can be used to repair broken Quorum files.

speed: `int` (optional)
 the maximum speed, in bytes per second

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, or only new I/O).

granularity: `int` (optional)
 granularity of the dirty bitmap, default is 64K if the image format doesn't have clusters, 4K if the clusters are smaller than that, else the cluster size. Must be a power of 2 between 512 and 64M

buf-size: `int` (optional)
 maximum amount of data in flight from source to target

on-source-error: `BlockdevOnError` (optional)
 the action to take on an error on the source, default 'report'. 'stop' and 'enospc' can only be used if the block device supports io-status (see `BlockInfo`).

on-target-error: `BlockdevOnError` (optional)
 the action to take on an error on the target, default 'report' (no limitations, since this applies to a different block device than `device`).

filter-node-name: `string` (optional)
 the node name that should be assigned to the filter driver that the mirror job inserts into the graph above `device`. If this option is not given, a node name is autogenerated. (Since: 2.9)

Returns: nothing on success.

Since: 2.6

Example:

```
-> { "execute": "blockdev-mirror",
      "arguments": { "device": "ide-hd0",
                    "target": "target0",
                    "sync": "full" } }
<- { "return": {} }
```

block_set_io_throttle [Command]

Change I/O throttle limits for a block drive.

Since QEMU 2.4, each device with I/O limits is member of a throttle group.

If two or more devices are members of the same group, the limits will apply to the combined I/O of the whole group in a round-robin fashion. Therefore, setting new I/O limits to a device will affect the whole group.

The name of the group can be specified using the 'group' parameter. If the parameter is unset, it is assumed to be the current group of that device. If it's not in any group yet, the name of the device will be used as the name for its group.

The 'group' parameter can also be used to move a device to a different group. In this case the limits specified in the parameters will be applied to the new group only.

I/O limits can be disabled by setting all of them to 0. In this case the device will be removed from its group and the rest of its members will not be affected. The 'group' parameter is ignored.

Arguments: the members of `BlockIOThrottle`

Returns: Nothing on success If device is not a valid block device, `DeviceNotFound`

Since: 1.1

Example:

```
-> { "execute": "block_set_io_throttle",
      "arguments": { "id": "ide0-1-0",
                    "bps": 1000000,
                    "bps_rd": 0,
                    "bps_wr": 0,
                    "iops": 0,
                    "iops_rd": 0,
                    "iops_wr": 0,
                    "bps_max": 8000000,
                    "bps_rd_max": 0,
                    "bps_wr_max": 0,
                    "iops_max": 0,
                    "iops_rd_max": 0,
                    "iops_wr_max": 0,
                    "bps_max_length": 60,
                    "iops_size": 0 } }

<- { "return": {} }
```

BlockIOThrottle [Object]

A set of parameters describing block throttling.

Members:

device: string (optional)

Block device name (deprecated, use `id` instead)

id: string (optional)

The name or QOM path of the guest device (since: 2.8)

bps: int total throughput limit in bytes per second

bps_rd: int
read throughput limit in bytes per second

bps_wr: int
write throughput limit in bytes per second

iops: int total I/O operations per second

iops_rd: int
read I/O operations per second

iops_wr: int
write I/O operations per second

bps_max: int (optional)
total throughput limit during bursts, in bytes (Since 1.7)

bps_rd_max: int (optional)
read throughput limit during bursts, in bytes (Since 1.7)

bps_wr_max: int (optional)
write throughput limit during bursts, in bytes (Since 1.7)

iops_max: int (optional)
total I/O operations per second during bursts, in bytes (Since 1.7)

iops_rd_max: int (optional)
read I/O operations per second during bursts, in bytes (Since 1.7)

iops_wr_max: int (optional)
write I/O operations per second during bursts, in bytes (Since 1.7)

bps_max_length: int (optional)
maximum length of the **bps_max** burst period, in seconds. It must only be set if **bps_max** is set as well. Defaults to 1. (Since 2.6)

bps_rd_max_length: int (optional)
maximum length of the **bps_rd_max** burst period, in seconds. It must only be set if **bps_rd_max** is set as well. Defaults to 1. (Since 2.6)

bps_wr_max_length: int (optional)
maximum length of the **bps_wr_max** burst period, in seconds. It must only be set if **bps_wr_max** is set as well. Defaults to 1. (Since 2.6)

iops_max_length: int (optional)
maximum length of the **iops** burst period, in seconds. It must only be set if **iops_max** is set as well. Defaults to 1. (Since 2.6)

iops_rd_max_length: int (optional)
maximum length of the **iops_rd_max** burst period, in seconds. It must only be set if **iops_rd_max** is set as well. Defaults to 1. (Since 2.6)

iops_wr_max_length: int (optional)
maximum length of the **iops_wr_max** burst period, in seconds. It must only be set if **iops_wr_max** is set as well. Defaults to 1. (Since 2.6)

`iops_size`: `int` (optional)
 an I/O size in bytes (Since 1.7)

`group`: `string` (optional)
 throttle group name (Since 2.4)

Since: 1.1

ThrottleLimits [Object]

Limit parameters for throttling. Since some limit combinations are illegal, limits should always be set in one transaction. All fields are optional. When setting limits, if a field is missing the current value is not changed.

Members:

`iops-total`: `int` (optional)
 limit total I/O operations per second

`iops-total-max`: `int` (optional)
 I/O operations burst

`iops-total-max-length`: `int` (optional)
 length of the `iops-total-max` burst period, in seconds It must only be set if `iops-total-max` is set as well.

`iops-read`: `int` (optional)
 limit read operations per second

`iops-read-max`: `int` (optional)
 I/O operations read burst

`iops-read-max-length`: `int` (optional)
 length of the `iops-read-max` burst period, in seconds It must only be set if `iops-read-max` is set as well.

`iops-write`: `int` (optional)
 limit write operations per second

`iops-write-max`: `int` (optional)
 I/O operations write burst

`iops-write-max-length`: `int` (optional)
 length of the `iops-write-max` burst period, in seconds It must only be set if `iops-write-max` is set as well.

`bps-total`: `int` (optional)
 limit total bytes per second

`bps-total-max`: `int` (optional)
 total bytes burst

`bps-total-max-length`: `int` (optional)
 length of the `bps-total-max` burst period, in seconds. It must only be set if `bps-total-max` is set as well.

`bps-read`: `int` (optional)
 limit read bytes per second

bps-read-max: `int` (optional)
total bytes read burst

bps-read-max-length: `int` (optional)
length of the `bps-read-max` burst period, in seconds It must only be set if `bps-read-max` is set as well.

bps-write: `int` (optional)
limit write bytes per second

bps-write-max: `int` (optional)
total bytes write burst

bps-write-max-length: `int` (optional)
length of the `bps-write-max` burst period, in seconds It must only be set if `bps-write-max` is set as well.

iops-size: `int` (optional)
when limiting by iops max size of an I/O in bytes

Since: 2.11

block-stream [Command]

Copy data from a backing file into a block device.

The block streaming operation is performed in the background until the entire backing file has been copied. This command returns immediately once streaming has started. The status of ongoing block streaming operations can be checked with `query-block-jobs`. The operation can be stopped before it has completed using the `block-job-cancel` command.

The node that receives the data is called the top image, can be located in any part of the chain (but always above the base image; see below) and can be specified using its device or node name. Earlier `qemu` versions only allowed 'device' to name the top level node; presence of the 'base-node' parameter during introspection can be used as a witness of the enhanced semantics of 'device'.

If a base file is specified then sectors are not copied from that base file and its backing chain. When streaming completes the image file will have the base file as its backing file. This can be used to stream a subset of the backing file chain instead of flattening the entire image.

On successful completion the image file is updated to drop the backing file and the `BLOCK_JOB_COMPLETED` event is emitted.

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 or node name of the top image

base: `string` (optional)
the common backing file name. It cannot be set if `base-node` is also set.

base-node: `string` (optional)
 the node name of the backing file. It cannot be set if `base` is also set.
 (Since 2.8)

backing-file: `string` (optional)
 The backing file string to write into the top image. This filename is not validated.

If a pathname string is such that it cannot be resolved by QEMU, that means that subsequent QMP or HMP commands must use node-names for the image in question, as filename lookup methods will fail.

If not specified, QEMU will automatically determine the backing file string to use, or error out if there is no obvious choice. Care should be taken when specifying the string, to specify a valid filename or protocol.
 (Since 2.1)

speed: `int` (optional)
 the maximum speed, in bytes per second

on-error: `BlockdevOnError` (optional)
 the action to take on an error (default report). 'stop' and 'enospc' can only be used if the block device supports io-status (see `BlockInfo`). Since 1.3.

Returns: Nothing on success. If device does not exist, `DeviceNotFound`.

Since: 1.1

Example:

```
-> { "execute": "block-stream",
      "arguments": { "device": "virtio0",
                    "base": "/tmp/master.qcow2" } }
<- { "return": {} }
```

block-job-set-speed [Command]

Set maximum speed for a background block operation.

This command can only be issued when there is an active block job.

Throttling can be disabled by setting the speed to 0.

Arguments:

device: `string`
 The job identifier. This used to be a device name (hence the name of the parameter), but since QEMU 2.7 it can have other values.

speed: `int`
 the maximum speed, in bytes per second, or 0 for unlimited. Defaults to 0.

Returns: Nothing on success If no background operation is active on this device, `DeviceNotActive`

Since: 1.1

block-job-cancel [Command]

Stop an active background block operation.

This command returns immediately after marking the active background block operation for cancellation. It is an error to call this command if no operation is in progress.

The operation will cancel as soon as possible and then emit the `BLOCK_JOB_CANCELLED` event. Before that happens the job is still visible when enumerated using `query-block-jobs`.

For streaming, the image file retains its backing file unless the streaming operation happens to complete just as it is being cancelled. A new streaming operation can be started at a later time to finish copying all data from the backing file.

Arguments:

device: `string`

The job identifier. This used to be a device name (hence the name of the parameter), but since QEMU 2.7 it can have other values.

force: `boolean` (optional)

whether to allow cancellation of a paused job (default false). Since 1.3.

Returns: Nothing on success If no background operation is active on this device, `DeviceNotActive`

Since: 1.1

block-job-pause [Command]

Pause an active background block operation.

This command returns immediately after marking the active background block operation for pausing. It is an error to call this command if no operation is in progress. Pausing an already paused job has no cumulative effect; a single `block-job-resume` command will resume the job.

The operation will pause as soon as possible. No event is emitted when the operation is actually paused. Cancelling a paused job automatically resumes it.

Arguments:

device: `string`

The job identifier. This used to be a device name (hence the name of the parameter), but since QEMU 2.7 it can have other values.

Returns: Nothing on success If no background operation is active on this device, `DeviceNotActive`

Since: 1.3

block-job-resume [Command]

Resume an active background block operation.

This command returns immediately after resuming a paused background block operation. It is an error to call this command if no operation is in progress. Resuming an already running job is not an error.

This command also clears the error status of the job.

Arguments:**device:** string

The job identifier. This used to be a device name (hence the name of the parameter), but since QEMU 2.7 it can have other values.

Returns: Nothing on success If no background operation is active on this device, DeviceNotActive

Since: 1.3

block-job-complete [Command]

Manually trigger completion of an active background block operation. This is supported for drive mirroring, where it also switches the device to write to the target path only. The ability to complete is signaled with a BLOCK_JOB_READY event.

This command completes an active background block operation synchronously. The ordering of this command's return with the BLOCK_JOB_COMPLETED event is not defined. Note that if an I/O error occurs during the processing of this command: 1) the command itself will fail; 2) the error will be processed according to the error/werror arguments that were specified when starting the operation.

A cancelled or paused job cannot be completed.

Arguments:**device:** string

The job identifier. This used to be a device name (hence the name of the parameter), but since QEMU 2.7 it can have other values.

Returns: Nothing on success If no background operation is active on this device, DeviceNotActive

Since: 1.3

BlockdevDiscardOptions [Enum]

Determines how to handle discard requests.

Values:

ignore Ignore the request

unmap Forward as an unmap request

Since: 2.9

BlockdevDetectZeroesOptions [Enum]

Describes the operation mode for the automatic conversion of plain zero writes by the OS to driver specific optimized zero write commands.

Values:

off Disabled (default)

on Enabled

unmap Enabled and even try to unmap blocks if possible. This requires also that BlockdevDiscardOptions is set to unmap for this device.

Since: 2.1

BlockdevAioOptions [Enum]

Selects the AIO backend to handle I/O requests

Values:

threads Use qemu's thread pool

native Use native AIO backend (only Linux and Windows)

Since: 2.9

BlockdevCacheOptions [Object]

Includes cache-related options for block devices

Members:

direct: **boolean** (optional)

enables use of O_DIRECT (bypass the host page cache; default: false)

no-flush: **boolean** (optional)

ignore any flush requests for the device (default: false)

Since: 2.9

BlockdevDriver [Enum]

Drivers that are supported in block device operations.

Values:

vxhs Since 2.10

throttle Since 2.11

blkdebug Not documented

blkverify
Not documented

bochs Not documented

cloop Not documented

dmg Not documented

file Not documented

ftp Not documented

ftps Not documented

gluster Not documented

host_cdrom
Not documented

host_device
Not documented

http Not documented

https Not documented

`iscsi` Not documented
`luks` Not documented
`nbd` Not documented
`nfs` Not documented
`null-aio` Not documented
`null-co` Not documented
`parallels`
 Not documented
`qcow` Not documented
`qcow2` Not documented
`qed` Not documented
`quorum` Not documented
`raw` Not documented
`rbd` Not documented
`replication`
 Not documented
`sheepdog` Not documented
`ssh` Not documented
`vdi` Not documented
`vhdx` Not documented
`vmdk` Not documented
`vpc` Not documented
`vfat` Not documented

Since: 2.9

BlockdevOptionsFile

[Object]

Driver specific block device options for the file backend.

Members:

`filename`: `string`
 path to the image file

`aio`: `BlockdevAioOptions` (optional)
 AIO backend (default: threads) (since: 2.8)

`locking`: `OnOffAuto` (optional)
 whether to enable file locking. If set to 'auto', only enable when Open File Descriptor (OFD) locking API is available (default: auto, since 2.10)

Since: 2.9

BlockdevOptionsNull [Object]

Driver specific block device options for the null backend.

Members:

size: `int` (optional)
size of the device in bytes.

latency-ns: `int` (optional)
emulated latency (in nanoseconds) in processing requests. Default to zero which completes requests immediately. (Since 2.4)

Since: 2.9

BlockdevOptionsVVFAT [Object]

Driver specific block device options for the vvfat protocol.

Members:

dir: `string`
directory to be exported as FAT image

fat-type: `int` (optional)
FAT type: 12, 16 or 32

floppy: `boolean` (optional)
whether to export a floppy image (true) or partitioned hard disk (false; default)

label: `string` (optional)
set the volume label, limited to 11 bytes. FAT16 and FAT32 traditionally have some restrictions on labels, which are ignored by most operating systems. Defaults to "QEMU VVFAT". (since 2.4)

rw: `boolean` (optional)
whether to allow write operations (default: false)

Since: 2.9

BlockdevOptionsGenericFormat [Object]

Driver specific block device options for image format that have no option besides their data source.

Members:

file: `BlockdevRef`
reference to or definition of the data source block device

Since: 2.9

BlockdevOptionsLUKS [Object]

Driver specific block device options for LUKS.

Members:

key-secret: `string` (optional)
the ID of a `QCryptoSecret` object providing the decryption key (since 2.6). Mandatory except when doing a metadata-only probe of the image.

The members of `BlockdevOptionsGenericFormat`

Since: 2.9

`BlockdevOptionsGenericCOWFormat` [Object]

Driver specific block device options for image format that have no option besides their data source and an optional backing file.

Members:

`backing`: `BlockdevRefOrNull` (optional)
reference to or definition of the backing file block device, null disables the backing file entirely. Defaults to the backing file stored the image file.

The members of `BlockdevOptionsGenericFormat`

Since: 2.9

`Qcow2OverlapCheckMode` [Enum]

General overlap check modes.

Values:

`none` Do not perform any checks

`constant` Perform only checks which can be done in constant time and without reading anything from disk

`cached` Perform only checks which can be done without reading anything from disk

`all` Perform all available overlap checks

Since: 2.9

`Qcow2OverlapCheckFlags` [Object]

Structure of flags for each metadata structure. Setting a field to 'true' makes qemu guard that structure against unintended overwriting. The default value is chosen according to the template given.

Members:

`template`: `Qcow2OverlapCheckMode` (optional)
Specifies a template mode which can be adjusted using the other flags, defaults to 'cached'

`main-header`: `boolean` (optional)
Not documented

`active-l1`: `boolean` (optional)
Not documented

`active-l2`: `boolean` (optional)
Not documented

`refcount-table`: `boolean` (optional)
Not documented

`refcount-block`: `boolean` (optional)
Not documented

`snapshot-table`: `boolean` (optional)
Not documented

`inactive-l1`: `boolean` (optional)
Not documented

`inactive-l2`: `boolean` (optional)
Not documented

Since: 2.9

`Qcow2OverlapChecks` [Alternate]

Specifies which metadata structures should be guarded against unintended overwriting.

Members:

`flags`: `Qcow2OverlapCheckFlags`
set of flags for separate specification of each metadata structure type

`mode`: `Qcow2OverlapCheckMode`
named mode which chooses a specific set of flags

Since: 2.9

`BlockdevQcowEncryptionFormat` [Enum]

Values:

`aes` AES-CBC with plain64 initialization vectors

Since: 2.10

`BlockdevQcowEncryption` [Object]

Members:

`format`: `BlockdevQcowEncryptionFormat`
Not documented

The members of `QCryptoBlockOptionsQCow` when `format` is "aes"

Since: 2.10

`BlockdevOptionsQcow` [Object]

Driver specific block device options for qcow.

Members:

`encrypt`: `BlockdevQcowEncryption` (optional)
Image decryption options. Mandatory for encrypted images, except when doing a metadata-only probe of the image.

The members of `BlockdevOptionsGenericCOWFormat`

Since: 2.10

`BlockdevQcow2EncryptionFormat` [Enum]

Values:

`aes` AES-CBC with plain64 initialization venvtors

`luks` Not documented

Since: 2.10

`BlockdevQcow2Encryption` [Object]

Members:

`format: BlockdevQcow2EncryptionFormat`
Not documented

The members of `QCryptoBlockOptionsQcow` when `format` is "aes"

The members of `QCryptoBlockOptionsLUKS` when `format` is "luks"

Since: 2.10

`BlockdevOptionsQcow2` [Object]

Driver specific block device options for qcow2.

Members:

`lazy-refcounts: boolean` (optional)
whether to enable the lazy refcounts feature (default is taken from the image file)

`pass-discard-request: boolean` (optional)
whether discard requests to the qcow2 device should be forwarded to the data source

`pass-discard-snapshot: boolean` (optional)
whether discard requests for the data source should be issued when a snapshot operation (e.g. deleting a snapshot) frees clusters in the qcow2 file

`pass-discard-other: boolean` (optional)
whether discard requests for the data source should be issued on other occasions where a cluster gets freed

`overlap-check: Qcow2OverlapChecks` (optional)
which overlap checks to perform for writes to the image, defaults to 'cached' (since 2.2)

`cache-size: int` (optional)
the maximum total size of the L2 table and refcount block caches in bytes (since 2.2)

`l2-cache-size: int` (optional)
the maximum size of the L2 table cache in bytes (since 2.2)

`refcount-cache-size: int` (optional)
the maximum size of the refcount block cache in bytes (since 2.2)

`cache-clean-interval: int` (optional)
clean unused entries in the L2 and refcount caches. The interval is in seconds. The default value is 0 and it disables this feature (since 2.5)

encrypt: `BlockdevQcow2Encryption` (optional)
Image decryption options. Mandatory for encrypted images, except when doing a metadata-only probe of the image. (since 2.10)

The members of `BlockdevOptionsGenericCOWFormat`

Since: 2.9

`BlockdevOptionsSsh` [Object]

Members:

server: `InetSocketAddress`
host address

path: `string`
path to the image on the host

user: `string` (optional)
user as which to connect, defaults to current local user name

TODO: Expose the `host_key_check` option in QMP

Since: 2.9

`BlkdebugEvent` [Enum]

Trigger events supported by blkdebug.

Values:

`l1_update`
Not documented

`l1_grow_alloc_table`
Not documented

`l1_grow_write_table`
Not documented

`l1_grow_activate_table`
Not documented

`l2_load` Not documented

`l2_update`
Not documented

`l2_update_compressed`
Not documented

`l2_alloc_cow_read`
Not documented

`l2_alloc_write`
Not documented

`read_aio` Not documented

`read_backing_aio`
Not documented

`read_compressed`
Not documented

`write_aio`
Not documented

`write_compressed`
Not documented

`vmstate_load`
Not documented

`vmstate_save`
Not documented

`cow_read` Not documented

`cow_write`
Not documented

`reftable_load`
Not documented

`reftable_grow`
Not documented

`reftable_update`
Not documented

`refblock_load`
Not documented

`refblock_update`
Not documented

`refblock_update_part`
Not documented

`refblock_alloc`
Not documented

`refblock_alloc_hookup`
Not documented

`refblock_alloc_write`
Not documented

`refblock_alloc_write_blocks`
Not documented

`refblock_alloc_write_table`
Not documented

`refblock_alloc_switch_table`
Not documented

`cluster_alloc`
Not documented

`cluster_alloc_bytes`
Not documented

`cluster_free`
Not documented

`flush_to_os`
Not documented

`flush_to_disk`
Not documented

`pwritev_rmw_head`
Not documented

`pwritev_rmw_after_head`
Not documented

`pwritev_rmw_tail`
Not documented

`pwritev_rmw_after_tail`
Not documented

`pwritev` Not documented

`pwritev_zero`
Not documented

`pwritev_done`
Not documented

`empty_image_prepare`
Not documented

Since: 2.9

BlkdebugInjectErrorOptions

[Object]

Describes a single error injection for blkdebug.

Members:

`event`: `BlkdebugEvent`
trigger event

`state`: `int` (optional)
the state identifier blkdebug needs to be in to actually trigger the event;
defaults to "any"

`errno`: `int` (optional)
error identifier (`errno`) to be returned; defaults to `EIO`

`sector`: `int` (optional)
specifies the sector index which has to be affected in order to actually
trigger the event; defaults to "any sector"

`once`: `boolean` (optional)
disables further events after this one has been triggered; defaults to `false`

`immediately`: `boolean` (optional)
fail immediately; defaults to false

Since: 2.9

`BlkdebugSetStateOptions` [Object]

Describes a single state-change event for blkdebug.

Members:

`event`: `BlkdebugEvent`
trigger event

`state`: `int` (optional)
the current state identifier blkdebug needs to be in; defaults to "any"

`new_state`: `int`
the state identifier blkdebug is supposed to assume if this event is triggered

Since: 2.9

`BlockdevOptionsBlkdebug` [Object]

Driver specific block device options for blkdebug.

Members:

`image`: `BlockdevRef`
underlying raw block device (or image file)

`config`: `string` (optional)
filename of the configuration file

`align`: `int` (optional)
required alignment for requests in bytes, must be positive power of 2, or 0 for default

`max-transfer`: `int` (optional)
maximum size for I/O transfers in bytes, must be positive multiple of `align` and of the underlying file's request alignment (but need not be a power of 2), or 0 for default (since 2.10)

`opt-write-zero`: `int` (optional)
preferred alignment for write zero requests in bytes, must be positive multiple of `align` and of the underlying file's request alignment (but need not be a power of 2), or 0 for default (since 2.10)

`max-write-zero`: `int` (optional)
maximum size for write zero requests in bytes, must be positive multiple of `align`, of `opt-write-zero`, and of the underlying file's request alignment (but need not be a power of 2), or 0 for default (since 2.10)

`opt-discard`: `int` (optional)
preferred alignment for discard requests in bytes, must be positive multiple of `align` and of the underlying file's request alignment (but need not be a power of 2), or 0 for default (since 2.10)

max-discard: `int` (optional)
 maximum size for discard requests in bytes, must be positive multiple of `align`, of `opt-discard`, and of the underlying file's request alignment (but need not be a power of 2), or 0 for default (since 2.10)

inject-error: array of `BlkdebugInjectErrorOptions` (optional)
 array of error injection descriptions

set-state: array of `BlkdebugSetStateOptions` (optional)
 array of state-change descriptions

Since: 2.9

BlockdevOptionsBlkverify [Object]

Driver specific block device options for blkverify.

Members:

test: `BlockdevRef`
 block device to be tested

raw: `BlockdevRef`
 raw image used for verification

Since: 2.9

QuorumReadPattern [Enum]

An enumeration of quorum read patterns.

Values:

quorum read all the children and do a quorum vote on reads

fifo read only from the first child that has not failed

Since: 2.9

BlockdevOptionsQuorum [Object]

Driver specific block device options for Quorum

Members:

blkverify: `boolean` (optional)
 true if the driver must print content mismatch set to false by default

children: array of `BlockdevRef`
 the children block devices to use

vote-threshold: `int`
 the vote limit under which a read will fail

rewrite-corrupted: `boolean` (optional)
 rewrite corrupted data when quorum is reached (Since 2.1)

read-pattern: `QuorumReadPattern` (optional)
 choose read pattern and set to quorum by default (Since 2.2)

Since: 2.9

BlockdevOptionsGluster [Object]

Driver specific block device options for Gluster

Members:

volume: `string`
name of gluster volume where VM image resides

path: `string`
absolute path to image file in gluster volume

server: `array of SocketAddress`
gluster servers description

debug: `int` (optional)
libgfapi log level (default '4' which is Error) (Since 2.8)

logfile: `string` (optional)
libgfapi log file (default /dev/stderr) (Since 2.8)

Since: 2.9

IscsiTransport [Enum]

An enumeration of libiscsi transport types

Values:

`tcp` Not documented

`iser` Not documented

Since: 2.9

IscsiHeaderDigest [Enum]

An enumeration of header digests supported by libiscsi

Values:

`crc32c` Not documented

`none` Not documented

`crc32c-none`
Not documented

`none-crc32c`
Not documented

Since: 2.9

BlockdevOptionsIscsi [Object]**Members:**

transport: `IscsiTransport`
The iscsi transport type

portal: `string`
The address of the iscsi portal

target: string
The target iqn name

lun: int (optional)
LUN to connect to. Defaults to 0.

user: string (optional)
User name to log in with. If omitted, no CHAP authentication is performed.

password-secret: string (optional)
The ID of a QCryptoSecret object providing the password for the login. This option is required if **user** is specified.

initiator-name: string (optional)
The iqn name we want to identify to the target as. If this option is not specified, an initiator name is generated automatically.

header-digest: IscsiHeaderDigest (optional)
The desired header digest. Defaults to none-crc32c.

timeout: int (optional)
Timeout in seconds after which a request will timeout. 0 means no timeout and is the default.

Driver specific block device options for iscsi

Since: 2.9

BlockdevOptionsRbd [Object]

Members:

pool: string
Ceph pool name.

image: string
Image name in the Ceph pool.

conf: string (optional)
path to Ceph configuration file. Values in the configuration file will be overridden by options specified via QAPI.

snapshot: string (optional)
Ceph snapshot name.

user: string (optional)
Ceph id name.

server: array of InetSocketAddressBase (optional)
Monitor host address and port. This maps to the "mon_host" Ceph option.

Since: 2.9

BlockdevOptionsSheepdog [Object]

Driver specific block device options for sheepdog

Members:

vdi: `string`
Virtual disk image name

server: `SocketAddress`
The Sheepdog server to connect to

snap-id: `int` (optional)
Snapshot ID

tag: `string` (optional)
Snapshot tag name

Only one of `snap-id` and `tag` may be present.

Since: 2.9

ReplicationMode [Enum]

An enumeration of replication modes.

Values:

primary Primary mode, the vm's state will be sent to secondary QEMU.

secondary Secondary mode, receive the vm's state from primary QEMU.

Since: 2.9

BlockdevOptionsReplication [Object]

Driver specific block device options for replication

Members:

mode: `ReplicationMode`
the replication mode

top-id: `string` (optional)
In secondary mode, node name or device ID of the root node who owns the replication node chain. Must not be given in primary mode.

The members of `BlockdevOptionsGenericFormat`

Since: 2.9

NFSTransport [Enum]

An enumeration of NFS transport types

Values:

inet TCP transport

Since: 2.9

NFSServer [Object]

Captures the address of the socket

Members:

type: `NFSTransport`
transport type used for NFS (only TCP supported)

host: `string`
host address for NFS server

Since: 2.9

BlockdevOptionsNfs [Object]

Driver specific block device option for NFS

Members:

server: `NFSServer`
host address

path: `string`
path of the image on the host

user: `int` (optional)
UID value to use when talking to the server (defaults to 65534 on Windows and `getuid()` on unix)

group: `int` (optional)
GID value to use when talking to the server (defaults to 65534 on Windows and `getgid()` in unix)

tcp-syn-count: `int` (optional)
number of SYNs during the session establishment (defaults to libnfs default)

readahead-size: `int` (optional)
set the readahead size in bytes (defaults to libnfs default)

page-cache-size: `int` (optional)
set the pagecache size in bytes (defaults to libnfs default)

debug: `int` (optional)
set the NFS debug level (max 2) (defaults to libnfs default)

Since: 2.9

BlockdevOptionsCurlBase [Object]

Driver specific block device options shared by all protocols supported by the curl backend.

Members:

url: `string`
URL of the image file

readahead: `int` (optional)
Size of the read-ahead cache; must be a multiple of 512 (defaults to 256 kB)

timeout: `int` (optional)
 Timeout for connections, in seconds (defaults to 5)

username: `string` (optional)
 Username for authentication (defaults to none)

password-secret: `string` (optional)
 ID of a `QCryptoSecret` object providing a password for authentication (defaults to no password)

proxy-username: `string` (optional)
 Username for proxy authentication (defaults to none)

proxy-password-secret: `string` (optional)
 ID of a `QCryptoSecret` object providing a password for proxy authentication (defaults to no password)

Since: 2.9

BlockdevOptionsCurlHttp [Object]

Driver specific block device options for HTTP connections over the curl backend. URLs must start with "http://".

Members:

cookie: `string` (optional)
 List of cookies to set; format is "name1=content1; name2=content2;" as explained by `CURLOPT_COOKIE(3)`. Defaults to no cookies.

cookie-secret: `string` (optional)
 ID of a `QCryptoSecret` object providing the cookie data in a secure way. See `cookie` for the format. (since 2.10)

The members of `BlockdevOptionsCurlBase`

Since: 2.9

BlockdevOptionsCurlHttps [Object]

Driver specific block device options for HTTPS connections over the curl backend. URLs must start with "https://".

Members:

cookie: `string` (optional)
 List of cookies to set; format is "name1=content1; name2=content2;" as explained by `CURLOPT_COOKIE(3)`. Defaults to no cookies.

sslverify: `boolean` (optional)
 Whether to verify the SSL certificate's validity (defaults to true)

cookie-secret: `string` (optional)
 ID of a `QCryptoSecret` object providing the cookie data in a secure way. See `cookie` for the format. (since 2.10)

The members of `BlockdevOptionsCurlBase`

Since: 2.9

BlockdevOptionsCurlFtp [Object]

Driver specific block device options for FTP connections over the curl backend. URLs must start with "ftp://".

Members:

The members of `BlockdevOptionsCurlBase`

Since: 2.9

BlockdevOptionsCurlFtps [Object]

Driver specific block device options for FTPS connections over the curl backend. URLs must start with "ftps://".

Members:

`sslverify: boolean` (optional)

Whether to verify the SSL certificate's validity (defaults to true)

The members of `BlockdevOptionsCurlBase`

Since: 2.9

BlockdevOptionsNbd [Object]

Driver specific block device options for NBD.

Members:

`server: SocketAddress`

NBD server address

`export: string` (optional)

export name

`tls-creds: string` (optional)

TLS credentials ID

Since: 2.9

BlockdevOptionsRaw [Object]

Driver specific block device options for the raw driver.

Members:

`offset: int` (optional)

position where the block device starts

`size: int` (optional)

the assumed size of the device

The members of `BlockdevOptionsGenericFormat`

Since: 2.9

BlockdevOptionsVxHS [Object]

Driver specific block device options for VxHS

Members:

`vdisk-id: string`

UUID of VxHS volume

server: `InetSocketAddressBase`
vxhs server IP, port

tls-creds: `string` (optional)
TLS credentials ID

Since: 2.10

BlockdevOptionsThrottle [Object]

Driver specific block device options for the throttle driver

Members:

throttle-group: `string`
the name of the throttle-group object to use. It must already exist.

file: `BlockdevRef`
reference to or definition of the data source block device

Since: 2.11

BlockdevOptions [Object]

Options for creating a block device. Many options are available for all block devices, independent of the block driver:

Members:

driver: `BlockdevDriver`
block driver name

node-name: `string` (optional)
the node name of the new node (Since 2.0). This option is required on the top level of blockdev-add.

discard: `BlockdevDiscardOptions` (optional)
discard-related options (default: ignore)

cache: `BlockdevCacheOptions` (optional)
cache-related options

read-only: `boolean` (optional)
whether the block device should be read-only (default: false)

detect-zeroes: `BlockdevDetectZeroesOptions` (optional)
detect and optimize zero writes (Since 2.1) (default: off)

force-share: `boolean` (optional)
force share all permission on added nodes. Requires read-only=true. (Since 2.10)

The members of `BlockdevOptionsBlkdebug` when driver is "blkdebug"
 The members of `BlockdevOptionsBlkverify` when driver is "blkverify"
 The members of `BlockdevOptionsGenericFormat` when driver is "bochs"
 The members of `BlockdevOptionsGenericFormat` when driver is "cloop"
 The members of `BlockdevOptionsGenericFormat` when driver is "dmg"
 The members of `BlockdevOptionsFile` when driver is "file"
 The members of `BlockdevOptionsCurlFtp` when driver is "ftp"
 The members of `BlockdevOptionsCurlFtps` when driver is "ftps"
 The members of `BlockdevOptionsGluster` when driver is "gluster"
 The members of `BlockdevOptionsFile` when driver is "host_cdrom"
 The members of `BlockdevOptionsFile` when driver is "host_device"
 The members of `BlockdevOptionsCurlHttp` when driver is "http"
 The members of `BlockdevOptionsCurlHttps` when driver is "https"
 The members of `BlockdevOptionsIscsi` when driver is "iscsi"
 The members of `BlockdevOptionsLUKS` when driver is "luks"
 The members of `BlockdevOptionsNbd` when driver is "nbd"
 The members of `BlockdevOptionsNfs` when driver is "nfs"
 The members of `BlockdevOptionsNull` when driver is "null-aio"
 The members of `BlockdevOptionsNull` when driver is "null-co"
 The members of `BlockdevOptionsGenericFormat` when driver is "parallels"
 The members of `BlockdevOptionsQcow2` when driver is "qcow2"
 The members of `BlockdevOptionsQcow` when driver is "qcow"
 The members of `BlockdevOptionsGenericCOWFormat` when driver is "qed"
 The members of `BlockdevOptionsQuorum` when driver is "quorum"
 The members of `BlockdevOptionsRaw` when driver is "raw"
 The members of `BlockdevOptionsRbd` when driver is "rbd"
 The members of `BlockdevOptionsReplication` when driver is "replication"
 The members of `BlockdevOptionsSheepdog` when driver is "sheepdog"
 The members of `BlockdevOptionsSsh` when driver is "ssh"
 The members of `BlockdevOptionsThrottle` when driver is "throttle"
 The members of `BlockdevOptionsGenericFormat` when driver is "vdi"
 The members of `BlockdevOptionsGenericFormat` when driver is "vhdx"
 The members of `BlockdevOptionsGenericCOWFormat` when driver is "vmdk"
 The members of `BlockdevOptionsGenericFormat` when driver is "vpc"
 The members of `BlockdevOptionsVVFAT` when driver is "vvfat"
 The members of `BlockdevOptionsVxHS` when driver is "vxhs"
 Remaining options are determined by the block driver.

Since: 2.9

BlockdevRef

[Alternate]

Reference to a block device.

Members:

definition: `BlockdevOptions`
 defines a new block device inline

reference: `string`
 references the ID of an existing block device

Since: 2.9

BlockdevRefOrNull

[Alternate]

Reference to a block device.

Members:

definition: BlockdevOptions
defines a new block device inline

reference: string
references the ID of an existing block device. An empty string means that no block device should be referenced. Deprecated; use null instead.

null: null
No block device should be referenced (since 2.10)

Since: 2.9

blockdev-add

[Command]

Creates a new block device. If the `id` option is given at the top level, a BlockBackend will be created; otherwise, `node-name` is mandatory at the top level and no BlockBackend will be created.

Arguments: the members of BlockdevOptions

Since: 2.9

Example:

```
1.
-> { "execute": "blockdev-add",
    "arguments": {
      "driver": "qcow2",
      "node-name": "test1",
      "file": {
        "driver": "file",
        "filename": "test.qcow2"
      }
    }
  }
<- { "return": {} }
```

```
2.
-> { "execute": "blockdev-add",
    "arguments": {
      "driver": "qcow2",
      "node-name": "node0",
      "discard": "unmap",
      "cache": {
        "direct": true
      },
      "file": {
        "driver": "file",
```

```

        "filename": "/tmp/test.qcow2"
    },
    "backing": {
        "driver": "raw",
        "file": {
            "driver": "file",
            "filename": "/dev/fdset/4"
        }
    }
}
}
}

```

```
<- { "return": {} }
```

blockdev-del [Command]

Deletes a block device that has been added using `blockdev-add`. The command will fail if the node is attached to a device or is otherwise being used.

Arguments:

node-name: string
Name of the graph node to delete.

Since: 2.9

Example:

```

-> { "execute": "blockdev-add",
    "arguments": {
        "driver": "qcow2",
        "node-name": "node0",
        "file": {
            "driver": "file",
            "filename": "test.qcow2"
        }
    }
}
}
<- { "return": {} }

-> { "execute": "blockdev-del",
    "arguments": { "node-name": "node0" }
}
<- { "return": {} }

```

blockdev-open-tray [Command]

Opens a block device's tray. If there is a block driver state tree inserted as a medium, it will become inaccessible to the guest (but it will remain associated to the block device, so closing the tray will make it accessible again).

If the tray was already open before, this will be a no-op.

Once the tray opens, a `DEVICE_TRAY_MOVED` event is emitted. There are cases in which no such event will be generated, these include:

- if the guest has locked the tray, `force` is false and the guest does not respond to the eject request
- if the BlockBackend denoted by `device` does not have a guest device attached to it
- if the guest device does not have an actual tray

Arguments:

`device`: string (optional)
Block device name (deprecated, use `id` instead)

`id`: string (optional)
The name or QOM path of the guest device (since: 2.8)

`force`: boolean (optional)
if false (the default), an eject request will be sent to the guest if it has locked the tray (and the tray will not be opened immediately); if true, the tray will be opened regardless of whether it is locked

Since: 2.5**Example:**

```
-> { "execute": "blockdev-open-tray",
      "arguments": { "id": "ide0-1-0" } }

<- { "timestamp": { "seconds": 1418751016,
                   "microseconds": 716996 },
      "event": "DEVICE_TRAY_MOVED",
      "data": { "device": "ide1-cd0",
                "id": "ide0-1-0",
                "tray-open": true } }

<- { "return": {} }
```

`blockdev-close-tray` [Command]

Closes a block device's tray. If there is a block driver state tree associated with the block device (which is currently ejected), that tree will be loaded as the medium.

If the tray was already closed before, this will be a no-op.

Arguments:

`device`: string (optional)
Block device name (deprecated, use `id` instead)

`id`: string (optional)
The name or QOM path of the guest device (since: 2.8)

Since: 2.5**Example:**

```
-> { "execute": "blockdev-close-tray",
      "arguments": { "id": "ide0-1-0" } }
```

```

<- { "timestamp": { "seconds": 1418751345,
                   "microseconds": 272147 },
     "event": "DEVICE_TRAY_MOVED",
     "data": { "device": "ide1-cd0",
               "id": "ide0-1-0",
               "tray-open": false } }

<- { "return": {} }

```

`x-blockdev-remove-medium` [Command]

Removes a medium (a block driver state tree) from a block device. That block device's tray must currently be open (unless there is no attached guest device).

If the tray is open and there is no medium inserted, this will be a no-op.

Arguments:

`device`: string (optional)
Block device name (deprecated, use `id` instead)

`id`: string (optional)
The name or QOM path of the guest device (since: 2.8)

Note: This command is still a work in progress and is considered experimental. Stay away from it unless you want to help with its development.

Since: 2.5

Example:

```

-> { "execute": "x-blockdev-remove-medium",
     "arguments": { "id": "ide0-1-0" } }

<- { "error": { "class": "GenericError",
               "desc": "Tray of device 'ide0-1-0' is not open" } }

-> { "execute": "blockdev-open-tray",
     "arguments": { "id": "ide0-1-0" } }

<- { "timestamp": { "seconds": 1418751627,
                   "microseconds": 549958 },
     "event": "DEVICE_TRAY_MOVED",
     "data": { "device": "ide1-cd0",
               "id": "ide0-1-0",
               "tray-open": true } }

<- { "return": {} }

-> { "execute": "x-blockdev-remove-medium",
     "arguments": { "id": "ide0-1-0" } }

<- { "return": {} }

```

x-blockdev-insert-medium [Command]

Inserts a medium (a block driver state tree) into a block device. That block device's tray must currently be open (unless there is no attached guest device) and there must be no medium inserted already.

Arguments:

device: string (optional)
Block device name (deprecated, use `id` instead)

id: string (optional)
The name or QOM path of the guest device (since: 2.8)

node-name: string
name of a node in the block driver state graph

Note: This command is still a work in progress and is considered experimental. Stay away from it unless you want to help with its development.

Since: 2.5

Example:

```
-> { "execute": "blockdev-add",
      "arguments": {
        "node-name": "node0",
        "driver": "raw",
        "file": { "driver": "file",
                  "filename": "fedora.iso" } } }
<- { "return": {} }

-> { "execute": "x-blockdev-insert-medium",
      "arguments": { "id": "ide0-1-0",
                    "node-name": "node0" } }

<- { "return": {} }
```

BlockdevChangeReadOnlyMode [Enum]

Specifies the new read-only mode of a block device subject to the `blockdev-change-medium` command.

Values:

retain Retains the current read-only mode

read-only Makes the device read-only

read-write Makes the device writable

Since: 2.3

blockdev-change-medium [Command]

Changes the medium inserted into a block device by ejecting the current medium and loading a new image file which is inserted as the new medium (this command

combines `blockdev-open-tray`, `x-blockdev-remove-medium`, `x-blockdev-insert-medium` and `blockdev-close-tray`).

Arguments:

`device`: `string` (optional)
 Block device name (deprecated, use `id` instead)

`id`: `string` (optional)
 The name or QOM path of the guest device (since: 2.8)

`filename`: `string`
 filename of the new image to be loaded

`format`: `string` (optional)
 format to open the new image with (defaults to the probed format)

`read-only-mode`: `BlockdevChangeReadOnlyMode` (optional)
 change the read-only mode of the device; defaults to 'retain'

Since: 2.5

Examples:

1. Change a removable medium

```
-> { "execute": "blockdev-change-medium",
      "arguments": { "id": "ide0-1-0",
                    "filename": "/srv/images/Fedora-12-x86_64-DVD.iso",
                    "format": "raw" } }
<- { "return": {} }
```

2. Load a read-only medium into a writable drive

```
-> { "execute": "blockdev-change-medium",
      "arguments": { "id": "floppyA",
                    "filename": "/srv/images/ro.img",
                    "format": "raw",
                    "read-only-mode": "retain" } }

<- { "error":
      { "class": "GenericError",
        "desc": "Could not open '/srv/images/ro.img': Permission denied" } }

-> { "execute": "blockdev-change-medium",
      "arguments": { "id": "floppyA",
                    "filename": "/srv/images/ro.img",
                    "format": "raw",
                    "read-only-mode": "read-only" } }

<- { "return": {} }
```

BlockErrorAction [Enum]

An enumeration of action that has been taken when a DISK I/O occurs

Values:

ignore error has been ignored
report error has been reported to the device
stop error caused VM to be stopped

Since: 2.1

BLOCK_IMAGE_CORRUPTED [Event]

Emitted when a disk image is being marked corrupt. The image can be identified by its device or node name. The 'device' field is always present for compatibility reasons, but it can be empty ("") if the image does not have a device name associated.

Arguments:

device: string
device name. This is always present for compatibility reasons, but it can be empty ("") if the image does not have a device name associated.

node-name: string (optional)
node name (Since: 2.4)

msg: string
informative message for human consumption, such as the kind of corruption being detected. It should not be parsed by machine as it is not guaranteed to be stable

offset: int (optional)
if the corruption resulted from an image access, this is the host's access offset into the image

size: int (optional)
if the corruption resulted from an image access, this is the access size

fatal: boolean
if set, the image is marked corrupt and therefore unusable after this event and must be repaired (Since 2.2; before, every BLOCK_IMAGE_CORRUPTED event was fatal)

Note: If action is "stop", a STOP event will eventually follow the BLOCK_IO_ERROR event.

Example:

```
<- { "event": "BLOCK_IMAGE_CORRUPTED",
      "data": { "device": "ide0-hd0", "node-name": "node0",
                "msg": "Prevented active L1 table overwrite", "offset": 196608,
                "size": 65536 },
      "timestamp": { "seconds": 1378126126, "microseconds": 966463 } }
```

Since: 1.7

BLOCK_IO_ERROR [Event]

Emitted when a disk I/O error occurs

Arguments:

device: string

device name. This is always present for compatibility reasons, but it can be empty ("") if the image does not have a device name associated.

node-name: string

node name. Note that errors may be reported for the root node that is directly attached to a guest device rather than for the node where the error occurred. (Since: 2.8)

operation: IoOperationType

I/O operation

action: BlockErrorAction

action that has been taken

nospace: boolean (optional)

true if I/O error was caused due to a no-space condition. This key is only present if query-block's io-status is present, please see query-block documentation for more information (since: 2.2)

reason: string

human readable string describing the error cause. (This field is a debugging aid for humans, it should not be parsed by applications) (since: 2.2)

Note: If action is "stop", a STOP event will eventually follow the BLOCK_IO_ERROR event

Since: 0.13.0

Example:

```
<- { "event": "BLOCK_IO_ERROR",
      "data": { "device": "ide0-hd1",
                "node-name": "#block212",
                "operation": "write",
                "action": "stop" },
      "timestamp": { "seconds": 1265044230, "microseconds": 450486 } }
```

BLOCK_JOB_COMPLETED [Event]

Emitted when a block job has completed

Arguments:

type: BlockJobType

job type

device: string

The job identifier. Originally the device name but other values are allowed since QEMU 2.7

len: int maximum progress value

offset: int
current progress value. On success this is equal to len. On failure this is less than len

speed: int
rate limit, bytes per second

error: string (optional)
error message. Only present on failure. This field contains a human-readable error message. There are no semantics other than that streaming has failed and clients should not try to interpret the error string

Since: 1.1

Example:

```
<- { "event": "BLOCK_JOB_COMPLETED",
      "data": { "type": "stream", "device": "virtio-disk0",
                "len": 10737418240, "offset": 10737418240,
                "speed": 0 },
      "timestamp": { "seconds": 1267061043, "microseconds": 959568 } }
```

BLOCK_JOB_CANCELLED [Event]

Emitted when a block job has been cancelled

Arguments:

type: BlockJobType
job type

device: string
The job identifier. Originally the device name but other values are allowed since QEMU 2.7

len: int maximum progress value

offset: int
current progress value. On success this is equal to len. On failure this is less than len

speed: int
rate limit, bytes per second

Since: 1.1

Example:

```
<- { "event": "BLOCK_JOB_CANCELLED",
      "data": { "type": "stream", "device": "virtio-disk0",
                "len": 10737418240, "offset": 134217728,
                "speed": 0 },
      "timestamp": { "seconds": 1267061043, "microseconds": 959568 } }
```

BLOCK_JOB_ERROR [Event]

Emitted when a block job encounters an error

Arguments:**device:** string

The job identifier. Originally the device name but other values are allowed since QEMU 2.7

operation: IoOperationType

I/O operation

action: BlockErrorAction

action that has been taken

Since: 1.3**Example:**

```
<- { "event": "BLOCK_JOB_ERROR",
      "data": { "device": "ide0-hd1",
                "operation": "write",
                "action": "stop" },
      "timestamp": { "seconds": 1265044230, "microseconds": 450486 } }
```

BLOCK_JOB_READY

[Event]

Emitted when a block job is ready to complete

Arguments:**type:** BlockJobType

job type

device: string

The job identifier. Originally the device name but other values are allowed since QEMU 2.7

len: int maximum progress value**offset:** int

current progress value. On success this is equal to len. On failure this is less than len

speed: int

rate limit, bytes per second

Note: The "ready to complete" status is always reset by a BLOCK_JOB_ERROR event**Since:** 1.3**Example:**

```
<- { "event": "BLOCK_JOB_READY",
      "data": { "device": "drive0", "type": "mirror", "speed": 0,
                "len": 2097152, "offset": 2097152 }
      "timestamp": { "seconds": 1265044230, "microseconds": 450486 } }
```

PreallocMode

[Enum]

Preallocation mode of QEMU image file

Values:**off** no preallocation

`metadata` preallocate only for metadata

`falloc` like `full` preallocation but allocate disk space by `posix_fallocate()` rather than writing zeros.

`full` preallocate all data by writing zeros to device to ensure disk space is really available. `full` preallocation also sets up metadata correctly.

Since: 2.2

`BLOCK_WRITE_THRESHOLD` [Event]

Emitted when writes on block device reaches or exceeds the configured write threshold. For thin-provisioned devices, this means the device should be extended to avoid pausing for disk exhaustion. The event is one shot. Once triggered, it needs to be re-registered with another `block-set-write-threshold` command.

Arguments:

`node-name: string`
graph node name on which the threshold was exceeded.

`amount-exceeded: int`
amount of data which exceeded the threshold, in bytes.

`write-threshold: int`
last configured threshold, in bytes.

Since: 2.3

`block-set-write-threshold` [Command]

Change the write threshold for a block drive. An event will be delivered if a write to this block drive crosses the configured threshold. The threshold is an offset, thus must be non-negative. Default is no write threshold. Setting the threshold to zero disables it.

This is useful to transparently resize thin-provisioned drives without the guest OS noticing.

Arguments:

`node-name: string`
graph node name on which the threshold must be set.

`write-threshold: int`
configured threshold for the block device, bytes. Use 0 to disable the threshold.

Since: 2.3

Example:

```
-> { "execute": "block-set-write-threshold",
      "arguments": { "node-name": "mydev",
                    "write-threshold": 17179869184 } }
<- { "return": {} }
```

x-blockdev-change [Command]

Dynamically reconfigure the block driver state graph. It can be used to add, remove, insert or replace a graph node. Currently only the Quorum driver implements this feature to add or remove its child. This is useful to fix a broken quorum child.

If `node` is specified, it will be inserted under `parent`. `child` may not be specified in this case. If both `parent` and `child` are specified but `node` is not, `child` will be detached from `parent`.

Arguments:

`parent: string`

the id or name of the parent node.

`child: string (optional)`

the name of a child under the given parent node.

`node: string (optional)`

the name of the node that will be added.

Note: this command is experimental, and its API is not stable. It does not support all kinds of operations, all kinds of children, nor all block drivers.

Warning: The data in a new quorum child **MUST** be consistent with that of the rest of the array.

Since: 2.7

Example:

1. Add a new node to a quorum

```
-> { "execute": "blockdev-add",
      "arguments": {
        "driver": "raw",
        "node-name": "new_node",
        "file": { "driver": "file",
                  "filename": "test.raw" } } }
<- { "return": {} }
-> { "execute": "x-blockdev-change",
      "arguments": { "parent": "disk1",
                    "node": "new_node" } }
<- { "return": {} }

```

2. Delete a quorum's node

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

```

1.7.2 Additional block stuff (VM related)

BiosAtaTranslation [Enum]

Policy that BIOS should use to interpret cylinder/head/sector addresses. Note that Bochs BIOS and SeaBIOS will not actually translate logical CHS to physical; instead, they will use logical block addressing.

Values:

auto	If cylinder/heads/sizes are passed, choose between none and LBA depending on the size of the disk. If they are not passed, choose none if QEMU can guess that the disk had 16 or fewer heads, large if QEMU can guess that the disk had 131072 or fewer tracks across all heads (i.e. cylinders*heads<131072), otherwise LBA.
none	The physical disk geometry is equal to the logical geometry.
lba	Assume 63 sectors per track and one of 16, 32, 64, 128 or 255 heads (if fewer than 255 are enough to cover the whole disk with 1024 cylinders/head). The number of cylinders/head is then computed based on the number of sectors and heads.
large	The number of cylinders per head is scaled down to 1024 by correspondingly scaling up the number of heads.
rechs	Same as large , but first convert a 16-head geometry to 15-head, by proportionally scaling up the number of cylinders/head.

Since: 2.0**FloppyDriveType** [Enum]

Type of Floppy drive to be emulated by the Floppy Disk Controller.

Values:

144	1.44MB 3.5" drive
288	2.88MB 3.5" drive
120	1.2MB 5.25" drive
none	No drive connected
auto	Automatically determined by inserted media at boot

Since: 2.6**BlockdevSnapshotInternal** [Object]**Members:**

device: string	the device name or node-name of a root node to generate the snapshot from
name: string	the name of the internal snapshot to be created

Notes: In transaction, if **name** is empty, or any snapshot matching **name** exists, the operation will fail. Only some image formats support it, for example, qcow2, rbd, and sheepdog.

Since: 1.7

blockdev-snapshot-internal-sync [Command]

Synchronously take an internal snapshot of a block device, when the format of the image used supports it. If the name is an empty string, or a snapshot with name already exists, the operation will fail.

For the arguments, see the documentation of `BlockdevSnapshotInternal`.

Returns: nothing on success

If `device` is not a valid block device, `GenericError`

If any snapshot matching `name` exists, or `name` is empty, `GenericError`

If the format of the image used does not support it, `BlockFormatFeatureNotSupported`

Since: 1.7

Example:

```
-> { "execute": "blockdev-snapshot-internal-sync",
      "arguments": { "device": "ide-hd0",
                    "name": "snapshot0" }
    }
<- { "return": {} }
```

blockdev-snapshot-delete-internal-sync [Command]

Synchronously delete an internal snapshot of a block device, when the format of the image used support it. The snapshot is identified by name or id or both. One of the name or id is required. Return `SnapshotInfo` for the successfully deleted snapshot.

Arguments:

device: string

the device name or node-name of a root node to delete the snapshot from

id: string (optional)

optional the snapshot's ID to be deleted

name: string (optional)

optional the snapshot's name to be deleted

Returns: `SnapshotInfo` on success If `device` is not a valid block device, `GenericError` If snapshot not found, `GenericError` If the format of the image used does not support it, `BlockFormatFeatureNotSupported` If `id` and `name` are both not specified, `GenericError`

Since: 1.7

Example:

```
-> { "execute": "blockdev-snapshot-delete-internal-sync",
      "arguments": { "device": "ide-hd0",
                    "name": "snapshot0" }
    }
<- { "return": {
      "id": "1",
      "name": "snapshot0",
      "vm-state-size": 0,
      "date-sec": 1000012,
    } }
```

```

        "date-nsec": 10,
        "vm-clock-sec": 100,
        "vm-clock-nsec": 20
    }
}

```

eject [Command]

Ejects a device from a removable drive.

Arguments:

device: `string` (optional)

Block device name (deprecated, use `id` instead)

id: `string` (optional)

The name or QOM path of the guest device (since: 2.8)

force: `boolean` (optional)

If true, eject regardless of whether the drive is locked. If not specified, the default value is false.

Returns: Nothing on success

If `device` is not a valid block device, `DeviceNotFound`

Notes: Ejecting a device with no media results in success

Since: 0.14.0

Example:

```

-> { "execute": "eject", "arguments": { "id": "ide1-0-1" } }
<- { "return": {} }

```

nbds-server-start [Command]

Start an NBD server listening on the given host and port. Block devices can then be exported using `nbds-server-add`. The NBD server will present them as named exports; for example, another QEMU instance could refer to them as `"nbd:HOST:PORT:exportname=NAME"`.

Arguments:

addr: `SocketAddressLegacy`

Address on which to listen.

tls-creds: `string` (optional)

(optional) ID of the TLS credentials object. Since 2.6

Returns: error if the server is already running.

Since: 1.3.0

nbds-server-add [Command]

Export a block node to QEMU's embedded NBD server.

Arguments:

device: `string`

The device name or node name of the node to be exported

writable: `boolean` (optional)
Whether clients should be able to write to the device via the NBD connection (default `false`).

Returns: error if the device is already marked for export.

Since: 1.3.0

nbds-server-stop [Command]
Stop QEMU's embedded NBD server, and unregister all devices previously added via `nbds-server-add`.

Since: 1.3.0

DEVICE_TRAY_MOVED [Event]
Emitted whenever the tray of a removable device is moved by the guest or by HMP/QMP commands

Arguments:

device: `string`
Block device name. This is always present for compatibility reasons, but it can be empty (`""`) if the image does not have a device name associated.

id: `string`
The name or QOM path of the guest device (since 2.8)

tray-open: `boolean`
`true` if the tray has been opened or `false` if it has been closed

Since: 1.1

Example:

```
<- { "event": "DEVICE_TRAY_MOVED",
      "data": { "device": "ide1-cd0",
                "id": "/machine/unattached/device[22]",
                "tray-open": true
              },
      "timestamp": { "seconds": 1265044230, "microseconds": 450486 } }
```

QuorumOpType [Enum]
An enumeration of the quorum operation types

Values:

`read` read operation

`write` write operation

`flush` flush operation

Since: 2.6

QUORUM_FAILURE [Event]
Emitted by the Quorum block driver if it fails to establish a quorum

Arguments:

reference: `string`
device name if defined else node name

sector-num: int
number of the first sector of the failed read operation

sectors-count: int
failed read operation sector count

Note: This event is rate-limited.

Since: 2.0

Example:

```
<- { "event": "QUORUM_FAILURE",
      "data": { "reference": "usr1", "sector-num": 345435, "sectors-count": 5 },
      "timestamp": { "seconds": 1344522075, "microseconds": 745528 } }
```

QUORUM_REPORT_BAD [Event]

Emitted to report a corruption of a Quorum file

Arguments:

type: QuorumOpType
quorum operation type (Since 2.6)

error: string (optional)
error message. Only present on failure. This field contains a human-readable error message. There are no semantics other than that the block layer reported an error and clients should not try to interpret the error string.

node-name: string
the graph node name of the block driver state

sector-num: int
number of the first sector of the failed read operation

sectors-count: int
failed read operation sector count

Note: This event is rate-limited.

Since: 2.0

Example:

1. Read operation

```
{ "event": "QUORUM_REPORT_BAD",
  "data": { "node-name": "node0", "sector-num": 345435, "sectors-count": 5,
            "type": "read" },
  "timestamp": { "seconds": 1344522075, "microseconds": 745528 } }
```

2. Flush operation

```
{ "event": "QUORUM_REPORT_BAD",
  "data": { "node-name": "node0", "sector-num": 0, "sectors-count": 2097120,
            "type": "flush", "error": "Broken pipe" },
  "timestamp": { "seconds": 1456406829, "microseconds": 291763 } }
```

1.8 Character devices

ChardevInfo [Object]

Information about a character device.

Members:

label: string
the label of the character device

filename: string
the filename of the character device

frontend-open: boolean
shows whether the frontend device attached to this backend (eg. with the `chardev=...` option) is in open or closed state (since 2.1)

Notes: `filename` is encoded using the QEMU command line character device encoding. See the QEMU man page for details.

Since: 0.14.0

query-chardev [Command]

Returns information about current character devices.

Returns: a list of `ChardevInfo`

Since: 0.14.0

Example:

```
-> { "execute": "query-chardev" }
<- {
  "return": [
    {
      "label": "charchannel0",
      "filename": "unix:/var/lib/libvirt/qemu/seabios.rhel6.agent,server",
      "frontend-open": false
    },
    {
      "label": "charmonitor",
      "filename": "unix:/var/lib/libvirt/qemu/seabios.rhel6.monitor,server",
      "frontend-open": true
    },
    {
      "label": "charserial0",
      "filename": "pty:/dev/pts/2",
      "frontend-open": true
    }
  ]
}
```

ChardevBackendInfo [Object]

Information about a character device backend

Members:

name: string
The backend name

Since: 2.0

query-chardev-backends

[Command]

Returns information about character device backends.

Returns: a list of ChardevBackendInfo

Since: 2.0

Example:

```
-> { "execute": "query-chardev-backends" }
<- {
  "return": [
    {
      "name": "udp"
    },
    {
      "name": "tcp"
    },
    {
      "name": "unix"
    },
    {
      "name": "spiceport"
    }
  ]
}
```

DataFormat

[Enum]

An enumeration of data format.

Values:

utf8 Data is a UTF-8 string (RFC 3629)

base64 Data is Base64 encoded binary (RFC 3548)

Since: 1.4

ringbuf-write

[Command]

Write to a ring buffer character device.

Arguments:

device: string
the ring buffer character device name

data: string
data to write

format: `DataFormat` (optional)
 data encoding (default 'utf8').

- base64: data must be base64 encoded text. Its binary decoding gets written.
- utf8: data's UTF-8 encoding is written
- data itself is always Unicode regardless of format, like any other string.

Returns: Nothing on success

Since: 1.4

Example:

```
-> { "execute": "ringbuf-write",
      "arguments": { "device": "foo",
                    "data": "abcdefgh",
                    "format": "utf8" } }

<- { "return": {} }
```

`ringbuf-read` [Command]

Read from a ring buffer character device.

Arguments:

device: `string`
 the ring buffer character device name

size: `int` how many bytes to read at most

format: `DataFormat` (optional)
 data encoding (default 'utf8').

- base64: the data read is returned in base64 encoding.
- utf8: the data read is interpreted as UTF-8. Bug: can screw up when the buffer contains invalid UTF-8 sequences, NUL characters, after the ring buffer lost data, and when reading stops because the size limit is reached.
- The return value is always Unicode regardless of format, like any other string.

Returns: data read from the device

Since: 1.4

Example:

```
-> { "execute": "ringbuf-read",
      "arguments": { "device": "foo",
                    "size": 1000,
                    "format": "utf8" } }

<- { "return": "abcdefgh" }
```

`ChardevCommon` [Object]

Configuration shared across all chardev backends

Members:

`logfile: string` (optional)
The name of a logfile to save output

`logappend: boolean` (optional)
true to append instead of truncate (default to false to truncate)

Since: 2.6

ChardevFile [Object]

Configuration info for file chardevs.

Members:

`in: string` (optional)
The name of the input file

`out: string`
The name of the output file

`append: boolean` (optional)
Open the file in append mode (default false to truncate) (Since 2.6)

The members of `ChardevCommon`

Since: 1.4

ChardevHostdev [Object]

Configuration info for device and pipe chardevs.

Members:

`device: string`
The name of the special file for the device, i.e. `/dev/ttyS0` on Unix or `COM1`: on Windows

The members of `ChardevCommon`

Since: 1.4

ChardevSocket [Object]

Configuration info for (stream) socket chardevs.

Members:

`addr: SocketAddressLegacy`
socket address to listen on (`server=true`) or connect to (`server=false`)

`tls-creds: string` (optional)
the ID of the TLS credentials object (since 2.6)

`server: boolean` (optional)
create server socket (default: true)

`wait: boolean` (optional)
wait for incoming connection on server sockets (default: false).

`nodelay: boolean` (optional)
set `TCP_NODELAY` socket option (default: false)

telnet: `boolean` (optional)
 enable telnet protocol on server sockets (default: false)

tn3270: `boolean` (optional)
 enable tn3270 protocol on server sockets (default: false) (Since: 2.10)

reconnect: `int` (optional)
 For a client socket, if a socket is disconnected, then attempt a reconnect after the given number of seconds. Setting this to zero disables this function. (default: 0) (Since: 2.2)

The members of `ChardevCommon`

Since: 1.4

`ChardevUdp` [Object]

Configuration info for datagram socket chardevs.

Members:

remote: `SocketAddressLegacy`
 remote address

local: `SocketAddressLegacy` (optional)
 local address

The members of `ChardevCommon`

Since: 1.5

`ChardevMux` [Object]

Configuration info for mux chardevs.

Members:

chardev: `string`
 name of the base chardev.

The members of `ChardevCommon`

Since: 1.5

`ChardevStdio` [Object]

Configuration info for stdio chardevs.

Members:

signal: `boolean` (optional)
 Allow signals (such as SIGINT triggered by ^C) be delivered to qemu.
 Default: true in -nographic mode, false otherwise.

The members of `ChardevCommon`

Since: 1.5

`ChardevSpiceChannel` [Object]

Configuration info for spice vm channel chardevs.

Members:

type: `string`
 kind of channel (for example vdagent).

The members of `ChardevCommon`

Since: 1.5

`ChardevSpicePort` [Object]

Configuration info for spice port chardevs.

Members:

`fqdn: string`
name of the channel (see docs/spice-port-fqdn.txt)

The members of `ChardevCommon`

Since: 1.5

`ChardevVC` [Object]

Configuration info for virtual console chardevs.

Members:

`width: int` (optional)
console width, in pixels

`height: int` (optional)
console height, in pixels

`cols: int` (optional)
console width, in chars

`rows: int` (optional)
console height, in chars

The members of `ChardevCommon`

Since: 1.5

`ChardevRingbuf` [Object]

Configuration info for ring buffer chardevs.

Members:

`size: int` (optional)
ring buffer size, must be power of two, default is 65536

The members of `ChardevCommon`

Since: 1.5

`ChardevBackend` [Object]

Configuration info for the new chardev backend.

Members:

`type` One of "file", "serial", "parallel", "pipe", "socket", "udp",
"pty", "null", "mux", "msmouse", "wctablet", "braille", "testdev",
"stdio", "console", "spicevmc", "spiceport", "vc", "ringbuf",
"memory"

data: ChardevFile when type is "file"
 data: ChardevHostdev when type is "serial"
 data: ChardevHostdev when type is "parallel"
 data: ChardevHostdev when type is "pipe"
 data: ChardevSocket when type is "socket"
 data: ChardevUdp when type is "udp"
 data: ChardevCommon when type is "pty"
 data: ChardevCommon when type is "null"
 data: ChardevMux when type is "mux"
 data: ChardevCommon when type is "msmouse"
 data: ChardevCommon when type is "wctablet"
 data: ChardevCommon when type is "braille"
 data: ChardevCommon when type is "testdev"
 data: ChardevStdio when type is "stdio"
 data: ChardevCommon when type is "console"
 data: ChardevSpiceChannel when type is "spicevmc"
 data: ChardevSpicePort when type is "spiceport"
 data: ChardevVC when type is "vc"
 data: ChardevRingbuf when type is "ringbuf"
 data: ChardevRingbuf when type is "memory"
Since: 1.4 (testdev since 2.2, wctablet since 2.9)

ChardevReturn [Object]

Return info about the chardev backend just created.

Members:

pty: string (optional)
 name of the slave pseudoterminal device, present if and only if a chardev
 of type 'pty' was created

Since: 1.4

chardev-add [Command]

Add a character device backend

Arguments:

id: string
 the chardev's ID, must be unique

backend: ChardevBackend
 backend type and parameters

Returns: ChardevReturn.

Since: 1.4

Example:

```

-> { "execute" : "chardev-add",
     "arguments" : { "id" : "foo",
                    "backend" : { "type" : "null", "data" : {} } } }
<- { "return": {} }
  
```



```

-> { "execute" : "chardev-add",
      "arguments" : { "id" : "bar",
                      "backend" : { "type" : "file",
                                    "data" : { "out" : "/tmp/bar.log" } } } } }
<- { "return": {} }

-> { "execute" : "chardev-add",
      "arguments" : { "id" : "baz",
                      "backend" : { "type" : "pty", "data" : {} } } } }
<- { "return": { "pty" : "/dev/pty/42" } }

```

`chardev-change` [Command]

Change a character device backend

Arguments:

`id`: string
the chardev's ID, must exist

`backend`: ChardevBackend
new backend type and parameters

Returns: ChardevReturn.

Since: 2.10

Example:

```

-> { "execute" : "chardev-change",
      "arguments" : { "id" : "baz",
                      "backend" : { "type" : "pty", "data" : {} } } } }
<- { "return": { "pty" : "/dev/pty/42" } }

-> {"execute" : "chardev-change",
    "arguments" : {
      "id" : "charchannel2",
      "backend" : {
        "type" : "socket",
        "data" : {
          "addr" : {
            "type" : "unix" ,
            "data" : {
              "path" : "/tmp/charchannel2.socket"
            }
          }
        },
        "server" : true,
        "wait" : false }}}}
<- {"return": {}}

```

`chardev-remove` [Command]

Remove a character device backend

Arguments:

`id: string`
the chardev's ID, must exist and not be in use

Returns: Nothing on success

Since: 1.4

Example:

```
-> { "execute": "chardev-remove", "arguments": { "id" : "foo" } }
<- { "return": {} }
```

`chardev-send-break` [Command]

Send a break to a character device

Arguments:

`id: string`
the chardev's ID, must exist

Returns: Nothing on success

Since: 2.10

Example:

```
-> { "execute": "chardev-send-break", "arguments": { "id" : "foo" } }
<- { "return": {} }
```

`VSERPORT_CHANGE` [Event]

Emitted when the guest opens or closes a virtio-serial port.

Arguments:

`id: string`
device identifier of the virtio-serial port

`open: boolean`
true if the guest has opened the virtio-serial port

Since: 2.1

Example:

```
<- { "event": "VSERPORT_CHANGE",
      "data": { "id": "channel0", "open": true },
      "timestamp": { "seconds": 1401385907, "microseconds": 422329 } }
```

1.9 Net devices

`set_link` [Command]

Sets the link status of a virtual network adapter.

Arguments:

`name: string`
the device name of the virtual network adapter

up: boolean
 true to set the link status to be up

Returns: Nothing on success If **name** is not a valid network device, DeviceNotFound

Since: 0.14.0

Notes: Not all network adapters support setting link status. This command will succeed even if the network adapter does not support link status notification.

Example:

```
-> { "execute": "set_link",
      "arguments": { "name": "e1000.0", "up": false } }
<- { "return": {} }
```

netdev_add [Command]

Add a network backend.

Arguments:

type: string
 the type of network backend. Current valid values are 'user', 'tap', 'vde', 'socket', 'dump' and 'bridge'

id: string
 the name of the new network backend

Additional arguments depend on the type.

TODO: This command effectively bypasses QAPI completely due to its "additional arguments" business. It shouldn't have been added to the schema in this form. It should be qapified properly, or replaced by a properly qapified command.

Since: 0.14.0

Returns: Nothing on success If **type** is not a valid network backend, DeviceNotFound

Example:

```
-> { "execute": "netdev_add",
      "arguments": { "type": "user", "id": "netdev1",
                    "dnssearch": "example.org" } }
<- { "return": {} }
```

netdev_del [Command]

Remove a network backend.

Arguments:

id: string
 the name of the network backend to remove

Returns: Nothing on success If **id** is not a valid network backend, DeviceNotFound

Since: 0.14.0

Example:

```
-> { "execute": "netdev_del", "arguments": { "id": "netdev1" } }
<- { "return": {} }
```

NetdevNoneOptions [Object]

Use it alone to have zero network devices.

Since: 1.2

NetLegacyNicOptions [Object]

Create a new Network Interface Card.

Members:

netdev: string (optional)

id of -netdev to connect to

macaddr: string (optional)

MAC address

model: string (optional)

device model (e1000, rtl8139, virtio etc.)

addr: string (optional)

PCI device address

vectors: int (optional)

number of MSI-x vectors, 0 to disable MSI-X

Since: 1.2

NetdevUserOptions [Object]

Use the user mode network stack which requires no administrator privilege to run.

Members:

hostname: string (optional)

client hostname reported by the builtin DHCP server

restrict: boolean (optional)

isolate the guest from the host

ipv4: boolean (optional)

whether to support IPv4, default true for enabled (since 2.6)

ipv6: boolean (optional)

whether to support IPv6, default true for enabled (since 2.6)

ip: string (optional)

legacy parameter, use net= instead

net: string (optional)

IP network address that the guest will see, in the form addr[/netmask]
The netmask is optional, and can be either in the form a.b.c.d or as a number of valid top-most bits. Default is 10.0.2.0/24.

host: string (optional)

guest-visible address of the host

tftp: string (optional)

root directory of the built-in TFTP server

bootfile: `string` (optional)
BOOTP filename, for use with `tftp=`

dhcpstart: `string` (optional)
the first of the 16 IPs the built-in DHCP server can assign

dns: `string` (optional)
guest-visible address of the virtual nameserver

dnssearch: `array of String` (optional)
list of DNS suffixes to search, passed as DHCP option to the guest

ipv6-prefix: `string` (optional)
IPv6 network prefix (default is `fec0::`) (since 2.6). The network prefix is given in the usual hexadecimal IPv6 address notation.

ipv6-prefixlen: `int` (optional)
IPv6 network prefix length (default is 64) (since 2.6)

ipv6-host: `string` (optional)
guest-visible IPv6 address of the host (since 2.6)

ipv6-dns: `string` (optional)
guest-visible IPv6 address of the virtual nameserver (since 2.6)

smb: `string` (optional)
root directory of the built-in SMB server

smbserver: `string` (optional)
IP address of the built-in SMB server

hostfwd: `array of String` (optional)
redirect incoming TCP or UDP host connections to guest endpoints

guestfwd: `array of String` (optional)
forward guest TCP connections

Since: 1.2

NetdevTapOptions

[Object]

Connect the host TAP network interface name to the VLAN.

Members:

ifname: `string` (optional)
interface name

fd: `string` (optional)
file descriptor of an already opened tap

fds: `string` (optional)
multiple file descriptors of already opened multiqueue capable tap

script: `string` (optional)
script to initialize the interface

downscript: `string` (optional)
script to shut down the interface

br: **string** (optional)
 bridge name (since 2.8)

helper: **string** (optional)
 command to execute to configure bridge

sndbuf: **int** (optional)
 send buffer limit. Understands [TGMKkb] suffixes.

vnet_hdr: **boolean** (optional)
 enable the IFF_VNET_HDR flag on the tap interface

vhost: **boolean** (optional)
 enable vhost-net network accelerator

vhostfd: **string** (optional)
 file descriptor of an already opened vhost net device

vhostfds: **string** (optional)
 file descriptors of multiple already opened vhost net devices

vhostforce: **boolean** (optional)
 vhost on for non-MSIX virtio guests

queues: **int** (optional)
 number of queues to be created for multiqueue capable tap

poll-us: **int** (optional)
 maximum number of microseconds that could be spent on busy polling for tap (since 2.7)

Since: 1.2

NetdevSocketOptions [Object]

Connect the VLAN to a remote VLAN in another QEMU virtual machine using a TCP socket connection.

Members:

fd: **string** (optional)
 file descriptor of an already opened socket

listen: **string** (optional)
 port number, and optional hostname, to listen on

connect: **string** (optional)
 port number, and optional hostname, to connect to

mcast: **string** (optional)
 UDP multicast address and port number

localaddr: **string** (optional)
 source address and port for multicast and udp packets

udp: **string** (optional)
 UDP unicast address and port number

Since: 1.2

NetdevL2TPv3Options [Object]

Connect the VLAN to Ethernet over L2TPv3 Static tunnel

Members:

src: string
source address

dst: string
destination address

srcport: string (optional)
source port - mandatory for udp, optional for ip

dstport: string (optional)
destination port - mandatory for udp, optional for ip

ipv6: boolean (optional)
force the use of ipv6

udp: boolean (optional)
use the udp version of l2tpv3 encapsulation

cookie64: boolean (optional)
use 64 bit cookies

counter: boolean (optional)
have sequence counter

pincounter: boolean (optional)
pin sequence counter to zero - workaround for buggy implementations or networks with packet reorder

txcookie: int (optional)
32 or 64 bit transmit cookie

rxcookie: int (optional)
32 or 64 bit receive cookie

txsession: int
32 bit transmit session

rxsession: int (optional)
32 bit receive session - if not specified set to the same value as transmit

offset: int (optional)
additional offset - allows the insertion of additional application-specific data before the packet payload

Since: 2.1

NetdevVdeOptions [Object]

Connect the VLAN to a vde switch running on the host.

Members:

sock: string (optional)
socket path

port: `int` (optional)
 port number

group: `string` (optional)
 group owner of socket

mode: `int` (optional)
 permissions for socket

Since: 1.2

NetdevDumpOptions [Object]

Dump VLAN network traffic to a file.

Members:

len: `int` (optional)
 per-packet size limit (64k default). Understands [TGMKkb] suffixes.

file: `string` (optional)
 dump file path (default is qemu-vlan0.pcap)

Since: 1.2

NetdevBridgeOptions [Object]

Connect a host TAP network interface to a host bridge device.

Members:

br: `string` (optional)
 bridge name

helper: `string` (optional)
 command to execute to configure bridge

Since: 1.2

NetdevHubPortOptions [Object]

Connect two or more net clients through a software hub.

Members:

hubid: `int`
 hub identifier number

Since: 1.2

NetdevNetmapOptions [Object]

Connect a client to a netmap-enabled NIC or to a VALE switch port

Members:

ifname: `string`
 Either the name of an existing network interface supported by netmap, or the name of a VALE port (created on the fly). A VALE port name is in the form 'valeXXX:YYY', where XXX and YYY are non-negative integers. XXX identifies a switch and YYY identifies a port of the switch. VALE ports having the same XXX are therefore connected to the same switch.

devname: `string` (optional)
 path of the netmap device (default: `'/dev/netmap'`).

Since: 2.0

NetdevVhostUserOptions [Object]

Vhost-user network backend

Members:

chardev: `string`
 name of a unix socket chardev

vhostforce: `boolean` (optional)
 vhost on for non-MSIX virtio guests (default: `false`).

queues: `int` (optional)
 number of queues to be created for multiqueue vhost-user (default: 1)
 (Since 2.5)

Since: 2.1

NetClientDriver [Enum]

Available netdev drivers.

Values:

`none` Not documented

`nic` Not documented

`user` Not documented

`tap` Not documented

`12tpv3` Not documented

`socket` Not documented

`vde` Not documented

`dump` Not documented

`bridge` Not documented

`hubport` Not documented

`netmap` Not documented

`vhost-user`
 Not documented

Since: 2.7

Netdev [Object]

Captures the configuration of a network device.

Members:

id: `string`
 identifier for monitor commands.

type: `NetClientDriver`

Specify the driver used for interpreting remaining arguments.

The members of `NetdevNoneOptions` when `type` is "none"

The members of `NetLegacyNicOptions` when `type` is "nic"

The members of `NetdevUserOptions` when `type` is "user"

The members of `NetdevTapOptions` when `type` is "tap"

The members of `NetdevL2TPv3Options` when `type` is "l2tpv3"

The members of `NetdevSocketOptions` when `type` is "socket"

The members of `NetdevVdeOptions` when `type` is "vde"

The members of `NetdevDumpOptions` when `type` is "dump"

The members of `NetdevBridgeOptions` when `type` is "bridge"

The members of `NetdevHubPortOptions` when `type` is "hubport"

The members of `NetdevNetmapOptions` when `type` is "netmap"

The members of `NetdevVhostUserOptions` when `type` is "vhost-user"

Since: 1.2

'l2tpv3' - since 2.1

NetLegacy

[Object]

Captures the configuration of a network device; legacy.

Members:

vlan: `int` (optional)
vlan number

id: `string` (optional)
identifier for monitor commands

name: `string` (optional)
identifier for monitor commands, ignored if `id` is present

opts: `NetLegacyOptions`
device type specific properties (legacy)

Since: 1.2

NetLegacyOptionsType

[Enum]

Values:

`none` Not documented

`nic` Not documented

`user` Not documented

`tap` Not documented

`l2tpv3` Not documented

`socket` Not documented

`vde` Not documented

`dump` Not documented

bridge Not documented
 netmap Not documented
 vhost-user
 Not documented

Since: 1.2

NetLegacyOptions [Object]

Like Netdev, but for use only by the legacy command line options

Members:

type: NetLegacyOptionsType
 Not documented

The members of NetdevNoneOptions when type is "none"

The members of NetLegacyNicOptions when type is "nic"

The members of NetdevUserOptions when type is "user"

The members of NetdevTapOptions when type is "tap"

The members of NetdevL2TPv3Options when type is "l2tpv3"

The members of NetdevSocketOptions when type is "socket"

The members of NetdevVdeOptions when type is "vde"

The members of NetdevDumpOptions when type is "dump"

The members of NetdevBridgeOptions when type is "bridge"

The members of NetdevNetmapOptions when type is "netmap"

The members of NetdevVhostUserOptions when type is "vhost-user"

Since: 1.2

NetFilterDirection [Enum]

Indicates whether a netfilter is attached to a netdev's transmit queue or receive queue or both.

Values:

all the filter is attached both to the receive and the transmit queue of the netdev (default).

rx the filter is attached to the receive queue of the netdev, where it will receive packets sent to the netdev.

tx the filter is attached to the transmit queue of the netdev, where it will receive packets sent by the netdev.

Since: 2.5

RxState [Enum]

Packets receiving state

Values:

normal filter assigned packets according to the mac-table

none don't receive any assigned packet

all receive all assigned packets

Since: 1.6

RxFilterInfo [Object]

Rx-filter information for a NIC.

Members:

name: string
net client name

promiscuous: boolean
whether promiscuous mode is enabled

multicast: RxState
multicast receive state

unicast: RxState
unicast receive state

vlan: RxState
vlan receive state (Since 2.0)

broadcast-allowed: boolean
whether to receive broadcast

multicast-overflow: boolean
multicast table is overflowed or not

unicast-overflow: boolean
unicast table is overflowed or not

main-mac: string
the main macaddr string

vlan-table: array of int
a list of active vlan id

unicast-table: array of string
a list of unicast macaddr string

multicast-table: array of string
a list of multicast macaddr string

Since: 1.6

query-rx-filter [Command]

Return rx-filter information for all NICs (or for the given NIC).

Arguments:

name: string (optional)
net client name

Returns: list of **RxFilterInfo** for all NICs (or for the given NIC). Returns an error if the given **name** doesn't exist, or given NIC doesn't support rx-filter querying, or given net client isn't a NIC.

Since: 1.6

Example:

```
-> { "execute": "query-rx-filter", "arguments": { "name": "vnet0" } }
```

```

<- { "return": [
  {
    "promiscuous": true,
    "name": "vnet0",
    "main-mac": "52:54:00:12:34:56",
    "unicast": "normal",
    "vlan": "normal",
    "vlan-table": [
      4,
      0
    ],
    "unicast-table": [
    ],
    "multicast": "normal",
    "multicast-overflow": false,
    "unicast-overflow": false,
    "multicast-table": [
      "01:00:5e:00:00:01",
      "33:33:00:00:00:01",
      "33:33:ff:12:34:56"
    ],
    "broadcast-allowed": false
  }
]
}

```

NIC_RX_FILTER_CHANGED [Event]

Emitted once until the 'query-rx-filter' command is executed, the first event will always be emitted

Arguments:

name: string (optional)
net client name

path: string
device path

Since: 1.6

Example:

```

<- { "event": "NIC_RX_FILTER_CHANGED",
  "data": { "name": "vnet0",
    "path": "/machine/peripheral/vnet0/virtio-backend" },
  "timestamp": { "seconds": 1368697518, "microseconds": 326866 } }
}

```

1.10 Rocker switch device

RockerSwitch

[Object]

Rocker switch information.

Members:

`name: string`
 switch name

`id: int` switch ID

`ports: int`
 number of front-panel ports

Since: 2.4

`query-rocker` [Command]

Return rocker switch information.

Arguments:

`name: string`
 Not documented

Returns: Rocker information**Since:** 2.4**Example:**

```
-> { "execute": "query-rocker", "arguments": { "name": "sw1" } }
<- { "return": {"name": "sw1", "ports": 2, "id": 1327446905938}}
```

`RockerPortDuplex` [Enum]

An enumeration of port duplex states.

Values:

`half` half duplex

`full` full duplex

Since: 2.4

`RockerPortAutoneg` [Enum]

An enumeration of port autoneg states.

Values:

`off` autoneg is off

`on` autoneg is on

Since: 2.4

`RockerPort` [Object]

Rocker switch port information.

Members:

`name: string`
 port name

`enabled: boolean`
 port is enabled for I/O

link-up: boolean
physical link is UP on port

speed: int
port link speed in Mbps

duplex: RockerPortDuplex
port link duplex

autoneg: RockerPortAutoneg
port link autoneg

Since: 2.4

query-rocker-ports [Command]

Return rocker switch port information.

Arguments:

name: string
Not documented

Returns: a list of RockerPort information

Since: 2.4

Example:

```
-> { "execute": "query-rocker-ports", "arguments": { "name": "sw1" } }
<- { "return": [ {"duplex": "full", "enabled": true, "name": "sw1.1",
                  "autoneg": "off", "link-up": true, "speed": 10000},
                 {"duplex": "full", "enabled": true, "name": "sw1.2",
                  "autoneg": "off", "link-up": true, "speed": 10000}
    ] }
```

RockerOfDpaFlowKey [Object]

Rocker switch OF-DPA flow key

Members:

priority: int
key priority, 0 being lowest priority

tbl-id: int
flow table ID

in-pport: int (optional)
physical input port

tunnel-id: int (optional)
tunnel ID

vlan-id: int (optional)
VLAN ID

eth-type: int (optional)
Ethernet header type

eth-src: `string` (optional)
Ethernet header source MAC address

eth-dst: `string` (optional)
Ethernet header destination MAC address

ip-proto: `int` (optional)
IP Header protocol field

ip-tos: `int` (optional)
IP header TOS field

ip-dst: `string` (optional)
IP header destination address

Note: optional members may or may not appear in the flow key depending if they're relevant to the flow key.

Since: 2.4

RockerOfDpaFlowMask [Object]

Rocker switch OF-DPA flow mask

Members:

in-pport: `int` (optional)
physical input port

tunnel-id: `int` (optional)
tunnel ID

vlan-id: `int` (optional)
VLAN ID

eth-src: `string` (optional)
Ethernet header source MAC address

eth-dst: `string` (optional)
Ethernet header destination MAC address

ip-proto: `int` (optional)
IP Header protocol field

ip-tos: `int` (optional)
IP header TOS field

Note: optional members may or may not appear in the flow mask depending if they're relevant to the flow mask.

Since: 2.4

RockerOfDpaFlowAction [Object]

Rocker switch OF-DPA flow action

Members:

goto-tbl: `int` (optional)
next table ID

group-id: int (optional)
 group ID
tunnel-lport: int (optional)
 tunnel logical port ID
vlan-id: int (optional)
 VLAN ID
new-vlan-id: int (optional)
 new VLAN ID
out-pport: int (optional)
 physical output port

Note: optional members may or may not appear in the flow action depending if they're relevant to the flow action.

Since: 2.4

RockerOfDpaFlow [Object]

Rocker switch OF-DPA flow

Members:

cookie: int
 flow unique cookie ID
hits: int count of matches (hits) on flow
key: RockerOfDpaFlowKey
 flow key
mask: RockerOfDpaFlowMask
 flow mask
action: RockerOfDpaFlowAction
 flow action

Since: 2.4

query-rocker-of-dpa-flows [Command]

Return rocker OF-DPA flow information.

Arguments:

name: string
 switch name
tbl-id: int (optional)
 flow table ID. If tbl-id is not specified, returns flow information for all tables.

Returns: rocker OF-DPA flow information

Since: 2.4

Example:

```
-> { "execute": "query-rocker-of-dpa-flows",
```

```

    "arguments": { "name": "sw1" } }
  <- { "return": [ {"key": {"in-pport": 0, "priority": 1, "tbl-id": 0},
                  "hits": 138,
                  "cookie": 0,
                  "action": {"goto-tbl": 10},
                  "mask": {"in-pport": 4294901760}
                },
        {...more...},
  ]}

```

RockerOfDpaGroup

[Object]

Rocker switch OF-DPA group

Members:

```

id: int    group unique ID
type: int  group type
vlan-id: int (optional)
           VLAN ID
pport: int (optional)
        physical port number
index: int (optional)
        group index, unique with group type
out-pport: int (optional)
           output physical port number
group-id: int (optional)
           next group ID
set-vlan-id: int (optional)
            VLAN ID to set
pop-vlan: int (optional)
          pop VLAN headr from packet
group-ids: array of int (optional)
           list of next group IDs
set-eth-src: string (optional)
            set source MAC address in Ethernet header
set-eth-dst: string (optional)
            set destination MAC address in Ethernet header
ttl-check: int (optional)
           perform TTL check

```

Note: optional members may or may not appear in the group depending if they're relevant to the group type.

Since: 2.4

`query-rocker-of-dpa-groups` [Command]

Return rocker OF-DPA group information.

Arguments:

`name: string`

switch name

`type: int` (optional)

group type. If type is not specified, returns group information for all group types.

Returns: rocker OF-DPA group information

Since: 2.4

Example:

```
-> { "execute": "query-rocker-of-dpa-groups",
      "arguments": { "name": "sw1" } }
<- { "return": [ {"type": 0, "out-pport": 2,
                  "pport": 2, "vlan-id": 3841,
                  "pop-vlan": 1, "id": 251723778},
                 {"type": 0, "out-pport": 0,
                  "pport": 0, "vlan-id": 3841,
                  "pop-vlan": 1, "id": 251723776},
                 {"type": 0, "out-pport": 1,
                  "pport": 1, "vlan-id": 3840,
                  "pop-vlan": 1, "id": 251658241},
                 {"type": 0, "out-pport": 0,
                  "pport": 0, "vlan-id": 3840,
                  "pop-vlan": 1, "id": 251658240}
      ] }
```

1.11 TPM (trusted platform module) devices

`TpmModel` [Enum]

An enumeration of TPM models

Values:

`tpm-tis` TPM TIS model

Since: 1.5

`query-tpm-models` [Command]

Return a list of supported TPM models

Returns: a list of `TpmModel`

Since: 1.5

Example:

```
-> { "execute": "query-tpm-models" }
<- { "return": [ "tpm-tis" ] }
```

- TpmType** [Enum]
 An enumeration of TPM types
Values:
 passthrough
 TPM passthrough type
Since: 1.5
- query-tpm-types** [Command]
 Return a list of supported TPM types
Returns: a list of TpmType
Since: 1.5
Example:
 -> { "execute": "query-tpm-types" }
 <- { "return": ["passthrough"] }
- TPMPassthroughOptions** [Object]
 Information about the TPM passthrough type
Members:
 path: string (optional)
 string describing the path used for accessing the TPM device
 cancel-path: string (optional)
 string showing the TPM's sysfs cancel file for cancellation of TPM commands while they are executing
Since: 1.5
- TpmTypeOptions** [Object]
 A union referencing different TPM backend types' configuration options
Members:
 type 'passthrough' The configuration options for the TPM passthrough type
 data: TPMPassthroughOptions when type is "passthrough"
Since: 1.5
- TPMInfo** [Object]
 Information about the TPM
Members:
 id: string
 The Id of the TPM
 model: TpmModel
 The TPM frontend model
 options: TpmTypeOptions
 The TPM (backend) type configuration options
Since: 1.5

`query-tpm` [Command]

Return information about the TPM device

Returns: TPMInfo on success

Since: 1.5

Example:

```
-> { "execute": "query-tpm" }
<- { "return":
  [
    { "model": "tpm-tis",
      "options":
        { "type": "passthrough",
          "data":
            { "cancel-path": "/sys/class/misc/tpm0/device/cancel",
              "path": "/dev/tpm0"
            }
          },
      "id": "tpm0"
    }
  ]
}
```

1.12 Remote desktop

`set_password` [Command]

Sets the password of a remote display session.

Arguments:

`protocol`: string

‘vnc’ to modify the VNC server password ‘spice’ to modify the Spice server password

`password`: string

the new password

`connected`: string (optional)

how to handle existing clients when changing the password. If nothing is specified, defaults to ‘keep’ ‘fail’ to fail the command if clients are connected ‘disconnect’ to disconnect existing clients ‘keep’ to maintain existing clients

Returns: Nothing on success If Spice is not enabled, DeviceNotFound

Since: 0.14.0

Example:

```
-> { "execute": "set_password", "arguments": { "protocol": "vnc",
                                             "password": "secret" } }
<- { "return": {} }
```

`expire_password` [Command]

Expire the password of a remote display server.

Arguments:

`protocol: string`

the name of the remote display protocol ‘vnc’ or ‘spice’

`time: string`

when to expire the password. ‘now’ to expire the password immediately ‘never’ to cancel password expiration ‘+INT’ where INT is the number of seconds from now (integer) ‘INT’ where INT is the absolute time in seconds

Returns: Nothing on success If `protocol` is ‘spice’ and Spice is not active, DeviceNotFound

Since: 0.14.0

Notes: Time is relative to the server and currently there is no way to coordinate server time with client time. It is not recommended to use the absolute time version of the `time` parameter unless you’re sure you are on the same machine as the QEMU instance.

Example:

```
-> { "execute": "expire_password", "arguments": { "protocol": "vnc",
                                                "time": "+60" } }
<- { "return": {} }
```

`screendump` [Command]

Write a PPM of the VGA screen to a file.

Arguments:

`filename: string`

the path of a new PPM file to store the image

Returns: Nothing on success

Since: 0.14.0

Example:

```
-> { "execute": "screendump",
      "arguments": { "filename": "/tmp/image" } }
<- { "return": {} }
```

1.12.1 Spice

`SpiceBasicInfo` [Object]

The basic information for SPICE network connection

Members:

`host: string`

IP address

`port: string`

port number

family: `NetworkAddressFamily`
address family

Since: 2.1

SpiceServerInfo [Object]

Information about a SPICE server

Members:

auth: `string` (optional)
authentication method

The members of `SpiceBasicInfo`

Since: 2.1

SpiceChannel [Object]

Information about a SPICE client channel.

Members:

connection-id: `int`
SPICE connection id number. All channels with the same id belong to the same SPICE session.

channel-type: `int`
SPICE channel type number. "1" is the main control channel, filter for this one if you want to track spice sessions only

channel-id: `int`
SPICE channel ID number. Usually "0", might be different when multiple channels of the same type exist, such as multiple display channels in a multihead setup

tls: `boolean`
true if the channel is encrypted, false otherwise.

The members of `SpiceBasicInfo`

Since: 0.14.0

SpiceQueryMouseMode [Enum]

An enumeration of Spice mouse states.

Values:

client Mouse cursor position is determined by the client.

server Mouse cursor position is determined by the server.

unknown No information is available about mouse mode used by the spice server.

Note: `spice/enums.h` has a `SpiceMouseMode` already, hence the name.

Since: 1.1

SpiceInfo [Object]

Information about the SPICE session.

Members:

enabled: `boolean`

true if the SPICE server is enabled, false otherwise

migrated: `boolean`

true if the last guest migration completed and spice migration had completed as well. false otherwise. (since 1.4)

host: `string` (optional)

The hostname the SPICE server is bound to. This depends on the name resolution on the host and may be an IP address.

port: `int` (optional)

The SPICE server's port number.

compiled-version: `string` (optional)

SPICE server version.

tls-port: `int` (optional)

The SPICE server's TLS port number.

auth: `string` (optional)

the current authentication type used by the server 'none' if no authentication is being used 'spice' uses SASL or direct TLS authentication, depending on command line options

mouse-mode: `SpiceQueryMouseMode`

The mode in which the mouse cursor is displayed currently. Can be determined by the client or the server, or unknown if spice server doesn't provide this information. (since: 1.1)

channels: `array of SpiceChannel` (optional)

a list of `SpiceChannel` for each active spice channel

Since: 0.14.0

query-spice [Command]

Returns information about the current SPICE server

Returns: `SpiceInfo`

Since: 0.14.0

Example:

```
-> { "execute": "query-spice" }
<- { "return": {
    "enabled": true,
    "auth": "spice",
    "port": 5920,
    "tls-port": 5921,
    "host": "0.0.0.0",
    "channels": [
```



```

    {
      "port": "54924",
      "family": "ipv4",
      "channel-type": 1,
      "connection-id": 1804289383,
      "host": "127.0.0.1",
      "channel-id": 0,
      "tls": true
    },
    {
      "port": "36710",
      "family": "ipv4",
      "channel-type": 4,
      "connection-id": 1804289383,
      "host": "127.0.0.1",
      "channel-id": 0,
      "tls": false
    },
    [ ... more channels follow ... ]
  ]
}

```

SPICE_CONNECTED

[Event]

Emitted when a SPICE client establishes a connection

Arguments:

server: SpiceBasicInfo
server information

client: SpiceBasicInfo
client information

Since: 0.14.0

Example:

```

<- { "timestamp": {"seconds": 1290688046, "microseconds": 388707},
      "event": "SPICE_CONNECTED",
      "data": {
        "server": { "port": "5920", "family": "ipv4", "host": "127.0.0.1"},
        "client": {"port": "52873", "family": "ipv4", "host": "127.0.0.1"}
      }
}

```

SPICE_INITIALIZED

[Event]

Emitted after initial handshake and authentication takes place (if any) and the SPICE channel is up and running

Arguments:

server: SpiceServerInfo
server information

```
client: SpiceChannel
    client information
```

Since: 0.14.0

Example:

```
<- { "timestamp": {"seconds": 1290688046, "microseconds": 417172},
    "event": "SPICE_INITIALIZED",
    "data": {"server": {"auth": "spice", "port": "5921",
                       "family": "ipv4", "host": "127.0.0.1"},
            "client": {"port": "49004", "family": "ipv4", "channel-type": 3,
                       "connection-id": 1804289383, "host": "127.0.0.1",
                       "channel-id": 0, "tls": true}
    }}
```

SPICE_DISCONNECTED [Event]

Emitted when the SPICE connection is closed

Arguments:

```
server: SpiceBasicInfo
    server information
```

```
client: SpiceBasicInfo
    client information
```

Since: 0.14.0

Example:

```
<- { "timestamp": {"seconds": 1290688046, "microseconds": 388707},
    "event": "SPICE_DISCONNECTED",
    "data": {
        "server": { "port": "5920", "family": "ipv4", "host": "127.0.0.1"},
        "client": {"port": "52873", "family": "ipv4", "host": "127.0.0.1"}
    }
}}
```

SPICE_MIGRATE_COMPLETED [Event]

Emitted when SPICE migration has completed

Since: 1.3

Example:

```
<- { "timestamp": {"seconds": 1290688046, "microseconds": 417172},
    "event": "SPICE_MIGRATE_COMPLETED" }
```

1.12.2 VNC

VncBasicInfo [Object]

The basic information for vnc network connection

Members:

```
host: string
    IP address
```

service: string
The service name of the vnc port. This may depend on the host system's service database so symbolic names should not be relied on.

family: NetworkAddressFamily
address family

websocket: boolean
true in case the socket is a websocket (since 2.3).

Since: 2.1

VncServerInfo [Object]

The network connection information for server

Members:

auth: string (optional)
authentication method used for the plain (non-websocket) VNC server

The members of **VncBasicInfo**

Since: 2.1

VncClientInfo [Object]

Information about a connected VNC client.

Members:

x509_dname: string (optional)
If x509 authentication is in use, the Distinguished Name of the client.

sasl_username: string (optional)
If SASL authentication is in use, the SASL username used for authentication.

The members of **VncBasicInfo**

Since: 0.14.0

VncInfo [Object]

Information about the VNC session.

Members:

enabled: boolean
true if the VNC server is enabled, false otherwise

host: string (optional)
The hostname the VNC server is bound to. This depends on the name resolution on the host and may be an IP address.

family: NetworkAddressFamily (optional)
'ipv6' if the host is listening for IPv6 connections 'ipv4' if the host is listening for IPv4 connections 'unix' if the host is listening on a unix domain socket 'unknown' otherwise

service: string (optional)

The service name of the server's port. This may depends on the host system's service database so symbolic names should not be relied on.

auth: string (optional)

the current authentication type used by the server 'none' if no authentication is being used 'vnc' if VNC authentication is being used 'vnc+encrypt+plain' if VEncrypt is used with plain text authentication 'vnc+encrypt+tls+none' if VEncrypt is used with TLS and no authentication 'vnc+encrypt+tls+vnc' if VEncrypt is used with TLS and VNC authentication 'vnc+encrypt+tls+plain' if VEncrypt is used with TLS and plain text auth 'vnc+encrypt+x509+none' if VEncrypt is used with x509 and no auth 'vnc+encrypt+x509+vnc' if VEncrypt is used with x509 and VNC auth 'vnc+encrypt+x509+plain' if VEncrypt is used with x509 and plain text auth 'vnc+encrypt+tls+sasl' if VEncrypt is used with TLS and SASL auth 'vnc+encrypt+x509+sasl' if VEncrypt is used with x509 and SASL auth

clients: array of VncClientInfo (optional)

a list of VncClientInfo of all currently connected clients

Since: 0.14.0

VncPrimaryAuth

[Enum]

vnc primary authentication method.

Values:

none	Not documented
vnc	Not documented
ra2	Not documented
ra2ne	Not documented
tight	Not documented
ultra	Not documented
tls	Not documented
vnc+encrypt	Not documented
sasl	Not documented

Since: 2.3

VncVncEncryptSubAuth

[Enum]

vnc sub authentication method with vnc+encrypt.

Values:

plain	Not documented
tls-none	Not documented
x509-none	Not documented

`tls-vnc` Not documented

`x509-vnc` Not documented

`tls-plain`
Not documented

`x509-plain`
Not documented

`tls-sasl` Not documented

`x509-sasl`
Not documented

Since: 2.3

VncServerInfo2 [Object]

The network connection information for server

Members:

`auth`: `VncPrimaryAuth`
The current authentication type used by the servers

`vencrypt`: `VncVencryptSubAuth` (optional)
The vencrypt sub authentication type used by the servers, only specified in case `auth == vencrypt`.

The members of `VncBasicInfo`

Since: 2.9

VncInfo2 [Object]

Information about a vnc server

Members:

`id`: `string`
vnc server name.

`server`: `array of VncServerInfo2`
A list of `VncBasicInfo` describing all listening sockets. The list can be empty (in case the vnc server is disabled). It also may have multiple entries: normal + websocket, possibly also ipv4 + ipv6 in the future.

`clients`: `array of VncClientInfo`
A list of `VncClientInfo` of all currently connected clients. The list can be empty, for obvious reasons.

`auth`: `VncPrimaryAuth`
The current authentication type used by the non-websockets servers

`vencrypt`: `VncVencryptSubAuth` (optional)
The vencrypt authentication type used by the servers, only specified in case `auth == vencrypt`.

`display`: `string` (optional)
The display device the vnc server is linked to.

Since: 2.3

query-vnc [Command]

Returns information about the current VNC server

Returns: VncInfo

Since: 0.14.0

Example:

```
-> { "execute": "query-vnc" }
<- { "return": {
    "enabled":true,
    "host":"0.0.0.0",
    "service":"50402",
    "auth":"vnc",
    "family":"ipv4",
    "clients":[
      {
        "host":"127.0.0.1",
        "service":"50401",
        "family":"ipv4"
      }
    ]
  }
}
```

query-vnc-servers [Command]

Returns a list of vnc servers. The list can be empty.

Returns: a list of VncInfo2

Since: 2.3

change-vnc-password [Command]

Change the VNC server password.

Arguments:

password: string
the new password to use with VNC authentication

Since: 1.1

Notes: An empty password in this command will set the password to the empty string. Existing clients are unaffected by executing this command.

VNC_CONNECTED [Event]

Emitted when a VNC client establishes a connection

Arguments:

server: VncServerInfo
server information

client: VncBasicInfo
client information

Note: This event is emitted before any authentication takes place, thus the authentication ID is not provided

Since: 0.13.0

Example:

```
<- { "event": "VNC_CONNECTED",
      "data": {
        "server": { "auth": "sasl", "family": "ipv4",
                    "service": "5901", "host": "0.0.0.0" },
        "client": { "family": "ipv4", "service": "58425",
                    "host": "127.0.0.1" } },
      "timestamp": { "seconds": 1262976601, "microseconds": 975795 } }
```

VNC_INITIALIZED [Event]

Emitted after authentication takes place (if any) and the VNC session is made active

Arguments:

server: VncServerInfo
server information

client: VncClientInfo
client information

Since: 0.13.0

Example:

```
<- { "event": "VNC_INITIALIZED",
      "data": {
        "server": { "auth": "sasl", "family": "ipv4",
                    "service": "5901", "host": "0.0.0.0"},
        "client": { "family": "ipv4", "service": "46089",
                    "host": "127.0.0.1", "sasl_username": "luiz" } },
      "timestamp": { "seconds": 1263475302, "microseconds": 150772 } }
```

VNC_DISCONNECTED [Event]

Emitted when the connection is closed

Arguments:

server: VncServerInfo
server information

client: VncClientInfo
client information

Since: 0.13.0

Example:

```
<- { "event": "VNC_DISCONNECTED",
      "data": {
        "server": { "auth": "sasl", "family": "ipv4",
                    "service": "5901", "host": "0.0.0.0" },
        "client": { "family": "ipv4", "service": "58425",
```

```

        "host": "127.0.0.1", "sasl_username": "luiz" } },
    "timestamp": { "seconds": 1262976601, "microseconds": 975795 } }

```

1.13 Input

MouseInfo [Object]

Information about a mouse device.

Members:

```

name: string
    the name of the mouse device

index: int
    the index of the mouse device

current: boolean
    true if this device is currently receiving mouse events

absolute: boolean
    true if this device supports absolute coordinates as input

```

Since: 0.14.0

query-mice [Command]

Returns information about each active mouse device

Returns: a list of `MouseInfo` for each device

Since: 0.14.0

Example:

```

-> { "execute": "query-mice" }
<- { "return": [
    {
        "name":"QEMU Microsoft Mouse",
        "index":0,
        "current":false,
        "absolute":false
    },
    {
        "name":"QEMU PS/2 Mouse",
        "index":1,
        "current":true,
        "absolute":true
    }
]
}

```

QKeyCode [Enum]

An enumeration of key name.

This is used by the `send-key` command.

Values:

`unmapped` since 2.0

<code>pause</code>	since 2.0
<code>ro</code>	since 2.4
<code>kp_comma</code>	since 2.4
<code>kp_equals</code>	since 2.6
<code>power</code>	since 2.6
<code>hiragana</code>	since 2.9
<code>henkan</code>	since 2.9
<code>yen</code>	since 2.9
<code>sleep</code>	since 2.10
<code>wake</code>	since 2.10
<code>audionext</code>	since 2.10
<code>audioprev</code>	since 2.10
<code>audiostop</code>	since 2.10
<code>audioplay</code>	since 2.10
<code>audiomute</code>	since 2.10
<code>volumeup</code>	since 2.10
<code>volumedown</code>	since 2.10
<code>mediaselect</code>	since 2.10
<code>mail</code>	since 2.10
<code>calculator</code>	since 2.10
<code>computer</code>	since 2.10
<code>ac_home</code>	since 2.10
<code>ac_back</code>	since 2.10
<code>ac_forward</code>	since 2.10
<code>ac_refresh</code>	since 2.10

<code>ac_bookmarks</code>	since 2.10 altgr, altgr_r: dropped in 2.10
<code>shift</code>	Not documented
<code>shift_r</code>	Not documented
<code>alt</code>	Not documented
<code>alt_r</code>	Not documented
<code>ctrl</code>	Not documented
<code>ctrl_r</code>	Not documented
<code>menu</code>	Not documented
<code>esc</code>	Not documented
<code>1</code>	Not documented
<code>2</code>	Not documented
<code>3</code>	Not documented
<code>4</code>	Not documented
<code>5</code>	Not documented
<code>6</code>	Not documented
<code>7</code>	Not documented
<code>8</code>	Not documented
<code>9</code>	Not documented
<code>0</code>	Not documented
<code>minus</code>	Not documented
<code>equal</code>	Not documented
<code>backspace</code>	Not documented
<code>tab</code>	Not documented
<code>q</code>	Not documented
<code>w</code>	Not documented
<code>e</code>	Not documented
<code>r</code>	Not documented
<code>t</code>	Not documented
<code>y</code>	Not documented
<code>u</code>	Not documented
<code>i</code>	Not documented

o Not documented

p Not documented

`bracket_left`
Not documented

`bracket_right`
Not documented

`ret` Not documented

a Not documented

s Not documented

d Not documented

f Not documented

g Not documented

h Not documented

j Not documented

k Not documented

l Not documented

`semicolon`
Not documented

`apostrophe`
Not documented

`grave_accent`
Not documented

`backslash`
Not documented

z Not documented

x Not documented

c Not documented

v Not documented

b Not documented

n Not documented

m Not documented

`comma` Not documented

`dot` Not documented

`slash` Not documented

`asterisk` Not documented

<code>spc</code>	Not documented
<code>caps_lock</code>	Not documented
<code>f1</code>	Not documented
<code>f2</code>	Not documented
<code>f3</code>	Not documented
<code>f4</code>	Not documented
<code>f5</code>	Not documented
<code>f6</code>	Not documented
<code>f7</code>	Not documented
<code>f8</code>	Not documented
<code>f9</code>	Not documented
<code>f10</code>	Not documented
<code>num_lock</code>	Not documented
<code>scroll_lock</code>	Not documented
<code>kp_divide</code>	Not documented
<code>kp_multiply</code>	Not documented
<code>kp_subtract</code>	Not documented
<code>kp_add</code>	Not documented
<code>kp_enter</code>	Not documented
<code>kp_decimal</code>	Not documented
<code>sysrq</code>	Not documented
<code>kp_0</code>	Not documented
<code>kp_1</code>	Not documented
<code>kp_2</code>	Not documented
<code>kp_3</code>	Not documented
<code>kp_4</code>	Not documented
<code>kp_5</code>	Not documented
<code>kp_6</code>	Not documented
<code>kp_7</code>	Not documented

<code>kp_8</code>	Not documented
<code>kp_9</code>	Not documented
<code>less</code>	Not documented
<code>f11</code>	Not documented
<code>f12</code>	Not documented
<code>print</code>	Not documented
<code>home</code>	Not documented
<code>pgup</code>	Not documented
<code>pgdn</code>	Not documented
<code>end</code>	Not documented
<code>left</code>	Not documented
<code>up</code>	Not documented
<code>down</code>	Not documented
<code>right</code>	Not documented
<code>insert</code>	Not documented
<code>delete</code>	Not documented
<code>stop</code>	Not documented
<code>again</code>	Not documented
<code>props</code>	Not documented
<code>undo</code>	Not documented
<code>front</code>	Not documented
<code>copy</code>	Not documented
<code>open</code>	Not documented
<code>paste</code>	Not documented
<code>find</code>	Not documented
<code>cut</code>	Not documented
<code>lf</code>	Not documented
<code>help</code>	Not documented
<code>meta_l</code>	Not documented
<code>meta_r</code>	Not documented
<code>compose</code>	Not documented

Since: 1.3.0

KeyValue [Object]

Represents a keyboard key.

Members:

type One of "number", "qcode"

data: int when type is "number"

data: QKeyCode when type is "qcode"

Since: 1.3.0

send-key [Command]

Send keys to guest.

Arguments:

keys: array of KeyValue

An array of `KeyValue` elements. All `KeyValues` in this array are simultaneously sent to the guest. A `KeyValue.number` value is sent directly to the guest, while `KeyValue.qcode` must be a valid `QKeyCode` value

hold-time: int (optional)

time to delay key up events, milliseconds. Defaults to 100

Returns: Nothing on success If key is unknown or redundant, `InvalidParameter`

Since: 1.3.0

Example:

```
-> { "execute": "send-key",
    "arguments": { "keys": [ { "type": "qcode", "data": "ctrl" },
                            { "type": "qcode", "data": "alt" },
                            { "type": "qcode", "data": "delete" } ] } } }
<- { "return": {} }
```

InputButton [Enum]

Button of a pointer input device (mouse, tablet).

Values:

side front side button of a 5-button mouse (since 2.9)

extra rear side button of a 5-button mouse (since 2.9)

left Not documented

middle Not documented

right Not documented

wheel-up Not documented

wheel-down
Not documented

Since: 2.0

- InputAxis** [Enum]
Position axis of a pointer input device (mouse, tablet).
Values:
x Not documented
y Not documented
Since: 2.0
- InputKeyEvent** [Object]
Keyboard input event.
Members:
key: KeyValue
Which key this event is for.
down: boolean
True for key-down and false for key-up events.
Since: 2.0
- InputBtnEvent** [Object]
Pointer button input event.
Members:
button: InputButton
Which button this event is for.
down: boolean
True for key-down and false for key-up events.
Since: 2.0
- InputMoveEvent** [Object]
Pointer motion input event.
Members:
axis: InputAxis
Which axis is referenced by `value`.
value: int
Pointer position. For absolute coordinates the valid range is 0 -> 0x7fff
Since: 2.0
- InputEvent** [Object]
Input event union.
Members:
type the input type, one of:
– 'key': Input event of Keyboard
– 'btn': Input event of pointer buttons
– 'rel': Input event of relative pointer motion

– 'abs': Input event of absolute pointer motion

data: InputKeyEvent when type is "key"
 data: InputBtnEvent when type is "btn"
 data: InputMoveEvent when type is "rel"
 data: InputMoveEvent when type is "abs"
Since: 2.0

`input-send-event` [Command]

Send input event(s) to guest.

Arguments:

device: string (optional)
 display device to send event(s) to.

head: int (optional)
 head to send event(s) to, in case the display device supports multiple scanouts.

events: array of InputEvent
 List of InputEvent union.

Returns: Nothing on success.

The **device** and **head** parameters can be used to send the input event to specific input devices in case (a) multiple input devices of the same kind are added to the virtual machine and (b) you have configured input routing (see docs/multiseat.txt) for those input devices. The parameters work exactly like the device and head properties of input devices. If **device** is missing, only devices that have no input routing config are admissible. If **device** is specified, both input devices with and without input routing config are admissible, but devices with input routing config take precedence.

Since: 2.6

Note: The consoles are visible in the qom tree, under `/backend/console[$index]`. They have a device link and head property, so it is possible to map which console belongs to which device and display.

Example:

1. Press left mouse button.

```
-> { "execute": "input-send-event",
      "arguments": { "device": "video0",
                    "events": [ { "type": "btn",
                                  "data" : { "down": true, "button": "left" } } ] } }
<- { "return": {} }
```

```
-> { "execute": "input-send-event",
      "arguments": { "device": "video0",
                    "events": [ { "type": "btn",
                                  "data" : { "down": false, "button": "left" } } ] } }
<- { "return": {} }
```


2. Press ctrl-alt-del.

```
-> { "execute": "input-send-event",
      "arguments": { "events": [
        { "type": "key", "data" : { "down": true,
                                   "key": {"type": "qcode", "data": "ctrl" } } },
        { "type": "key", "data" : { "down": true,
                                   "key": {"type": "qcode", "data": "alt" } } },
        { "type": "key", "data" : { "down": true,
                                   "key": {"type": "qcode", "data": "delete" } } } ] } }
<- { "return": {} }
```

3. Move mouse pointer to absolute coordinates (20000, 400).

```
-> { "execute": "input-send-event" ,
      "arguments": { "events": [
        { "type": "abs", "data" : { "axis": "x", "value" : 20000 } },
        { "type": "abs", "data" : { "axis": "y", "value" : 400 } } ] } }
<- { "return": {} }
```

1.14 Migration

MigrationStats

[Object]

Detailed migration status.

Members:

transferred: int

amount of bytes already transferred to the target VM

remaining: int

amount of bytes remaining to be transferred to the target VM

total: int

total amount of bytes involved in the migration process

duplicate: int

number of duplicate (zero) pages (since 1.2)

skipped: int

number of skipped zero pages (since 1.5)

normal: int

number of normal pages (since 1.2)

normal-bytes: int

number of normal bytes sent (since 1.2)

dirty-pages-rate: int

number of pages dirtied by second by the guest (since 1.3)

mbps: number
throughput in megabits/sec. (since 1.6)

dirty-sync-count: int
number of times that dirty ram was synchronized (since 2.1)

postcopy-requests: int
The number of page requests received from the destination (since 2.7)

page-size: int
The number of bytes per page for the various page-based statistics (since 2.10)

Since: 0.14.0

XBZRLECacheStats [Object]

Detailed XBZRLE migration cache statistics

Members:

cache-size: int
XBZRLE cache size

bytes: int
amount of bytes already transferred to the target VM

pages: int
amount of pages transferred to the target VM

cache-miss: int
number of cache miss

cache-miss-rate: number
rate of cache miss (since 2.1)

overflow: int
number of overflows

Since: 1.2

MigrationStatus [Enum]

An enumeration of migration status.

Values:

none no migration has ever happened.

setup migration process has been initiated.

cancelling in the process of cancelling migration.

cancelled cancelling migration is finished.

active in the process of doing migration.

postcopy-active like active, but now in postcopy mode. (since 2.5)

<code>completed</code>	migration is finished.
<code>failed</code>	some error occurred during migration process.
<code>colo</code>	VM is in the process of fault tolerance, VM can not get into this state unless colo capability is enabled for migration. (since 2.8)

Since: 2.3

MigrationInfo [Object]

Information about current migration process.

Members:

<code>status: MigrationStatus</code> (optional)	<code>MigrationStatus</code> describing the current migration status. If this field is not returned, no migration process has been initiated
<code>ram: MigrationStats</code> (optional)	<code>MigrationStats</code> containing detailed migration status, only returned if status is 'active' or 'completed'(since 1.2)
<code>disk: MigrationStats</code> (optional)	<code>MigrationStats</code> containing detailed disk migration status, only returned if status is 'active' and it is a block migration
<code>xbzrle-cache: XBZRLECacheStats</code> (optional)	<code>XBZRLECacheStats</code> containing detailed XBZRLE migration statistics, only returned if XBZRLE feature is on and status is 'active' or 'completed' (since 1.2)
<code>total-time: int</code> (optional)	total amount of milliseconds since migration started. If migration has ended, it returns the total migration time. (since 1.2)
<code>downtime: int</code> (optional)	only present when migration finishes correctly total downtime in milliseconds for the guest. (since 1.3)
<code>expected-downtime: int</code> (optional)	only present while migration is active expected downtime in milliseconds for the guest in last walk of the dirty bitmap. (since 1.3)
<code>setup-time: int</code> (optional)	amount of setup time in milliseconds <i>before</i> the iterations begin but <i>after</i> the QMP command is issued. This is designed to provide an accounting of any activities (such as RDMA pinning) which may be expensive, but do not actually occur during the iterative migration rounds themselves. (since 1.6)
<code>cpu-throttle-percentage: int</code> (optional)	percentage of time guest cpus are being throttled during auto-converge. This is only present when auto-converge has started throttling guest cpus. (Since 2.7)

error-desc: string (optional)
 the human readable error description string, when **status** is 'failed'.
 Clients should not attempt to parse the error strings. (Since 2.7)

Since: 0.14.0

query-migrate [Command]

Returns information about current migration process. If migration is active there will be another json-object with RAM migration status and if block migration is active another one with block migration status.

Returns: MigrationInfo

Since: 0.14.0

Example:

1. Before the first migration

```
-> { "execute": "query-migrate" }
<- { "return": {} }
```

2. Migration is done and has succeeded

```
-> { "execute": "query-migrate" }
<- { "return": {
  "status": "completed",
  "ram":{
    "transferred":123,
    "remaining":123,
    "total":246,
    "total-time":12345,
    "setup-time":12345,
    "downtime":12345,
    "duplicate":123,
    "normal":123,
    "normal-bytes":123456,
    "dirty-sync-count":15
  }
}
```

3. Migration is done and has failed

```
-> { "execute": "query-migrate" }
<- { "return": { "status": "failed" } }
```

4. Migration is being performed and is not a block migration:

```
-> { "execute": "query-migrate" }
<- {
```

```

    "return":{
      "status":"active",
      "ram":{
        "transferred":123,
        "remaining":123,
        "total":246,
        "total-time":12345,
        "setup-time":12345,
        "expected-downtime":12345,
        "duplicate":123,
        "normal":123,
        "normal-bytes":123456,
        "dirty-sync-count":15
      }
    }
  }
}

```

5. Migration is being performed and is a block migration:

```

-> { "execute": "query-migrate" }
<- {
  "return":{
    "status":"active",
    "ram":{
      "total":1057024,
      "remaining":1053304,
      "transferred":3720,
      "total-time":12345,
      "setup-time":12345,
      "expected-downtime":12345,
      "duplicate":123,
      "normal":123,
      "normal-bytes":123456,
      "dirty-sync-count":15
    },
    "disk":{
      "total":20971520,
      "remaining":20880384,
      "transferred":91136
    }
  }
}

```

6. Migration is being performed and XBZRLE is active:

```

-> { "execute": "query-migrate" }
<- {

```

```

"return":{
  "status":"active",
  "capabilities" : [ { "capability": "xbzrle", "state" : true } ],
  "ram":{
    "total":1057024,
    "remaining":1053304,
    "transferred":3720,
    "total-time":12345,
    "setup-time":12345,
    "expected-downtime":12345,
    "duplicate":10,
    "normal":3333,
    "normal-bytes":3412992,
    "dirty-sync-count":15
  },
  "xbzrle-cache":{
    "cache-size":67108864,
    "bytes":20971520,
    "pages":2444343,
    "cache-miss":2244,
    "cache-miss-rate":0.123,
    "overflow":34434
  }
}
}

```

MigrationCapability [Enum]

Migration capabilities enumeration

Values:

xbzrle Migration supports xbzrle (Xor Based Zero Run Length Encoding). This feature allows us to minimize migration traffic for certain work loads, by sending compressed difference of the pages

rdma-pin-all Controls whether or not the entire VM memory footprint is `mlock()`'d on demand or all at once. Refer to `docs/rdma.txt` for usage. Disabled by default. (since 2.0)

zero-blocks During storage migration encode blocks of zeroes efficiently. This essentially saves 1MB of zeroes per block on the wire. Enabling requires source and target VM to support this feature. To enable it is sufficient to enable the capability on the source VM. The feature is disabled by default. (since 1.6)

compress Use multiple compression threads to accelerate live migration. This feature can help to reduce the migration traffic, by sending compressed pages. Please note that if `compress` and `xbzrle` are both on, `compress`

only takes effect in the ram bulk stage, after that, it will be disabled and only xbzrle takes effect, this can help to minimize migration traffic. The feature is disabled by default. (since 2.4)

events generate events for each migration state change (since 2.4)

auto-converge

If enabled, QEMU will automatically throttle down the guest to speed up convergence of RAM migration. (since 1.6)

postcopy-ram

Start executing on the migration target before all of RAM has been migrated, pulling the remaining pages along as needed. NOTE: If the migration fails during postcopy the VM will fail. (since 2.6)

x-colo

If enabled, migration will never end, and the state of the VM on the primary side will be migrated continuously to the VM on secondary side, this process is called COarse-Grain LOck Stepping (COLO) for Non-stop Service. (since 2.8)

release-ram

if enabled, qemu will free the migrated ram pages on the source during postcopy-ram migration. (since 2.9)

block

If enabled, QEMU will also migrate the contents of all block devices. Default is disabled. A possible alternative uses mirror jobs to a builtin NBD server on the destination, which offers more flexibility. (Since 2.10)

return-path

If enabled, migration will use the return path even for precopy. (since 2.10)

Since: 1.2

MigrationCapabilityStatus

[Object]

Migration capability information

Members:

capability: MigrationCapability
capability enum

state: boolean
capability state bool

Since: 1.2

migrate-set-capabilities

[Command]

Enable/Disable the following migration capabilities (like xbzrle)

Arguments:

capabilities: array of MigrationCapabilityStatus
json array of capability modifications to make

Since: 1.2

Example:

```
-> { "execute": "migrate-set-capabilities" , "arguments":
      { "capabilities": [ { "capability": "xbzrle", "state": true } ] } }
```

query-migrate-capabilities [Command]

Returns information about the current migration capabilities status

Returns: MigrationCapabilitiesStatus

Since: 1.2

Example:

```
-> { "execute": "query-migrate-capabilities" }
<- { "return": [
      {"state": false, "capability": "xbzrle"},
      {"state": false, "capability": "rdma-pin-all"},
      {"state": false, "capability": "auto-converge"},
      {"state": false, "capability": "zero-blocks"},
      {"state": false, "capability": "compress"},
      {"state": true, "capability": "events"},
      {"state": false, "capability": "postcopy-ram"},
      {"state": false, "capability": "x-colo"}
    ]}
```

MigrationParameter [Enum]

Migration parameters enumeration

Values:

compress-level

Set the compression level to be used in live migration, the compression level is an integer between 0 and 9, where 0 means no compression, 1 means the best compression speed, and 9 means best compression ratio which will consume more CPU.

compress-threads

Set compression thread count to be used in live migration, the compression thread count is an integer between 1 and 255.

decompress-threads

Set decompression thread count to be used in live migration, the decompression thread count is an integer between 1 and 255. Usually, decompression is at least 4 times as fast as compression, so set the decompress-threads to the number about 1/4 of compress-threads is adequate.

cpu-throttle-initial

Initial percentage of time guest cpus are throttled when migration auto-converge is activated. The default value is 20. (Since 2.7)

cpu-throttle-increment

throttle percentage increase each time auto-converge detects that migration is not making progress. The default value is 10. (Since 2.7)

tls-creds

ID of the 'tls-creds' object that provides credentials for establishing a TLS connection over the migration data channel. On the outgoing side of the migration, the credentials must be for a 'client' endpoint, while for the incoming side the credentials must be for a 'server' endpoint. Setting this will enable TLS for all migrations. The default is unset, resulting in unsecured migration at the QEMU level. (Since 2.7)

tls-hostname

hostname of the target host for the migration. This is required when using x509 based TLS credentials and the migration URI does not already include a hostname. For example if using fd: or exec: based migration, the hostname must be provided so that the server's x509 certificate identity can be validated. (Since 2.7)

max-bandwidth

to set maximum speed for migration. maximum speed in bytes per second. (Since 2.8)

downtime-limit

set maximum tolerated downtime for migration. maximum downtime in milliseconds (Since 2.8)

x-checkpoint-delay

The delay time (in ms) between two COLO checkpoints in periodic mode. (Since 2.8)

block-incremental

Affects how much storage is migrated when the block migration capability is enabled. When false, the entire storage backing chain is migrated into a flattened image at the destination; when true, only the active qcow2 layer is migrated and the destination must already have access to the same backing chain as was used on the source. (since 2.10)

Since: 2.4

MigrateSetParameters

[Object]

Members:

compress-level: int (optional)
compression level

compress-threads: int (optional)
compression thread count

decompress-threads: int (optional)
decompression thread count

cpu-throttle-initial: int (optional)
Initial percentage of time guest cpus are throttled when migration auto-converge is activated. The default value is 20. (Since 2.7)

cpu-throttle-increment: int (optional)
throttle percentage increase each time auto-converge detects that migration is not making progress. The default value is 10. (Since 2.7)

tls-creds: `StrOrNull` (optional)

ID of the 'tls-creds' object that provides credentials for establishing a TLS connection over the migration data channel. On the outgoing side of the migration, the credentials must be for a 'client' endpoint, while for the incoming side the credentials must be for a 'server' endpoint. Setting this to a non-empty string enables TLS for all migrations. An empty string means that QEMU will use plain text mode for migration, rather than TLS (Since 2.9) Previously (since 2.7), this was reported by omitting `tls-creds` instead.

tls-hostname: `StrOrNull` (optional)

hostname of the target host for the migration. This is required when using x509 based TLS credentials and the migration URI does not already include a hostname. For example if using `fd:` or `exec:` based migration, the hostname must be provided so that the server's x509 certificate identity can be validated. (Since 2.7) An empty string means that QEMU will use the hostname associated with the migration URI, if any. (Since 2.9) Previously (since 2.7), this was reported by omitting `tls-hostname` instead.

max-bandwidth: `int` (optional)

to set maximum speed for migration. maximum speed in bytes per second. (Since 2.8)

downtime-limit: `int` (optional)

set maximum tolerated downtime for migration. maximum downtime in milliseconds (Since 2.8)

x-checkpoint-delay: `int` (optional)

the delay time between two COLO checkpoints. (Since 2.8)

block-incremental: `boolean` (optional)

Affects how much storage is migrated when the block migration capability is enabled. When false, the entire storage backing chain is migrated into a flattened image at the destination; when true, only the active qcow2 layer is migrated and the destination must already have access to the same backing chain as was used on the source. (since 2.10)

Since: 2.4

migrate-set-parameters

[Command]

Set various migration parameters.

Arguments: the members of `MigrateSetParameters`

Since: 2.4

Example:

```
-> { "execute": "migrate-set-parameters" ,
      "arguments": { "compress-level": 1 } }
```

MigrationParameters

[Object]

The optional members aren't actually optional.

Members:

- compress-level:** int (optional)
compression level
- compress-threads:** int (optional)
compression thread count
- decompress-threads:** int (optional)
decompression thread count
- cpu-throttle-initial:** int (optional)
Initial percentage of time guest cpus are throttled when migration auto-converge is activated. (Since 2.7)
- cpu-throttle-increment:** int (optional)
throttle percentage increase each time auto-converge detects that migration is not making progress. (Since 2.7)
- tls-creds:** string (optional)
ID of the 'tls-creds' object that provides credentials for establishing a TLS connection over the migration data channel. On the outgoing side of the migration, the credentials must be for a 'client' endpoint, while for the incoming side the credentials must be for a 'server' endpoint. An empty string means that QEMU will use plain text mode for migration, rather than TLS (Since 2.7) Note: 2.8 reports this by omitting tls-creds instead.
- tls-hostname:** string (optional)
hostname of the target host for the migration. This is required when using x509 based TLS credentials and the migration URI does not already include a hostname. For example if using fd: or exec: based migration, the hostname must be provided so that the server's x509 certificate identity can be validated. (Since 2.7) An empty string means that QEMU will use the hostname associated with the migration URI, if any. (Since 2.9) Note: 2.8 reports this by omitting tls-hostname instead.
- max-bandwidth:** int (optional)
to set maximum speed for migration. maximum speed in bytes per second. (Since 2.8)
- downtime-limit:** int (optional)
set maximum tolerated downtime for migration. maximum downtime in milliseconds (Since 2.8)
- x-checkpoint-delay:** int (optional)
the delay time between two COLO checkpoints. (Since 2.8)
- block-incremental:** boolean (optional)
Affects how much storage is migrated when the block migration capability is enabled. When false, the entire storage backing chain is migrated into a flattened image at the destination; when true, only the active qcow2 layer is migrated and the destination must already have access to the same backing chain as was used on the source. (since 2.10)

Since: 2.4

`query-migrate-parameters` [Command]

Returns information about the current migration parameters

Returns: MigrationParameters

Since: 2.4

Example:

```
-> { "execute": "query-migrate-parameters" }
<- { "return": {
    "decompress-threads": 2,
    "cpu-throttle-increment": 10,
    "compress-threads": 8,
    "compress-level": 1,
    "cpu-throttle-initial": 20,
    "max-bandwidth": 33554432,
    "downtime-limit": 300
  }
}
```

`client_migrate_info` [Command]

Set migration information for remote display. This makes the server ask the client to automatically reconnect using the new parameters once migration finished successfully. Only implemented for SPICE.

Arguments:

```
protocol: string
    must be "spice"

hostname: string
    migration target hostname

port: int (optional)
    spice tcp port for plaintext channels

tls-port: int (optional)
    spice tcp port for tls-secured channels

cert-subject: string (optional)
    server certificate subject
```

Since: 0.14.0

Example:

```
-> { "execute": "client_migrate_info",
    "arguments": { "protocol": "spice",
                  "hostname": "virt42.lab.kraxel.org",
                  "port": 1234 } }
<- { "return": {} }
```

`migrate-start-postcopy` [Command]

Followup to a migration command to switch the migration to postcopy mode. The `postcopy-ram` capability must be set before the original migration command.

Since: 2.5

Example:

```
-> { "execute": "migrate-start-postcopy" }
<- { "return": {} }
```

`MIGRATION` [Event]

Emitted when a migration event happens

Arguments:

`status`: `MigrationStatus`
 `MigrationStatus` describing the current migration status.

Since: 2.4

Example:

```
<- { "timestamp": { "seconds": 1432121972, "microseconds": 744001 },
      "event": "MIGRATION",
      "data": { "status": "completed" } }
```

`MIGRATION_PASS` [Event]

Emitted from the source side of a migration at the start of each pass (when it syncs the dirty bitmap)

Arguments:

`pass`: `int` An incrementing count (starting at 1 on the first pass)

Since: 2.6

Example:

```
{ "timestamp": { "seconds": 1449669631, "microseconds": 239225 },
  "event": "MIGRATION_PASS", "data": { "pass": 2 } }
```

`COLOMessage` [Enum]

The message transmission between Primary side and Secondary side.

Values:

`checkpoint-ready`
 Secondary VM (SVM) is ready for checkpointing

`checkpoint-request`
 Primary VM (PVM) tells SVM to prepare for checkpointing

`checkpoint-reply`
 SVM gets PVM's checkpoint request

`vmstate-send`
 VM's state will be sent by PVM.

`vmstate-size`
 The total size of VMstate.

`vmstate-received`
VM's state has been received by SVM.

`vmstate-loaded`
VM's state has been loaded by SVM.

Since: 2.8

`COLOMode` [Enum]

The colo mode

Values:

`unknown` unknown mode

`primary` master side

`secondary`
slave side

Since: 2.8

`FailoverStatus` [Enum]

An enumeration of COLO failover status

Values:

`none` no failover has ever happened

`require` got failover requirement but not handled

`active` in the process of doing failover

`completed`
finish the process of failover

`relaunch` restart the failover process, from 'none' -> 'completed' (Since 2.9)

Since: 2.8

`x-colo-lost-heartbeat` [Command]

Tell qemu that heartbeat is lost, request it to do takeover procedures. If this command is sent to the PVM, the Primary side will exit COLO mode. If sent to the Secondary, the Secondary side will run failover work, then takes over server operation to become the service VM.

Since: 2.8

Example:

```
-> { "execute": "x-colo-lost-heartbeat" }
<- { "return": {} }
```

`migrate_cancel` [Command]

Cancel the current executing migration process.

Returns: nothing on success

Notes: This command succeeds even if there is no migration process running.

Since: 0.14.0

Example:

```
-> { "execute": "migrate_cancel" }
<- { "return": {} }
```

`migrate_set_downtime` [Command]

Set maximum tolerated downtime for migration.

Arguments:

`value: number`
maximum downtime in seconds

Returns: nothing on success

Notes: This command is deprecated in favor of 'migrate-set-parameters'

Since: 0.14.0

Example:

```
-> { "execute": "migrate_set_downtime", "arguments": { "value": 0.1 } }
<- { "return": {} }
```

`migrate_set_speed` [Command]

Set maximum speed for migration.

Arguments:

`value: int`
maximum speed in bytes per second.

Returns: nothing on success

Notes: This command is deprecated in favor of 'migrate-set-parameters'

Since: 0.14.0

Example:

```
-> { "execute": "migrate_set_speed", "arguments": { "value": 1024 } }
<- { "return": {} }
```

`migrate-set-cache-size` [Command]

Set cache size to be used by XBZRLE migration

Arguments:

`value: int`
cache size in bytes

The size will be rounded down to the nearest power of 2. The cache size can be modified before and during ongoing migration

Returns: nothing on success

Since: 1.2

Example:

```
-> { "execute": "migrate-set-cache-size",
      "arguments": { "value": 536870912 } }
<- { "return": {} }
```

`query-migrate-cache-size` [Command]

Query migration XBZRLE cache size

Returns: XBZRLE cache size in bytes

Since: 1.2

Example:

```
-> { "execute": "query-migrate-cache-size" }
<- { "return": 67108864 }
```

migrate [Command]

Migrates the current running guest to another Virtual Machine.

Arguments:

uri: string

the Uniform Resource Identifier of the destination VM

blk: boolean (optional)

do block migration (full disk copy)

inc: boolean (optional)

incremental disk copy migration

detach: boolean (optional)

this argument exists only for compatibility reasons and is ignored by QEMU

Returns: nothing on success

Since: 0.14.0

Notes:

1. The 'query-migrate' command should be used to check migration's progress and final result (this information is provided by the 'status' member)
2. All boolean arguments default to false
3. The user Monitor's "detach" argument is invalid in QMP and should not be used

Example:

```
-> { "execute": "migrate", "arguments": { "uri": "tcp:0:4446" } }
<- { "return": {} }
```

migrate-incoming [Command]

Start an incoming migration, the qemu must have been started with -incoming defer

Arguments:

uri: string

The Uniform Resource Identifier identifying the source or address to listen on

Returns: nothing on success

Since: 2.3

Notes:

1. It's a bad idea to use a string for the uri, but it needs to stay compatible with -incoming and the format of the uri is already exposed above libvirt.
2. QEMU must be started with -incoming defer to allow migrate-incoming to be used.

3. The uri format is the same as for -incoming

Example:

```
-> { "execute": "migrate-incoming",
      "arguments": { "uri": "tcp::4446" } }
<- { "return": {} }
```

xen-save-devices-state [Command]

Save the state of all devices to file. The RAM and the block devices of the VM are not saved by this command.

Arguments:

filename: string
the file to save the state of the devices to as binary data. See xen-save-devices-state.txt for a description of the binary format.

Returns: Nothing on success

Since: 1.1

Example:

```
-> { "execute": "xen-save-devices-state",
      "arguments": { "filename": "/tmp/save" } }
<- { "return": {} }
```

xen-set-replication [Command]

Enable or disable replication.

Arguments:

enable: boolean
true to enable, false to disable.

primary: boolean
true for primary or false for secondary.

failover: boolean (optional)
true to do failover, false to stop. but cannot be specified if 'enable' is true. default value is false.

Returns: nothing.

Example:

```
-> { "execute": "xen-set-replication",
      "arguments": {"enable": true, "primary": false} }
<- { "return": {} }
```

Since: 2.9

ReplicationStatus [Object]

The result format for 'query-xen-replication-status'.

Members:

error: boolean
true if an error happened, false if replication is normal.

desc: `string` (optional)
the human readable error description string, when `error` is 'true'.

Since: 2.9

`query-xen-replication-status` [Command]

Query replication status while the vm is running.

Returns: A `ReplicationResult` object showing the status.

Example:

```
-> { "execute": "query-xen-replication-status" }
<- { "return": { "error": false } }
```

Since: 2.9

`xen-colo-do-checkpoint` [Command]

Xen uses this command to notify replication to trigger a checkpoint.

Returns: nothing.

Example:

```
-> { "execute": "xen-colo-do-checkpoint" }
<- { "return": {} }
```

Since: 2.9

1.15 Transactions

`Abort` [Object]

This action can be used to test transaction failure.

Since: 1.6

`ActionCompletionMode` [Enum]

An enumeration of Transactional completion modes.

Values:

`individual`

Do not attempt to cancel any other Actions if any Actions fail after the Transaction request succeeds. All Actions that can complete successfully will do so without waiting on others. This is the default.

`grouped`

If any Action fails after the Transaction succeeds, cancel all Actions. Actions do not complete until all Actions are ready to complete. May be rejected by Actions that do not support this completion mode.

Since: 2.5

`TransactionAction` [Object]

A discriminated record of operations that can be performed with `transaction`. Action type can be:

- `abort`: since 1.6
- `block-dirty-bitmap-add`: since 2.5

- `block-dirty-bitmap-clear`: since 2.5
- `blockdev-backup`: since 2.3
- `blockdev-snapshot`: since 2.5
- `blockdev-snapshot-internal-sync`: since 1.7
- `blockdev-snapshot-sync`: since 1.1
- `drive-backup`: since 1.6

Members:

`type` One of "abort", "block-dirty-bitmap-add", "block-dirty-bitmap-clear", "blockdev-backup", "blockdev-snapshot", "blockdev-snapshot-internal-sync", "blockdev-snapshot-sync", "drive-backup"

`data`: Abort when type is "abort"

`data`: BlockDirtyBitmapAdd when type is "block-dirty-bitmap-add"

`data`: BlockDirtyBitmap when type is "block-dirty-bitmap-clear"

`data`: BlockdevBackup when type is "blockdev-backup"

`data`: BlockdevSnapshot when type is "blockdev-snapshot"

`data`: BlockdevSnapshotInternal when type is "blockdev-snapshot-internal-sync"

`data`: BlockdevSnapshotSync when type is "blockdev-snapshot-sync"

`data`: DriveBackup when type is "drive-backup"

Since: 1.1

`TransactionProperties` [Object]

Optional arguments to modify the behavior of a Transaction.

Members:

`completion-mode`: `ActionCompletionMode` (optional)

Controls how jobs launched asynchronously by Actions will complete or fail as a group. See `ActionCompletionMode` for details.

Since: 2.5

`transaction` [Command]

Executes a number of transactionable QMP commands atomically. If any operation fails, then the entire set of actions will be abandoned and the appropriate error returned.

For external snapshots, the dictionary contains the device, the file to use for the new snapshot, and the format. The default format, if not specified, is `qcow2`.

Each new snapshot defaults to being created by QEMU (wiping any contents if the file already exists), but it is also possible to reuse an externally-created file. In the latter case, you should ensure that the new image file has the same contents as the current one; QEMU cannot perform any meaningful check. Typically this is achieved by using the current image file as the backing file for the new image.

On failure, the original disks pre-snapshot attempt will be used.

For internal snapshots, the dictionary contains the device and the snapshot's name. If an internal snapshot matching name already exists, the request will be rejected. Only some image formats support it, for example, `qcow2`, `rbd`, and `sheepdog`.

On failure, qemu will try delete the newly created internal snapshot in the transaction. When an I/O error occurs during deletion, the user needs to fix it later with `qemu-img` or other command.

Arguments:

actions: array of `TransactionAction`

List of `TransactionAction`; information needed for the respective operations.

properties: `TransactionProperties` (optional)

structure of additional options to control the execution of the transaction. See `TransactionProperties` for additional detail.

Returns: nothing on success

Errors depend on the operations of the transaction

Note: The transaction aborts on the first failure. Therefore, there will be information on only one failed operation returned in an error condition, and subsequent actions will not have been attempted.

Since: 1.1

Example:

```
-> { "execute": "transaction",
    "arguments": { "actions": [
        { "type": "blockdev-snapshot-sync", "data" : { "device": "ide-hd0",
            "snapshot-file": "/some/place/my-image",
            "format": "qcow2" } },
        { "type": "blockdev-snapshot-sync", "data" : { "node-name": "myfile",
            "snapshot-file": "/some/place/my-image2",
            "snapshot-node-name": "node3432",
            "mode": "existing",
            "format": "qcow2" } },
        { "type": "blockdev-snapshot-sync", "data" : { "device": "ide-hd1",
            "snapshot-file": "/some/place/my-image2",
            "mode": "existing",
            "format": "qcow2" } },
        { "type": "blockdev-snapshot-internal-sync", "data" : {
            "device": "ide-hd2",
            "name": "snapshot0" } } ] } }

<- { "return": {} }
```

1.16 Tracing

`TraceEventState`

[Enum]

State of a tracing event.

Values:

`unavailable`

The event is statically disabled.

`disabled` The event is dynamically disabled.

`enabled` The event is dynamically enabled.

Since: 2.2

`TraceEventInfo` [Object]

Information of a tracing event.

Members:

`name: string`

Event name.

`state: TraceEventState`

Tracing state.

`vcpu: boolean`

Whether this is a per-vCPU event (since 2.7).

An event is per-vCPU if it has the "vcpu" property in the "trace-events" files.

Since: 2.2

`trace-event-get-state` [Command]

Query the state of events.

Arguments:

`name: string`

Event name pattern (case-sensitive glob).

`vcpu: int` (optional)

The vCPU to query (any by default; since 2.7).

Returns: a list of `TraceEventInfo` for the matching events

An event is returned if:

- its name matches the `name` pattern, and
- if `vcpu` is given, the event has the "vcpu" property.

Therefore, if `vcpu` is given, the operation will only match per-vCPU events, returning their state on the specified vCPU. Special case: if `name` is an exact match, `vcpu` is given and the event does not have the "vcpu" property, an error is returned.

Since: 2.2

Example:

```
-> { "execute": "trace-event-get-state",
      "arguments": { "name": "qemu_memalign" } }
<- { "return": [ { "name": "qemu_memalign", "state": "disabled" } ] }
```

`trace-event-set-state` [Command]

Set the dynamic tracing state of events.

Arguments:

`name: string`

Event name pattern (case-sensitive glob).

enable: boolean

Whether to enable tracing.

ignore-unavailable: boolean (optional)

Do not match unavailable events with **name**.

vcpu: int (optional)

The vCPU to act upon (all by default; since 2.7).

An event's state is modified if:

- its name matches the **name** pattern, and
- if **vcpu** is given, the event has the "vcpu" property.

Therefore, if **vcpu** is given, the operation will only match per-vCPU events, setting their state on the specified vCPU. Special case: if **name** is an exact match, **vcpu** is given and the event does not have the "vcpu" property, an error is returned.

Since: 2.2

Example:

```
-> { "execute": "trace-event-set-state",
      "arguments": { "name": "qemu_memalign", "enable": "true" } }
<- { "return": {} }
```

1.17 QMP introspection

query-qmp-schema

[Command]

Command `query-qmp-schema` exposes the QMP wire ABI as an array of `SchemaInfo`. This lets QMP clients figure out what commands and events are available in this QEMU, and their parameters and results.

However, the `SchemaInfo` can't reflect all the rules and restrictions that apply to QMP. It's interface introspection (figuring out what's there), not interface specification. The specification is in the QAPI schema.

Furthermore, while we strive to keep the QMP wire format backwards-compatible across qemu versions, the introspection output is not guaranteed to have the same stability. For example, one version of qemu may list an object member as an optional non-variant, while another lists the same member only through the object's variants; or the type of a member may change from a generic string into a specific enum or from one specific type into an alternate that includes the original type alongside something else.

Returns: array of `SchemaInfo`, where each element describes an entity in the ABI: command, event, type, ...

The order of the various `SchemaInfo` is unspecified; however, all names are guaranteed to be unique (no name will be duplicated with different meta-types).

Note: the QAPI schema is also used to help define **internal** interfaces, by defining QAPI types. These are not part of the QMP wire ABI, and therefore not returned by this command.

Since: 2.5

SchemaMetaType [Enum]

This is a `SchemaInfo`'s meta type, i.e. the kind of entity it describes.

Values:

`builtin` a predefined type such as 'int' or 'bool'.

`enum` an enumeration type

`array` an array type

`object` an object type (struct or union)

`alternate`
an alternate type

`command` a QMP command

`event` a QMP event

Since: 2.5

SchemaInfo [Object]**Members:**

`name: string`

the entity's name, inherited from `base`. The `SchemaInfo` is always referenced by this name. Commands and events have the name defined in the QAPI schema. Unlike command and event names, type names are not part of the wire ABI. Consequently, type names are meaningless strings here, although they are still guaranteed unique regardless of `meta-type`.

`meta-type: SchemaMetaType`

the entity's meta type, inherited from `base`.

The members of `SchemaInfoBuiltin` when `meta-type` is "builtin"

The members of `SchemaInfoEnum` when `meta-type` is "enum"

The members of `SchemaInfoArray` when `meta-type` is "array"

The members of `SchemaInfoObject` when `meta-type` is "object"

The members of `SchemaInfoAlternate` when `meta-type` is "alternate"

The members of `SchemaInfoCommand` when `meta-type` is "command"

The members of `SchemaInfoEvent` when `meta-type` is "event"

Additional members depend on the value of `meta-type`.

Since: 2.5

SchemaInfoBuiltin [Object]

Additional `SchemaInfo` members for meta-type 'builtin'.

Members:

`json-type: JSONType`

the JSON type used for this type on the wire.

Since: 2.5

JSONType [Enum]

The four primitive and two structured types according to RFC 7159 section 1, plus 'int' (split off 'number'), plus the obvious top type 'value'.

Values:

string Not documented

number Not documented

int Not documented

boolean Not documented

null Not documented

object Not documented

array Not documented

value Not documented

Since: 2.5

SchemaInfoEnum [Object]

Additional SchemaInfo members for meta-type 'enum'.

Members:

values: array of string
the enumeration type's values, in no particular order.

Values of this type are JSON string on the wire.

Since: 2.5

SchemaInfoArray [Object]

Additional SchemaInfo members for meta-type 'array'.

Members:

element-type: string
the array type's element type.

Values of this type are JSON array on the wire.

Since: 2.5

SchemaInfoObject [Object]

Additional SchemaInfo members for meta-type 'object'.

Members:

members: array of SchemaInfoObjectMember
the object type's (non-variant) members, in no particular order.

tag: string (optional)
the name of the member serving as type tag. An element of **members** with this name must exist.

variants: array of `SchemaInfoObjectVariant` (optional)
 variant members, i.e. additional members that depend on the type tag's value. Present exactly when `tag` is present. The variants are in no particular order, and may even differ from the order of the values of the enum type of the `tag`.

Values of this type are JSON object on the wire.

Since: 2.5

SchemaInfoObjectMember [Object]

An object member.

Members:

name: `string`
 the member's name, as defined in the QAPI schema.

type: `string`
 the name of the member's type.

default: `value` (optional)
 default when used as command parameter. If absent, the parameter is mandatory. If present, the value must be null. The parameter is optional, and behavior when it's missing is not specified here. Future extension: if present and non-null, the parameter is optional, and defaults to this value.

Since: 2.5

SchemaInfoObjectVariant [Object]

The variant members for a value of the type tag.

Members:

case: `string`
 a value of the type tag.

type: `string`
 the name of the object type that provides the variant members when the type tag has value `case`.

Since: 2.5

SchemaInfoAlternate [Object]

Additional `SchemaInfo` members for meta-type 'alternate'.

Members:

members: array of `SchemaInfoAlternateMember`
 the alternate type's members, in no particular order. The members' wire encoding is distinct, see `docs/devel/qapi-code-gen.txt` section `Alternate types`.

On the wire, this can be any of the members.

Since: 2.5

SchemaInfoAlternateMember [Object]

An alternate member.

Members:

type: `string`
the name of the member's type.

Since: 2.5

SchemaInfoCommand [Object]

Additional SchemaInfo members for meta-type 'command'.

Members:

arg-type: `string`
the name of the object type that provides the command's parameters.

ret-type: `string`
the name of the command's result type.

TODO: `success-response` (currently irrelevant, because it's QGA, not QMP)

Since: 2.5

SchemaInfoEvent [Object]

Additional SchemaInfo members for meta-type 'event'.

Members:

arg-type: `string`
the name of the object type that provides the event's parameters.

Since: 2.5

1.18 Miscellanea

qmp_capabilities [Command]

Enable QMP capabilities.

Arguments: None.

Example:

```
-> { "execute": "qmp_capabilities" }
<- { "return": {} }
```

Notes: This command is valid exactly when first connecting: it must be issued before any other command will be accepted, and will fail once the monitor is accepting other commands. (see qemu docs/interop/qmp-spec.txt)

Since: 0.13

VersionTriple [Object]

A three-part version number.

Members:

major: `int`
The major version number.

minor: int
The minor version number.

micro: int
The micro version number.

Since: 2.4

VersionInfo [Object]

A description of QEMU's version.

Members:

qemu: VersionTriple
The version of QEMU. By current convention, a micro version of 50 signifies a development branch. A micro version greater than or equal to 90 signifies a release candidate for the next minor version. A micro version of less than 50 signifies a stable release.

package: string
QEMU will always set this field to an empty string. Downstream versions of QEMU should set this to a non-empty string. The exact format depends on the downstream however it highly recommended that a unique name is used.

Since: 0.14.0

query-version [Command]

Returns the current version of QEMU.

Returns: A `VersionInfo` object describing the current version of QEMU.

Since: 0.14.0

Example:

```
-> { "execute": "query-version" }
<- {
  "return":{
    "qemu":{
      "major":0,
      "minor":11,
      "micro":5
    },
    "package":""
  }
}
```

CommandInfo [Object]

Information about a QMP command

Members:

name: string
The command name

Since: 0.14.0

query-commands [Command]

Return a list of supported QMP commands by this server

Returns: A list of `CommandInfo` for all supported commands

Since: 0.14.0

Example:

```
-> { "execute": "query-commands" }
<- {
  "return": [
    {
      "name": "query-balloon"
    },
    {
      "name": "system_powerdown"
    }
  ]
}
```

Note: This example has been shortened as the real response is too long.

LostTickPolicy [Enum]

Policy for handling lost ticks in timer devices.

Values:

- discard** throw away the missed tick(s) and continue with future injection normally. Guest time may be delayed, unless the OS has explicit handling of lost ticks
- delay** continue to deliver ticks at the normal rate. Guest time will be delayed due to the late tick
- merge** merge the missed tick(s) into one tick and inject. Guest time may be delayed, depending on how the OS reacts to the merging of ticks
- slew** deliver ticks at a higher rate to catch up with the missed tick. The guest time should not be delayed once catchup is complete.

Since: 2.0

add_client [Command]

Allow client connections for VNC, Spice and socket based character devices to be passed in to QEMU via `SCM_RIGHTS`.

Arguments:

protocol: string

protocol name. Valid names are "vnc", "spice" or the name of a character device (eg. from `-chardev id=XXXX`)

fdname: string

file descriptor name previously passed via 'getfd' command

skipauth: `boolean` (optional)
 whether to skip authentication. Only applies to "vnc" and "spice" protocols

tls: `boolean` (optional)
 whether to perform TLS. Only applies to the "spice" protocol

Returns: nothing on success.

Since: 0.14.0

Example:

```
-> { "execute": "add_client", "arguments": { "protocol": "vnc",
                                           "fdname": "myclient" } }
<- { "return": {} }
```

NameInfo [Object]

Guest name information.

Members:

name: `string` (optional)
 The name of the guest

Since: 0.14.0

query-name [Command]

Return the name information of a guest.

Returns: `NameInfo` of the guest

Since: 0.14.0

Example:

```
-> { "execute": "query-name" }
<- { "return": { "name": "qemu-name" } }
```

KvmInfo [Object]

Information about support for KVM acceleration

Members:

enabled: `boolean`
 true if KVM acceleration is active

present: `boolean`
 true if KVM acceleration is built into this executable

Since: 0.14.0

query-kvm [Command]

Returns information about KVM acceleration

Returns: `KvmInfo`

Since: 0.14.0

Example:

```
-> { "execute": "query-kvm" }
<- { "return": { "enabled": true, "present": true } }
```

UuidInfo [Object]

Guest UUID information (Universally Unique Identifier).

Members:

UUID: string
the UUID of the guest

Since: 0.14.0

Notes: If no UUID was specified for the guest, a null UUID is returned.

query-uuid [Command]

Query the guest UUID information.

Returns: The `UuidInfo` for the guest

Since: 0.14.0

Example:

```
-> { "execute": "query-uuid" }
<- { "return": { "UUID": "550e8400-e29b-41d4-a716-446655440000" } }
```

EventInfo [Object]

Information about a QMP event

Members:

name: string
The event name

Since: 1.2.0

query-events [Command]

Return a list of supported QMP events by this server

Returns: A list of `EventInfo` for all supported events

Since: 1.2.0

Example:

```
-> { "execute": "query-events" }
<- {
  "return": [
    {
      "name": "SHUTDOWN"
    },
    {
      "name": "RESET"
    }
  ]
}
```

Note: This example has been shortened as the real response is too long.

CpuInfoArch [Enum]

An enumeration of cpu types that enable additional information during `query-cpus`.

Values:

<code>x86</code>	Not documented
<code>sparc</code>	Not documented
<code>ppc</code>	Not documented
<code>mips</code>	Not documented
<code>tricore</code>	Not documented
<code>other</code>	Not documented

Since: 2.6

CpuInfo [Object]

Information about a virtual CPU

Members:

cpu: `int` the index of the virtual CPU

current: `boolean`
this only exists for backwards compatibility and should be ignored

halted: `boolean`
true if the virtual CPU is in the halt state. Halt usually refers to a processor specific low power mode.

qom_path: `string`
path to the CPU object in the QOM tree (since 2.4)

thread_id: `int`
ID of the underlying host thread

props: `CpuInstanceProperties` (optional)
properties describing to which node/socket/core/thread virtual CPU belongs to, provided if supported by board (since 2.10)

arch: `CpuInfoArch`
architecture of the cpu, which determines which additional fields will be listed (since 2.6)

The members of `CpuInfoX86` when `arch` is "x86"

The members of `CpuInfoSPARC` when `arch` is "sparc"

The members of `CpuInfoPPC` when `arch` is "ppc"

The members of `CpuInfoMIPS` when `arch` is "mips"

The members of `CpuInfoTricore` when `arch` is "tricore"

The members of `CpuInfoOther` when `arch` is "other"

Since: 0.14.0

Notes: `halted` is a transient state that changes frequently. By the time the data is sent to the client, the guest may no longer be halted.

- CpuInfoX86** [Object]
Additional information about a virtual i386 or x86_64 CPU
Members:
`pc: int` the 64-bit instruction pointer
Since: 2.6
- CpuInfoSPARC** [Object]
Additional information about a virtual SPARC CPU
Members:
`pc: int` the PC component of the instruction pointer
`npc: int` the NPC component of the instruction pointer
Since: 2.6
- CpuInfoPPC** [Object]
Additional information about a virtual PPC CPU
Members:
`nip: int` the instruction pointer
Since: 2.6
- CpuInfoMIPS** [Object]
Additional information about a virtual MIPS CPU
Members:
`PC: int` the instruction pointer
Since: 2.6
- CpuInfoTricore** [Object]
Additional information about a virtual Tricore CPU
Members:
`PC: int` the instruction pointer
Since: 2.6
- CpuInfoOther** [Object]
No additional information is available about the virtual CPU
Since: 2.6
- query-cpus** [Command]
Returns a list of information about each virtual CPU.
Returns: a list of `CpuInfo` for each virtual CPU
Since: 0.14.0
Example:

```
-> { "execute": "query-cpus" }  
<- { "return": [
```



```

    {
      "CPU":0,
      "current":true,
      "halted":false,
      "qom_path":"/machine/unattached/device[0]",
      "arch":"x86",
      "pc":3227107138,
      "thread_id":3134
    },
    {
      "CPU":1,
      "current":false,
      "halted":true,
      "qom_path":"/machine/unattached/device[2]",
      "arch":"x86",
      "pc":7108165,
      "thread_id":3135
    }
  ]
}

```

IOThreadInfo [Object]

Information about an iothread

Members:

id: string
the identifier of the iothread

thread-id: int
ID of the underlying host thread

poll-max-ns: int
maximum polling time in ns, 0 means polling is disabled (since 2.9)

poll-grow: int
how many ns will be added to polling time, 0 means that it's not configured (since 2.9)

poll-shrink: int
how many ns will be removed from polling time, 0 means that it's not configured (since 2.9)

Since: 2.0

query-iothreads [Command]

Returns a list of information about each iothread.

Note: this list excludes the QEMU main loop thread, which is not declared using the `-object iothread` command-line option. It is always the main thread of the process.

Returns: a list of **IOThreadInfo** for each iothread

Since: 2.0

Example:

```
-> { "execute": "query-iothreads" }
<- { "return": [
    {
      "id":"iothread0",
      "thread-id":3134
    },
    {
      "id":"iothread1",
      "thread-id":3135
    }
  ]
}
```

BalloonInfo

[Object]

Information about the guest balloon device.

Members:

actual: int

the number of bytes the balloon currently contains

Since: 0.14.0

query-balloon

[Command]

Return information about the balloon device.

Returns: BalloonInfo on success

If the balloon driver is enabled but not functional because the KVM kernel module cannot support it, KvmMissingCap

If no balloon device is present, DeviceNotActive

Since: 0.14.0

Example:

```
-> { "execute": "query-balloon" }
<- { "return": {
    "actual": 1073741824,
  }
}
```

BALLOON_CHANGE

[Event]

Emitted when the guest changes the actual BALLOON level. This value is equivalent to the **actual** field return by the 'query-balloon' command

Arguments:

actual: int

actual level of the guest memory balloon in bytes

Note: this event is rate-limited.

Since: 1.2

Example:

```
<- { "event": "BALLOON_CHANGE",
      "data": { "actual": 944766976 },
      "timestamp": { "seconds": 1267020223, "microseconds": 435656 } }
```

PciMemoryRange [Object]

A PCI device memory region

Members:

base: `int` the starting address (guest physical)

limit: `int`
the ending address (guest physical)

Since: 0.14.0

PciMemoryRegion [Object]

Information about a PCI device I/O region.

Members:

bar: `int` the index of the Base Address Register for this region

type: `string`
'io' if the region is a PIO region 'memory' if the region is a MMIO region

size: `int` memory size

prefetch: `boolean` (optional)
if type is 'memory', true if the memory is prefetchable

mem_type_64: `boolean` (optional)
if type is 'memory', true if the BAR is 64-bit

address: `int`
Not documented

Since: 0.14.0

PciBusInfo [Object]

Information about a bus of a PCI Bridge device

Members:

number: `int`
primary bus interface number. This should be the number of the bus the device resides on.

secondary: `int`
secondary bus interface number. This is the number of the main bus for the bridge

subordinate: `int`
This is the highest number bus that resides below the bridge.

io_range: `PciMemoryRange`
The PIO range for all devices on this bridge

`memory_range: PciMemoryRange`
 The MMIO range for all devices on this bridge

`prefetchable_range: PciMemoryRange`
 The range of prefetchable MMIO for all devices on this bridge

Since: 2.4

PciBridgeInfo [Object]

Information about a PCI Bridge device

Members:

`bus: PciBusInfo`
 information about the bus the device resides on

`devices: array of PciDeviceInfo (optional)`
 a list of `PciDeviceInfo` for each device on this bridge

Since: 0.14.0

PciDeviceClass [Object]

Information about the Class of a PCI device

Members:

`desc: string (optional)`
 a string description of the device's class

`class: int`
 the class code of the device

Since: 2.4

PciDeviceId [Object]

Information about the Id of a PCI device

Members:

`device: int`
 the PCI device id

`vendor: int`
 the PCI vendor id

Since: 2.4

PciDeviceInfo [Object]

Information about a PCI device

Members:

`bus: int` the bus number of the device

`slot: int` the slot the device is located in

`function: int`
 the function of the slot used by the device

class_info: `PciDeviceClass`
the class of the device

id: `PciDeviceId`
the PCI device id

irq: `int` (optional)
if an IRQ is assigned to the device, the IRQ number

qdev_id: `string`
the device name of the PCI device

pci_bridge: `PciBridgeInfo` (optional)
if the device is a PCI bridge, the bridge information

regions: array of `PciMemoryRegion`
a list of the PCI I/O regions associated with the device

Notes: the contents of `class_info.desc` are not stable and should only be treated as informational.

Since: 0.14.0

PciInfo [Object]

Information about a PCI bus

Members:

bus: `int` the bus index

devices: array of `PciDeviceInfo`
a list of devices on this bus

Since: 0.14.0

query-pci [Command]

Return information about the PCI bus topology of the guest.

Returns: a list of `PciInfo` for each PCI bus. Each bus is represented by a json-object, which has a key with a json-array of all PCI devices attached to it. Each device is represented by a json-object.

Since: 0.14.0

Example:

```
-> { "execute": "query-pci" }
<- { "return": [
  {
    "bus": 0,
    "devices": [
      {
        "bus": 0,
        "qdev_id": "",
        "slot": 0,
        "class_info": {
          "class": 1536,
```

```
        "desc": "Host bridge"
    },
    "id": {
        "device": 32902,
        "vendor": 4663
    },
    "function": 0,
    "regions": [
    ]
},
{
    "bus": 0,
    "qdev_id": "",
    "slot": 1,
    "class_info": {
        "class": 1537,
        "desc": "ISA bridge"
    },
    "id": {
        "device": 32902,
        "vendor": 28672
    },
    "function": 0,
    "regions": [
    ]
},
{
    "bus": 0,
    "qdev_id": "",
    "slot": 1,
    "class_info": {
        "class": 257,
        "desc": "IDE controller"
    },
    "id": {
        "device": 32902,
        "vendor": 28688
    },
    "function": 1,
    "regions": [
        {
            "bar": 4,
            "size": 16,
            "address": 49152,
            "type": "io"
        }
    ]
}
]
```

```
    },
    {
      "bus": 0,
      "qdev_id": "",
      "slot": 2,
      "class_info": {
        "class": 768,
        "desc": "VGA controller"
      },
      "id": {
        "device": 4115,
        "vendor": 184
      },
      "function": 0,
      "regions": [
        {
          "prefetch": true,
          "mem_type_64": false,
          "bar": 0,
          "size": 33554432,
          "address": 4026531840,
          "type": "memory"
        },
        {
          "prefetch": false,
          "mem_type_64": false,
          "bar": 1,
          "size": 4096,
          "address": 4060086272,
          "type": "memory"
        },
        {
          "prefetch": false,
          "mem_type_64": false,
          "bar": 6,
          "size": 65536,
          "address": -1,
          "type": "memory"
        }
      ]
    },
    {
      "bus": 0,
      "qdev_id": "",
      "irq": 11,
      "slot": 4,
      "class_info": {
```

```

        "class": 1280,
        "desc": "RAM controller"
    },
    "id": {
        "device": 6900,
        "vendor": 4098
    },
    "function": 0,
    "regions": [
        {
            "bar": 0,
            "size": 32,
            "address": 49280,
            "type": "io"
        }
    ]
}
]
}
]
}
}

```

Note: This example has been shortened as the real response is too long.

quit [Command]

This command will cause the QEMU process to exit gracefully. While every attempt is made to send the QMP response before terminating, this is not guaranteed. When using this interface, a premature EOF would not be unexpected.

Since: 0.14.0

Example:

```

-> { "execute": "quit" }
<- { "return": {} }

```

stop [Command]

Stop all guest VCPU execution.

Since: 0.14.0

Notes: This function will succeed even if the guest is already in the stopped state. In "inmigrate" state, it will ensure that the guest remains paused once migration finishes, as if the -S option was passed on the command line.

Example:

```

-> { "execute": "stop" }
<- { "return": {} }

```

system_reset [Command]

Performs a hard reset of a guest.

Since: 0.14.0

Example:

```
-> { "execute": "system_reset" }
<- { "return": {} }
```

system_powerdown [Command]

Requests that a guest perform a powerdown operation.

Since: 0.14.0

Notes: A guest may or may not respond to this command. This command returning does not indicate that a guest has accepted the request or that it has shut down. Many guests will respond to this command by prompting the user in some way.

Example:

```
-> { "execute": "system_powerdown" }
<- { "return": {} }
```

cpu [Command]

This command is a nop that is only provided for the purposes of compatibility.

Arguments:

index: int
Not documented

Since: 0.14.0

Notes: Do not use this command.

cpu-add [Command]

Adds CPU with specified ID

Arguments:

id: int ID of CPU to be created, valid values [0..max_cpus)

Returns: Nothing on success

Since: 1.5

Example:

```
-> { "execute": "cpu-add", "arguments": { "id": 2 } }
<- { "return": {} }
```

memsave [Command]

Save a portion of guest memory to a file.

Arguments:

val: int the virtual address of the guest to start from

size: int the size of memory region to save

filename: string
the file to save the memory to as binary data

cpu-index: int (optional)
the index of the virtual CPU to use for translating the virtual address (defaults to CPU 0)

Returns: Nothing on success

Since: 0.14.0

Notes: Errors were not reliably returned until 1.1

Example:

```
-> { "execute": "memsave",
      "arguments": { "val": 10,
                    "size": 100,
                    "filename": "/tmp/virtual-mem-dump" } }
<- { "return": {} }
```

`pmemsave` [Command]

Save a portion of guest physical memory to a file.

Arguments:

`val: int` the physical address of the guest to start from

`size: int` the size of memory region to save

`filename: string`

the file to save the memory to as binary data

Returns: Nothing on success

Since: 0.14.0

Notes: Errors were not reliably returned until 1.1

Example:

```
-> { "execute": "pmemsave",
      "arguments": { "val": 10,
                    "size": 100,
                    "filename": "/tmp/physical-mem-dump" } }
<- { "return": {} }
```

`cont` [Command]

Resume guest VCPU execution.

Since: 0.14.0

Returns: If successful, nothing

Notes: This command will succeed if the guest is currently running. It will also succeed if the guest is in the "inmigrate" state; in this case, the effect of the command is to make sure the guest starts once migration finishes, removing the effect of the `-S` command line option if it was passed.

Example:

```
-> { "execute": "cont" }
<- { "return": {} }
```

`system_wakeup` [Command]

Wakeup guest from suspend. Does nothing in case the guest isn't suspended.

Since: 1.1

Returns: nothing.

Example:

```
-> { "execute": "system_wakeup" }
<- { "return": {} }
```

inject-nmi [Command]

Injects a Non-Maskable Interrupt into the default CPU (x86/s390) or all CPUs (ppc64). The command fails when the guest doesn't support injecting.

Returns: If successful, nothing

Since: 0.14.0

Note: prior to 2.1, this command was only supported for x86 and s390 VMs

Example:

```
-> { "execute": "inject-nmi" }
<- { "return": {} }
```

balloon [Command]

Request the balloon driver to change its balloon size.

Arguments:

value: `int`

the target size of the balloon in bytes

Returns: Nothing on success If the balloon driver is enabled but not functional because the KVM kernel module cannot support it, `KvmMissingCap` If no balloon device is present, `DeviceNotActive`

Notes: This command just issues a request to the guest. When it returns, the balloon size may not have changed. A guest can change the balloon size independent of this command.

Since: 0.14.0

Example:

```
-> { "execute": "balloon", "arguments": { "value": 536870912 } }
<- { "return": {} }
```

human-monitor-command [Command]

Execute a command on the human monitor and return the output.

Arguments:

command-line: `string`

the command to execute in the human monitor

cpu-index: `int` (optional)

The CPU to use for commands that require an implicit CPU

Returns: the output of the command as a string

Since: 0.14.0

Notes: This command only exists as a stop-gap. Its use is highly discouraged. The semantics of this command are not guaranteed: this means that command names, arguments and responses can change or be removed at ANY time. Applications that rely on long term stability guarantees should NOT use this command.

Known limitations:

- This command is stateless, this means that commands that depend on state information (such as getfd) might not work
- Commands that prompt the user for data don't currently work

Example:

```
-> { "execute": "human-monitor-command",
      "arguments": { "command-line": "info kvm" } }
<- { "return": "kvm support: enabled\r\n" }
```

ObjectPropertyInfo [Object]

Members:

name: string

the name of the property

type: string

the type of the property. This will typically come in one of four forms:

- 1) A primitive type such as 'u8', 'u16', 'bool', 'str', or 'double'. These types are mapped to the appropriate JSON type.
- 2) A child type in the form 'child<subtype>' where subtype is a qdev device type name. Child properties create the composition tree.
- 3) A link type in the form 'link<subtype>' where subtype is a qdev device type name. Link properties form the device model graph.

Since: 1.2

qom-list [Command]

This command will list any properties of a object given a path in the object model.

Arguments:

path: string

the path within the object model. See `qom-get` for a description of this parameter.

Returns: a list of `ObjectPropertyInfo` that describe the properties of the object.

Since: 1.2

qom-get [Command]

This command will get a property from a object model path and return the value.

Arguments:

path: string

The path within the object model. There are two forms of supported paths—absolute and partial paths.

Absolute paths are derived from the root object and can follow child<> or link<> properties. Since they can follow link<> properties, they can be arbitrarily long. Absolute paths look like absolute filenames and are prefixed with a leading slash.

Partial paths look like relative filenames. They do not begin with a prefix. The matching rules for partial paths are subtle but designed to make specifying objects easy. At each level of the composition tree, the partial path is matched as an absolute path. The first match is not returned. At least two matches are searched for. A successful result is only returned if only one match is found. If more than one match is found, a flag is return to indicate that the match was ambiguous.

property: `string`

The property name to read

Returns: The property value. The type depends on the property type. `child<>` and `link<>` properties are returned as `#str` pathnames. All integer property types (`u8`, `u16`, etc) are returned as `#int`.

Since: 1.2

`qom-set`

[Command]

This command will set a property from a object model path.

Arguments:

path: `string`

see `qom-get` for a description of this parameter

property: `string`

the property name to set

value: `value`

a value who's type is appropriate for the property type. See `qom-get` for a description of type mapping.

Since: 1.2

`change`

[Command]

This command is multiple commands multiplexed together.

Arguments:

device: `string`

This is normally the name of a block device but it may also be `'vnc'`. when it's `'vnc'`, then sub command depends on `target`

target: `string`

If `device` is a block device, then this is the new filename. If `device` is `'vnc'`, then if the value `'password'` selects the `vnc change password` command. Otherwise, this specifies a new server URI address to listen to for VNC connections.

arg: `string` (optional)

If `device` is a block device, then this is an optional format to open the device with. If `device` is `'vnc'` and `target` is `'password'`, this is the new VNC password to set. See `change-vnc-password` for additional notes.

Returns: Nothing on success. If `device` is not a valid block device, `DeviceNotFound`

Notes: This interface is deprecated, and it is strongly recommended that you avoid using it. For changing block devices, use `blockdev-change-medium`; for changing VNC parameters, use `change-vnc-password`.

Since: 0.14.0

Example:

1. Change a removable medium

```
-> { "execute": "change",
      "arguments": { "device": "ide1-cd0",
                    "target": "/srv/images/Fedora-12-x86_64-DVD.iso" } }
<- { "return": {} }
```

2. Change VNC password

```
-> { "execute": "change",
      "arguments": { "device": "vnc", "target": "password",
                    "arg": "foobar1" } }
<- { "return": {} }
```

ObjectTypeInfo

[Object]

This structure describes a search result from `qom-list-types`

Members:

`name: string`

the type name found in the search

`abstract: boolean` (optional)

the type is abstract and can't be directly instantiated. Omitted if false.
(since 2.10)

`parent: string` (optional)

Name of parent type, if any (since 2.10)

Since: 1.1

qom-list-types

[Command]

This command will return a list of types given search parameters

Arguments:

`implements: string` (optional)

if specified, only return types that implement this type name

`abstract: boolean` (optional)

if true, include abstract types in the results

Returns: a list of `ObjectTypeInfo` or an empty list if no results are found

Since: 1.1

DevicePropertyInfo [Object]

Information about device properties.

Members:

name: string
the name of the property

type: string
the typename of the property

description: string (optional)
if specified, the description of the property. (since 2.2)

Since: 1.2

device-list-properties [Command]

List properties associated with a device.

Arguments:

typename: string
the type name of a device

Returns: a list of DevicePropertyInfo describing a devices properties

Since: 1.2

xen-set-global-dirty-log [Command]

Enable or disable the global dirty log mode.

Arguments:

enable: boolean
true to enable, false to disable.

Returns: nothing

Since: 1.3

Example:

```
-> { "execute": "xen-set-global-dirty-log",
      "arguments": { "enable": true } }
<- { "return": {} }
```

device_add [Command]

Arguments:

driver: string
the name of the new device's driver

bus: string (optional)
the device's parent bus (device tree path)

id: string (optional)
the device's ID, must be unique

Additional arguments depend on the type.

Add a device.

Notes:

1. For detailed information about this command, please refer to the 'docs/qdev-device-use.txt' file.
2. It's possible to list device properties by running QEMU with the "-device DEVICE,help" command-line argument, where DEVICE is the device's name

Example:

```
-> { "execute": "device_add",
      "arguments": { "driver": "e1000", "id": "net1",
                    "bus": "pci.0",
                    "mac": "52:54:00:12:34:56" } }
<- { "return": {} }
```

TODO: This command effectively bypasses QAPI completely due to its "additional arguments" business. It shouldn't have been added to the schema in this form. It should be qapified properly, or replaced by a properly qapified command.

Since: 0.13

`device_del` [Command]

Remove a device from a guest

Arguments:

`id`: string
the device's ID or QOM path

Returns: Nothing on success If `id` is not a valid device, DeviceNotFound

Notes: When this command completes, the device may not be removed from the guest. Hot removal is an operation that requires guest cooperation. This command merely requests that the guest begin the hot removal process. Completion of the device removal process is signaled with a DEVICE_DELETED event. Guest reset will automatically complete removal for all devices.

Since: 0.14.0

Example:

```
-> { "execute": "device_del",
      "arguments": { "id": "net1" } }
<- { "return": {} }

-> { "execute": "device_del",
      "arguments": { "id": "/machine/peripheral-anon/device[0]" } }
<- { "return": {} }
```

`DEVICE_DELETED` [Event]

Emitted whenever the device removal completion is acknowledged by the guest. At this point, it's safe to reuse the specified device ID. Device removal can be initiated by the guest or by HMP/QMP commands.

Arguments:

`device`: string (optional)
device name

`path`: string
device path

Since: 1.5

Example:

```
<- { "event": "DEVICE_DELETED",
      "data": { "device": "virtio-net-pci-0",
                "path": "/machine/peripheral/virtio-net-pci-0" },
      "timestamp": { "seconds": 1265044230, "microseconds": 450486 } }
```

DumpGuestMemoryFormat

[Enum]

An enumeration of guest-memory-dump's format.

Values:

`elf` elf format

`kdump-zlib`
kdump-compressed format with zlib-compressed

`kdump-lzo`
kdump-compressed format with lzo-compressed

`kdump-snappy`
kdump-compressed format with snappy-compressed

Since: 2.0

dump-guest-memory

[Command]

Dump guest's memory to vmcore. It is a synchronous operation that can take very long depending on the amount of guest memory.

Arguments:

`paging`: boolean

if true, do paging to get guest's memory mapping. This allows using gdb to process the core file.

IMPORTANT: this option can make QEMU allocate several gigabytes of RAM. This can happen for a large guest, or a malicious guest pretending to be large.

Also, `paging=true` has the following limitations:

1. The guest may be in a catastrophic state or can have corrupted memory, which cannot be trusted
2. The guest can be in real-mode even if paging is enabled. For example, the guest uses ACPI to sleep, and ACPI sleep state goes in real-mode
3. Currently only supported on i386 and x86_64.

protocol: `string`
 the filename or file descriptor of the vmcore. The supported protocols are:

1. `file`: the protocol starts with "file:", and the following string is the file's path.
2. `fd`: the protocol starts with "fd:", and the following string is the fd's name.

detach: `boolean` (optional)
 if true, QMP will return immediately rather than waiting for the dump to finish. The user can track progress using "query-dump". (since 2.6).

begin: `int` (optional)
 if specified, the starting physical address.

length: `int` (optional)
 if specified, the memory size, in bytes. If you don't want to dump all guest's memory, please specify the start `begin` and `length`

format: `DumpGuestMemoryFormat` (optional)
 if specified, the format of guest memory dump. But non-elf format is conflict with paging and filter, ie. `paging`, `begin` and `length` is not allowed to be specified with non-elf `format` at the same time (since 2.0)

Note: All boolean arguments default to false

Returns: nothing on success

Since: 1.2

Example:

```
-> { "execute": "dump-guest-memory",
      "arguments": { "protocol": "fd:dump" } }
<- { "return": {} }
```

DumpStatus [Enum]

Describe the status of a long-running background guest memory dump.

Values:

`none` no dump-guest-memory has started yet.
`active` there is one dump running in background.
`completed` the last dump has finished successfully.
`failed` the last dump has failed.

Since: 2.6

DumpQueryResult [Object]

The result format for 'query-dump'.

Members:

status: `DumpStatus`
 enum of `DumpStatus`, which shows current dump status

completed: int
bytes written in latest dump (uncompressed)

total: int
total bytes to be written in latest dump (uncompressed)

Since: 2.6

query-dump [Command]

Query latest dump status.

Returns: A DumpStatus object showing the dump status.

Since: 2.6

Example:

```
-> { "execute": "query-dump" }
<- { "return": { "status": "active", "completed": 1024000,
                "total": 2048000 } }
```

DUMP_COMPLETED [Event]

Emitted when background dump has completed

Arguments:

result: DumpQueryResult
DumpQueryResult type described in qapi-schema.json.

error: string (optional)
human-readable error string that provides hint on why dump failed. Only presents on failure. The user should not try to interpret the error string.

Since: 2.6

Example:

```
{ "event": "DUMP_COMPLETED",
  "data": { "result": { "total": 1090650112, "status": "completed",
                      "completed": 1090650112 } } }
```

DumpGuestMemoryCapability [Object]

A list of the available formats for dump-guest-memory

Members:

formats: array of DumpGuestMemoryFormat
Not documented

Since: 2.0

query-dump-guest-memory-capability [Command]

Returns the available formats for dump-guest-memory

Returns: A DumpGuestMemoryCapability object listing available formats for dump-guest-memory

Since: 2.0

Example:

```
-> { "execute": "query-dump-guest-memory-capability" }
<- { "return": { "formats":
                ["elf", "kdump-zlib", "kdump-lzo", "kdump-snappy"] } }
```

dump-skeys [Command]

Dump guest's storage keys

Arguments:

filename: string
the path to the file to dump to

This command is only supported on s390 architecture.

Since: 2.5

Example:

```
-> { "execute": "dump-skeys",
      "arguments": { "filename": "/tmp/skeys" } }
<- { "return": {} }
```

object-add [Command]

Create a QOM object.

Arguments:

qom-type: string
the class name for the object to be created

id: string
the name of the new object

props: value (optional)
a dictionary of properties to be passed to the backend

Returns: Nothing on success Error if **qom-type** is not a valid class name

Since: 2.0

Example:

```
-> { "execute": "object-add",
      "arguments": { "qom-type": "rng-random", "id": "rng1",
                    "props": { "filename": "/dev/hwrng" } } }
<- { "return": {} }
```

object-del [Command]

Remove a QOM object.

Arguments:

id: string
the name of the QOM object to remove

Returns: Nothing on success Error if **id** is not a valid id for a QOM object

Since: 2.0

Example:

```
-> { "execute": "object-del", "arguments": { "id": "rng1" } }
<- { "return": {} }
```

getfd [Command]

Receive a file descriptor via SCM rights and assign it a name

Arguments:

fdname: string
file descriptor name

Returns: Nothing on success

Since: 0.14.0

Notes: If **fdname** already exists, the file descriptor assigned to it will be closed and replaced by the received file descriptor.

The 'closefd' command can be used to explicitly close the file descriptor when it is no longer needed.

Example:

```
-> { "execute": "getfd", "arguments": { "fdname": "fd1" } }
<- { "return": {} }
```

closefd [Command]

Close a file descriptor previously passed via SCM rights

Arguments:

fdname: string
file descriptor name

Returns: Nothing on success

Since: 0.14.0

Example:

```
-> { "execute": "closefd", "arguments": { "fdname": "fd1" } }
<- { "return": {} }
```

MachineInfo [Object]

Information describing a machine.

Members:

name: string
the name of the machine

alias: string (optional)
an alias for the machine name

is-default: boolean (optional)
whether the machine is default

cpu-max: int
maximum number of CPUs supported by the machine type (since 1.5.0)

hotpluggable-cpus: boolean
cpu hotplug via -device is supported (since 2.7.0)

Since: 1.2.0

`query-machines` [Command]

Return a list of supported machines

Returns: a list of MachineInfo

Since: 1.2.0

`CpuDefinitionInfo` [Object]

Virtual CPU definition.

Members:

name: string

the name of the CPU definition

migration-safe: boolean (optional)

whether a CPU definition can be safely used for migration in combination with a QEMU compatibility machine when migrating between different QEMU versions and between hosts with different sets of (hardware or software) capabilities. If not provided, information is not available and callers should not assume the CPU definition to be migration-safe. (since 2.8)

static: boolean

whether a CPU definition is static and will not change depending on QEMU version, machine type, machine options and accelerator options. A static model is always migration-safe. (since 2.8)

unavailable-features: array of string (optional)

List of properties that prevent the CPU model from running in the current host. (since 2.8)

typename: string

Type name that can be used as argument to `device-list-properties`, to introspect properties configurable using `-cpu` or `-global`. (since 2.9)

unavailable-features is a list of QOM property names that represent CPU model attributes that prevent the CPU from running. If the QOM property is read-only, that means there's no known way to make the CPU model run in the current host. Implementations that choose not to provide specific information return the property name "type". If the property is read-write, it means that it MAY be possible to run the CPU model in the current host if that property is changed. Management software can use it as hints to suggest or choose an alternative for the user, or just to generate meaningful error messages explaining why the CPU model can't be used. If **unavailable-features** is an empty list, the CPU model is runnable using the current host and machine-type. If **unavailable-features** is not present, runnability information for the CPU is not available.

Since: 1.2.0

`query-cpu-definitions` [Command]

Return a list of supported virtual CPU definitions

Returns: a list of CpuDefInfo

Since: 1.2.0

CpuModelInfo [Object]

Virtual CPU model.

A CPU model consists of the name of a CPU definition, to which delta changes are applied (e.g. features added/removed). Most magic values that an architecture might require should be hidden behind the name. However, if required, architectures can expose relevant properties.

Members:

name: `string`

the name of the CPU definition the model is based on

props: `value` (optional)

a dictionary of QOM properties to be applied

Since: 2.8.0

CpuModelExpansionType [Enum]

An enumeration of CPU model expansion types.

Values:

static Expand to a static CPU model, a combination of a static base model name and property delta changes. As the static base model will never change, the expanded CPU model will be the same, independent of independent of QEMU version, machine type, machine options, and accelerator options. Therefore, the resulting model can be used by tooling without having to specify a compatibility machine - e.g. when displaying the "host" model. static CPU models are migration-safe.

full Expand all properties. The produced model is not guaranteed to be migration-safe, but allows tooling to get an insight and work with model details.

Note: When a non-migration-safe CPU model is expanded in static mode, some features enabled by the CPU model may be omitted, because they can't be implemented by a static CPU model definition (e.g. cache info passthrough and PMU passthrough in x86). If you need an accurate representation of the features enabled by a non-migration-safe CPU model, use `full`. If you need a static representation that will keep ABI compatibility even when changing QEMU version or machine-type, use `static` (but keep in mind that some features may be omitted).

Since: 2.8.0

CpuModelExpansionInfo [Object]

The result of a cpu model expansion.

Members:

model: `CpuModelInfo`

the expanded `CpuModelInfo`.

Since: 2.8.0

query-cpu-model-expansion [Command]

Expands a given CPU model (or a combination of CPU model + additional options) to different granularities, allowing tooling to get an understanding what a specific CPU model looks like in QEMU under a certain configuration.

This interface can be used to query the "host" CPU model.

The data returned by this command may be affected by:

- QEMU version: CPU models may look different depending on the QEMU version. (Except for CPU models reported as "static" in query-cpu-definitions.)
- machine-type: CPU model may look different depending on the machine-type. (Except for CPU models reported as "static" in query-cpu-definitions.)
- machine options (including accelerator): in some architectures, CPU models may look different depending on machine and accelerator options. (Except for CPU models reported as "static" in query-cpu-definitions.)
- "-cpu" arguments and global properties: arguments to the -cpu option and global properties may affect expansion of CPU models. Using query-cpu-model-expansion while using these is not advised.

Some architectures may not support all expansion types. s390x supports "full" and "static".

Arguments:

type: CpuModelExpansionType
Not documented

model: CpuModelInfo
Not documented

Returns: a CpuModelExpansionInfo. Returns an error if expanding CPU models is not supported, if the model cannot be expanded, if the model contains an unknown CPU definition name, unknown properties or properties with a wrong type. Also returns an error if an expansion type is not supported.

Since: 2.8.0

CpuModelCompareResult [Enum]

An enumeration of CPU model comparison results. The result is usually calculated using e.g. CPU features or CPU generations.

Values:

incompatible

If model A is incompatible to model B, model A is not guaranteed to run where model B runs and the other way around.

identical

If model A is identical to model B, model A is guaranteed to run where model B runs and the other way around.

superset

If model A is a superset of model B, model B is guaranteed to run where model A runs. There are no guarantees about the other way.

subset If model A is a subset of model B, model A is guaranteed to run where model B runs. There are no guarantees about the other way.

Since: 2.8.0

CpuModelCompareInfo [Object]

The result of a CPU model comparison.

Members:

result: **CpuModelCompareResult**

The result of the compare operation.

responsible-properties: array of string

List of properties that led to the comparison result not being identical.

responsible-properties is a list of QOM property names that led to both CPUs not being detected as identical. For identical models, this list is empty. If a QOM property is read-only, that means there's no known way to make the CPU models identical. If the special property name "type" is included, the models are by definition not identical and cannot be made identical.

Since: 2.8.0

query-cpu-model-comparison [Command]

Compares two CPU models, returning how they compare in a specific configuration. The results indicates how both models compare regarding runnability. This result can be used by tooling to make decisions if a certain CPU model will run in a certain configuration or if a compatible CPU model has to be created by baselining.

Usually, a CPU model is compared against the maximum possible CPU model of a certain configuration (e.g. the "host" model for KVM). If that CPU model is identical or a subset, it will run in that configuration.

The result returned by this command may be affected by:

- QEMU version: CPU models may look different depending on the QEMU version. (Except for CPU models reported as "static" in query-cpu-definitions.)
- machine-type: CPU model may look different depending on the machine-type. (Except for CPU models reported as "static" in query-cpu-definitions.)
- machine options (including accelerator): in some architectures, CPU models may look different depending on machine and accelerator options. (Except for CPU models reported as "static" in query-cpu-definitions.)
- "-cpu" arguments and global properties: arguments to the -cpu option and global properties may affect expansion of CPU models. Using query-cpu-model-expansion while using these is not advised.

Some architectures may not support comparing CPU models. s390x supports comparing CPU models.

Arguments:

modela: **CpuModelInfo**

Not documented

modelb: `CpuModelInfo`
Not documented

Returns: a `CpuModelBaselineInfo`. Returns an error if comparing CPU models is not supported, if a model cannot be used, if a model contains an unknown cpu definition name, unknown properties or properties with wrong types.

Since: 2.8.0

`CpuModelBaselineInfo` [Object]

The result of a CPU model baseline.

Members:

model: `CpuModelInfo`
the baselined `CpuModelInfo`.

Since: 2.8.0

`query-cpu-model-baseline` [Command]

Baseline two CPU models, creating a compatible third model. The created model will always be a static, migration-safe CPU model (see "static" CPU model expansion for details).

This interface can be used by tooling to create a compatible CPU model out two CPU models. The created CPU model will be identical to or a subset of both CPU models when comparing them. Therefore, the created CPU model is guaranteed to run where the given CPU models run.

The result returned by this command may be affected by:

- QEMU version: CPU models may look different depending on the QEMU version. (Except for CPU models reported as "static" in query-cpu-definitions.)
- machine-type: CPU model may look different depending on the machine-type. (Except for CPU models reported as "static" in query-cpu-definitions.)
- machine options (including accelerator): in some architectures, CPU models may look different depending on machine and accelerator options. (Except for CPU models reported as "static" in query-cpu-definitions.)
- "-cpu" arguments and global properties: arguments to the -cpu option and global properties may affect expansion of CPU models. Using query-cpu-model-expansion while using these is not advised.

Some architectures may not support baselining CPU models. s390x supports baselining CPU models.

Arguments:

modela: `CpuModelInfo`
Not documented

modelb: `CpuModelInfo`
Not documented

Returns: a `CpuModelBaselineInfo`. Returns an error if baselining CPU models is not supported, if a model cannot be used, if a model contains an unknown cpu definition name, unknown properties or properties with wrong types.

Since: 2.8.0

AddfdInfo [Object]

Information about a file descriptor that was added to an fd set.

Members:

fdset-id: `int`

The ID of the fd set that `fd` was added to.

fd: `int` The file descriptor that was received via SCM rights and added to the fd set.

Since: 1.2.0

add-fd [Command]

Add a file descriptor, that was passed via SCM rights, to an fd set.

Arguments:

fdset-id: `int` (optional)

The ID of the fd set to add the file descriptor to.

opaque: `string` (optional)

A free-form string that can be used to describe the fd.

Returns: `AddfdInfo` on success

If file descriptor was not received, `FdNotSupplied`

If `fdset-id` is a negative value, `InvalidParameterValue`

Notes: The list of fd sets is shared by all monitor connections.

If `fdset-id` is not specified, a new fd set will be created.

Since: 1.2.0

Example:

```
-> { "execute": "add-fd", "arguments": { "fdset-id": 1 } }
<- { "return": { "fdset-id": 1, "fd": 3 } }
```

remove-fd [Command]

Remove a file descriptor from an fd set.

Arguments:

fdset-id: `int`

The ID of the fd set that the file descriptor belongs to.

fd: `int` (optional)

The file descriptor that is to be removed.

Returns: Nothing on success If `fdset-id` or `fd` is not found, `FdNotFound`

Since: 1.2.0

Notes: The list of fd sets is shared by all monitor connections.

If `fd` is not specified, all file descriptors in `fdset-id` will be removed.

Example:

```
-> { "execute": "remove-fd", "arguments": { "fdset-id": 1, "fd": 3 } }
<- { "return": {} }
```

FdsetFdInfo [Object]

Information about a file descriptor that belongs to an fd set.

Members:

`fd`: `int` The file descriptor value.

`opaque`: `string` (optional)

A free-form string that can be used to describe the fd.

Since: 1.2.0

FdsetInfo [Object]

Information about an fd set.

Members:

`fdset-id`: `int`

The ID of the fd set.

`fds`: array of `FdsetFdInfo`

A list of file descriptors that belong to this fd set.

Since: 1.2.0

query-fdsets [Command]

Return information describing all fd sets.

Returns: A list of `FdsetInfo`

Since: 1.2.0

Note: The list of fd sets is shared by all monitor connections.

Example:

```
-> { "execute": "query-fdsets" }
<- { "return": [
  {
    "fds": [
      {
        "fd": 30,
        "opaque": "ronly:/path/to/file"
      },
      {
        "fd": 24,
        "opaque": "rdwr:/path/to/file"
      }
    ],
    "fdset-id": 1
  },
  {
    "fds": [
      {
        "fd": 28
      },
    ],
```

```

        {
            "fd": 29
        }
    ],
    "fdset-id": 0
}
]
}

```

TargetInfo [Object]

Information describing the QEMU target.

Members:

arch: string
the target architecture (eg "x86_64", "i386", etc)

Since: 1.2.0

query-target [Command]

Return information about the target for this QEMU

Returns: TargetInfo

Since: 1.2.0

AcpiTableOptions [Object]

Specify an ACPI table on the command line to load.

At most one of **file** and **data** can be specified. The list of files specified by any one of them is loaded and concatenated in order. If both are omitted, **data** is implied.

Other fields / optargs can be used to override fields of the generic ACPI table header; refer to the ACPI specification 5.0, section 5.2.6 System Description Table Header. If a header field is not overridden, then the corresponding value from the concatenated blob is used (in case of **file**), or it is filled in with a hard-coded value (in case of **data**).

String fields are copied into the matching ACPI member from lowest address upwards, and silently truncated / NUL-padded to length.

Members:

sig: string (optional)
table signature / identifier (4 bytes)

rev: int (optional)
table revision number (dependent on signature, 1 byte)

oem_id: string (optional)
OEM identifier (6 bytes)

oem_table_id: string (optional)
OEM table identifier (8 bytes)

oem_rev: int (optional)
OEM-supplied revision number (4 bytes)

asl_compiler_id: `string` (optional)
 identifier of the utility that created the table (4 bytes)

asl_compiler_rev: `int` (optional)
 revision number of the utility that created the table (4 bytes)

file: `string` (optional)
 colon (:) separated list of pathnames to load and concatenate as table data. The resultant binary blob is expected to have an ACPI table header. At least one file is required. This field excludes `data`.

data: `string` (optional)
 colon (:) separated list of pathnames to load and concatenate as table data. The resultant binary blob must not have an ACPI table header. At least one file is required. This field excludes `file`.

Since: 1.5

CommandLineParameterType [Enum]

Possible types for an option parameter.

Values:

string accepts a character string

boolean accepts "on" or "off"

number accepts a number

size accepts a number followed by an optional suffix (K)ilo, (M)ega, (G)iga, (T)era

Since: 1.5

CommandLineParameterInfo [Object]

Details about a single parameter of a command line option.

Members:

name: `string`
 parameter name

type: `CommandLineParameterType`
 parameter `CommandLineParameterType`

help: `string` (optional)
 human readable text string, not suitable for parsing.

default: `string` (optional)
 default value string (since 2.1)

Since: 1.5

CommandLineOptionInfo [Object]

Details about a command line option, including its list of parameter details

Members:

option: `string`
 option name

parameters: array of CommandLineParameterInfo
 an array of CommandLineParameterInfo

Since: 1.5

query-command-line-options [Command]

Query command line option schema.

Arguments:

option: string (optional)
 option name

Returns: list of CommandLineOptionInfo for all options (or for the given option).
 Returns an error if the given option doesn't exist.

Since: 1.5

Example:

```
-> { "execute": "query-command-line-options",
      "arguments": { "option": "option-rom" } }
<- { "return": [
      {
        "parameters": [
          {
            "name": "romfile",
            "type": "string"
          },
          {
            "name": "bootindex",
            "type": "number"
          }
        ],
        "option": "option-rom"
      }
    ]
  }
```

X86CPURegister32 [Enum]

A X86 32-bit register

Values:

EAX	Not documented
EBX	Not documented
ECX	Not documented
EDX	Not documented
ESP	Not documented
EBP	Not documented
ESI	Not documented

EDI Not documented

Since: 1.5

X86CPUFeatureWordInfo [Object]

Information about a X86 CPU feature word

Members:

`cpuid-input-eax`: int

Input EAX value for CPUID instruction for that feature word

`cpuid-input-ecx`: int (optional)

Input ECX value for CPUID instruction for that feature word

`cpuid-register`: X86CPURegister32

Output register containing the feature bits

`features`: int

value of output register, containing the feature bits

Since: 1.5

DummyForceArrays [Object]

Not used by QMP; hack to let us use X86CPUFeatureWordInfoList internally

Members:

`unused`: array of X86CPUFeatureWordInfo

Not documented

Since: 2.5

NumaOptionsType [Enum]

Values:

`node` NUMA nodes configuration

`dist` NUMA distance configuration (since 2.10)

`cpu` property based CPU(s) to node mapping (Since: 2.10)

Since: 2.1

NumaOptions [Object]

A discriminated record of NUMA options. (for OptsVisitor)

Members:

`type`: NumaOptionsType

Not documented

The members of `NumaNodeOptions` when `type` is "node"

The members of `NumaDistOptions` when `type` is "dist"

The members of `NumaCpuOptions` when `type` is "cpu"

Since: 2.1

NumaNodeOptions [Object]

Create a guest NUMA node. (for OptsVisitor)

Members:

nodeid: int (optional)

NUMA node ID (increase by 1 from 0 if omitted)

cpus: array of int (optional)

VCPUs belonging to this node (assign VCPUS round-robin if omitted)

mem: int (optional)

memory size of this node; mutually exclusive with **memdev**. Equally divide total memory among nodes if both **mem** and **memdev** are omitted.

memdev: string (optional)

memory backend object. If specified for one node, it must be specified for all nodes.

Since: 2.1

NumaDistOptions [Object]

Set the distance between 2 NUMA nodes.

Members:

src: int source NUMA node.

dst: int destination NUMA node.

val: int NUMA distance from source node to destination node. When a node is unreachable from another node, set the distance between them to 255.

Since: 2.10

NumaCpuOptions [Object]

Option "-numa cpu" overrides default cpu to node mapping. It accepts the same set of cpu properties as returned by `query-hotpluggable-cpus[].props`, where `node-id` could be used to override default node mapping.

Members:

The members of `CpuInstanceProperties`

Since: 2.10

HostMemPolicy [Enum]

Host memory policy types

Values:

default restore default policy, remove any nondefault policy

preferred

set the preferred host nodes for allocation

bind a strict policy that restricts memory allocation to the host nodes specified

interleave

memory allocations are interleaved across the set of host nodes specified

Since: 2.1

Memdev

[Object]

Information about memory backend

Members:**id:** string (optional)

backend's ID if backend has 'id' property (since 2.9)

size: int memory backend size**merge:** boolean

enables or disables memory merge support

dump: boolean

includes memory backend's memory in a core dump or not

prealloc: boolean

enables or disables memory preallocation

host-nodes: array of int

host nodes for its memory policy

policy: HostMemPolicy

memory policy of memory backend

Since: 2.1**query-memdev**

[Command]

Returns information for all memory backends.

Returns: a list of Memdev.**Since:** 2.1**Example:**

```
-> { "execute": "query-memdev" }
```

```
<- { "return": [
  {
    "id": "mem1",
    "size": 536870912,
    "merge": false,
    "dump": true,
    "prealloc": false,
    "host-nodes": [0, 1],
    "policy": "bind"
  },
  {
    "size": 536870912,
    "merge": false,
    "dump": true,
    "prealloc": true,
    "host-nodes": [2, 3],
    "policy": "preferred"
  }
]
}
```

PCDIMMDeviceInfo [Object]

PCDIMMDevice state information

Members:

`id: string` (optional)
device's ID

`addr: int` physical address, where device is mapped

`size: int` size of memory that the device provides

`slot: int` slot number at which device is plugged in

`node: int` NUMA node number where device is plugged in

`memdev: string`
memory backend linked with device

`hotplugged: boolean`
true if device was hotplugged

`hotpluggable: boolean`
true if device if could be added/removed while machine is running

Since: 2.1

MemoryDeviceInfo [Object]

Union containing information about a memory device

Members:

`type` One of "dimm"

`data: PCDIMMDeviceInfo` when type is "dimm"

Since: 2.1

query-memory-devices [Command]

Lists available memory devices and their state

Since: 2.1

Example:

```
-> { "execute": "query-memory-devices" }
<- { "return": [ { "data":
    { "addr": 5368709120,
      "hotpluggable": true,
      "hotplugged": true,
      "id": "d1",
      "memdev": "/objects/memX",
      "node": 0,
      "size": 1073741824,
      "slot": 0},
    "type": "dimm"
  } ] }
```

MEM_UNPLUG_ERROR [Event]

Emitted when memory hot unplug error occurs.

Arguments:

device: string
device name

msg: string
Informative message

Since: 2.4

Example:

```
<- { "event": "MEM_UNPLUG_ERROR"
      "data": { "device": "dimm1",
                "msg": "acpi: device unplug for unsupported device"
              },
      "timestamp": { "seconds": 1265044230, "microseconds": 450486 } }
```

ACPISlotType [Enum]

Values:

DIMM memory slot

CPU logical CPU slot (since 2.7)

ACPIOSTInfo [Object]

OSPM Status Indication for a device For description of possible values of **source** and **status** fields see "_OST (OSPM Status Indication)" chapter of ACPI5.0 spec.

Members:

device: string (optional)
device ID associated with slot

slot: string
slot ID, unique per slot of a given **slot-type**

slot-type: ACPISlotType
type of the slot

source: int
an integer containing the source event

status: int
an integer containing the status code

Since: 2.1

query-acpi-ospm-status [Command]

Return a list of ACPIOSTInfo for devices that support status reporting via ACPI _OST method.

Since: 2.1

Example:

```
-> { "execute": "query-acpi-ospm-status" }
```

```

<- { "return": [ { "device": "d1", "slot": "0", "slot-type": "DIMM", "source": 1, "sta
                { "slot": "1", "slot-type": "DIMM", "source": 0, "status": 0},
                { "slot": "2", "slot-type": "DIMM", "source": 0, "status": 0},
                { "slot": "3", "slot-type": "DIMM", "source": 0, "status": 0}
      ]}

```

ACPI_DEVICE_OST [Event]

Emitted when guest executes ACPI _OST method.

Arguments:

info: ACPIOSTInfo

ACPIOSTInfo type as described in qapi-schema.json

Since: 2.1

Example:

```

<- { "event": "ACPI_DEVICE_OST",
      "data": { "device": "d1", "slot": "0",
                "slot-type": "DIMM", "source": 1, "status": 0 } }

```

rtc-reset-reinjection [Command]

This command will reset the RTC interrupt reinjection backlog. Can be used if another mechanism to synchronize guest time is in effect, for example QEMU guest agent's guest-set-time command.

Since: 2.1

Example:

```

-> { "execute": "rtc-reset-reinjection" }
<- { "return": {} }

```

RTC_CHANGE [Event]

Emitted when the guest changes the RTC time.

Arguments:

offset: int

offset between base RTC clock (as specified by -rtc base), and new RTC clock value

Note: This event is rate-limited.

Since: 0.13.0

Example:

```

<- { "event": "RTC_CHANGE",
      "data": { "offset": 78 },
      "timestamp": { "seconds": 1267020223, "microseconds": 435656 } }

```

ReplayMode [Enum]

Mode of the replay subsystem.

Values:

none normal execution mode. Replay or record are not enabled.

record record mode. All non-deterministic data is written into the replay log.

play replay mode. Non-deterministic data required for system execution is read from the log.

Since: 2.5

xen-load-devices-state [Command]

Load the state of all devices from file. The RAM and the block devices of the VM are not loaded by this command.

Arguments:

filename: string
the file to load the state of the devices from as binary data. See xen-save-devices-state.txt for a description of the binary format.

Since: 2.7

Example:

```
-> { "execute": "xen-load-devices-state",
      "arguments": { "filename": "/tmp/resume" } }
<- { "return": {} }
```

GICCapability [Object]

The struct describes capability for a specific GIC (Generic Interrupt Controller) version. These bits are not only decided by QEMU/KVM software version, but also decided by the hardware that the program is running upon.

Members:

version: int
version of GIC to be described. Currently, only 2 and 3 are supported.

emulated: boolean
whether current QEMU/hardware supports emulated GIC device in user space.

kernel: boolean
whether current QEMU/hardware supports hardware accelerated GIC device in kernel.

Since: 2.6

query-gic-capabilities [Command]

This command is ARM-only. It will return a list of GICCapability objects that describe its capability bits.

Returns: a list of GICCapability objects.

Since: 2.6

Example:

```
-> { "execute": "query-gic-capabilities" }
<- { "return": [{ "version": 2, "emulated": true, "kernel": false },
                { "version": 3, "emulated": false, "kernel": true } ] }
```

CpuInstanceProperties [Object]

List of properties to be used for hotplugging a CPU instance, it should be passed by management with `device_add` command when a CPU is being hotplugged.

Members:

`node-id`: int (optional)

NUMA node ID the CPU belongs to

`socket-id`: int (optional)

socket number within node/board the CPU belongs to

`core-id`: int (optional)

core number within socket the CPU belongs to

`thread-id`: int (optional)

thread number within core the CPU belongs to

Note: currently there are 4 properties that could be present but management should be prepared to pass through other properties with `device_add` command to allow for future interface extension. This also requires the filed names to be kept in sync with the properties passed to `-device/device_add`.

Since: 2.7

HotpluggableCPU [Object]**Members:**

`type`: string

CPU object type for usage with `device_add` command

`props`: **CpuInstanceProperties**

list of properties to be used for hotplugging CPU

`vcpus-count`: int

number of logical VCPU threads **HotpluggableCPU** provides

`qom-path`: string (optional)

link to existing CPU object if CPU is present or omitted if CPU is not present.

Since: 2.7

query-hotpluggable-cpus [Command]

Returns: a list of **HotpluggableCPU** objects.

Since: 2.7

Example:

For pseries machine type started with `-smp 2,cores=2,maxcpus=4 -cpu POWER8:`

```
-> { "execute": "query-hotpluggable-cpus" }
<- {"return": [
  { "props": { "core": 8 }, "type": "POWER8-spapr-cpu-core",
    "vcpus-count": 1 },
  { "props": { "core": 0 }, "type": "POWER8-spapr-cpu-core",
```

```
    "vcpus-count": 1, "qom-path": "/machine/unattached/device[0]" }
  ]}'
```

For pc machine type started with `-smp 1,maxcpus=2`:

```
-> { "execute": "query-hotpluggable-cpus" }
<- {"return": [
  {
    "type": "qemu64-x86_64-cpu", "vcpus-count": 1,
    "props": {"core-id": 0, "socket-id": 1, "thread-id": 0}
  },
  {
    "qom-path": "/machine/unattached/device[0]",
    "type": "qemu64-x86_64-cpu", "vcpus-count": 1,
    "props": {"core-id": 0, "socket-id": 0, "thread-id": 0}
  }
]}
```

GuidInfo [Object]

GUID information.

Members:

`guid: string`
the globally unique identifier

Since: 2.9

query-vm-generation-id [Command]

Show Virtual Machine Generation ID

Since 2.9

Commands and Events Index

A

ACPI_DEVICE_OST 202
 add-fd 192
 add_client 161

B

balloon 176
 BALLOON_CHANGE 167
 block-commit 36
 block-dirty-bitmap-add 41
 block-dirty-bitmap-clear 42
 block-dirty-bitmap-remove 42
 block-job-cancel 49
 block-job-complete 50
 block-job-pause 49
 block-job-resume 49
 block-job-set-speed 48
 block-set-write-threshold 80
 block-stream 47
 block_passwd 32
 block_resize 32
 block_set_io_throttle 44
 BLOCK_IMAGE_CORRUPTED 76
 BLOCK_IO_ERROR 77
 BLOCK_JOB_CANCELED 78
 BLOCK_JOB_COMPLETED 77
 BLOCK_JOB_ERROR 78
 BLOCK_JOB_READY 79
 BLOCK_WRITE_THRESHOLD 80
 blockdev-add 70
 blockdev-backup 38
 blockdev-change-medium 74
 blockdev-close-tray 72
 blockdev-del 71
 blockdev-mirror 42
 blockdev-open-tray 71
 blockdev-snapshot 35
 blockdev-snapshot-delete-internal-sync 83
 blockdev-snapshot-internal-sync 83
 blockdev-snapshot-sync 35

C

change 178
 change-backing-file 36
 change-vnc-password 123
 chardev-add 93
 chardev-change 94
 chardev-remove 94
 chardev-send-break 95
 client_migrate_info 145
 closefd 186
 cont 175

cpu 174
 cpu-add 174

D

device-list-properties 180
 device_add 180
 device_del 181
 DEVICE_DELETED 181
 DEVICE_TRAY_MOVED 85
 drive-backup 37
 drive-mirror 39
 dump-guest-memory 182
 dump-skeys 185
 DUMP_COMPLETED 184

E

eject 84
 expire_password 115

G

getfd 186
 GUEST_PANICKED 9

H

human-monitor-command 176

I

inject-nmi 176
 input-send-event 133

M

MEM_UNPLUG_ERROR 201
 memsave 174
 migrate 149
 migrate-incoming 149
 migrate-set-cache-size 148
 migrate-set-capabilities 140
 migrate-set-parameters 143
 migrate-start-postcopy 146
 migrate_cancel 147
 migrate_set_downtime 148
 migrate_set_speed 148
 MIGRATION 146
 MIGRATION_PASS 146

N

nbd-server-add 84
 nbd-server-start 84
 nbd-server-stop 85
 netdev_add 96
 netdev_del 96
 NIC_RX_FILTER_CHANGED 106

O

object-add 185
 object-del 185

P

pmemsave 175
 POWERDOWN 7

Q

qmp_capabilities 159
 qom-get 177
 qom-list 177
 qom-list-types 179
 qom-set 178
 query-acpi-ospm-status 201
 query-balloon 167
 query-block 23
 query-block-jobs 32
 query-blockstats 28
 query-chardev 87
 query-chardev-backends 88
 query-command-line-options 196
 query-commands 161
 query-cpu-definitions 187
 query-cpu-model-baseline 191
 query-cpu-model-comparison 190
 query-cpu-model-expansion 189
 query-cpus 165
 query-dump 184
 query-dump-guest-memory-capability 184
 query-events 163
 query-fdsets 193
 query-gic-capabilities 203
 query-hotpluggable-cpus 204
 query-iothreads 166
 query-kvm 162
 query-machines 187
 query-memdev 199
 query-memory-devices 200
 query-mice 125
 query-migrate 137
 query-migrate-cache-size 148
 query-migrate-capabilities 141
 query-migrate-parameters 145
 query-name 162
 query-named-block-nodes 38
 query-pci 170

query-qmp-schema 155
 query-rocker 107
 query-rocker-of-dpa-flows 110
 query-rocker-of-dpa-groups 112
 query-rocker-ports 108
 query-rx-filter 105
 query-spice 117
 query-status 6
 query-target 194
 query-tpm 114
 query-tpm-models 112
 query-tpm-types 113
 query-uuid 163
 query-version 160
 query-vm-generation-id 205
 query-vnc 123
 query-vnc-servers 123
 query-xen-replication-status 151
 quit 173
 QUORUM_FAILURE 85
 QUORUM_REPORT_BAD 86

R

remove-fd 192
 RESET 7
 RESUME 7
 ringbuf-read 89
 ringbuf-write 88
 rtc-reset-reinjection 202
 RTC_CHANGE 202

S

screendump 115
 send-key 131
 set_link 95
 set_password 114
 SHUTDOWN 6
 SPICE_CONNECTED 118
 SPICE_DISCONNECTED 119
 SPICE_INITIALIZED 118
 SPICE_MIGRATE_COMPLETED 119
 stop 173
 STOP 7
 SUSPEND 7
 SUSPEND_DISK 8
 system_powerdown 174
 system_reset 173
 system_wakeup 175

T

trace-event-get-state 154
 trace-event-set-state 154
 transaction 152

V

VNC_CONNECTED	123
VNC_DISCONNECTED	124
VNC_INITIALIZED	124
VSERPORT_CHANGE	95

W

WAKEUP	8
WATCHDOG	8

X

x-blockdev-change	81
x-blockdev-insert-medium	74
x-blockdev-remove-medium	73
x-colo-lost-heartbeat	147
x-debug-block-dirty-bitmap-sha256	42
xen-colo-do-checkpoint	151
xen-load-devices-state	203
xen-save-devices-state	150
xen-set-global-dirty-log	180
xen-set-replication	150

Data Types Index

A

Abort.....	151
ACPIOSTInfo.....	201
ACPISlotType.....	201
AcpiTableOptions.....	194
ActionCompletionMode.....	151
AddfdInfo.....	192

B

BalloonInfo.....	167
BiosAtaTranslation.....	81
BlkdebugEvent.....	57
BlkdebugInjectErrorOptions.....	59
BlkdebugSetStateOptions.....	60
BlockdevAioOptions.....	51
BlockdevBackup.....	34
BlockdevCacheInfo.....	18
BlockdevCacheOptions.....	51
BlockdevChangeReadOnlyMode.....	74
BlockdevDetectZeroesOptions.....	50
BlockdevDiscardOptions.....	50
BlockdevDriver.....	51
BlockDeviceInfo.....	19
BlockDeviceIoStatus.....	21
BlockDeviceMapEntry.....	21
BlockDeviceStats.....	26
BlockDeviceTimedStats.....	25
BlockdevOnError.....	30
BlockdevOptions.....	68
BlockdevOptionsBlkdebug.....	60
BlockdevOptionsBlkverify.....	61
BlockdevOptionsCurlBase.....	65
BlockdevOptionsCurlFtp.....	67
BlockdevOptionsCurlFtps.....	67
BlockdevOptionsCurlHttp.....	66
BlockdevOptionsCurlHttps.....	66
BlockdevOptionsFile.....	52
BlockdevOptionsGenericCOWFormat.....	54
BlockdevOptionsGenericFormat.....	53
BlockdevOptionsGluster.....	62
BlockdevOptionsIscsi.....	62
BlockdevOptionsLUKS.....	53
BlockdevOptionsNbd.....	67
BlockdevOptionsNfs.....	65
BlockdevOptionsNull.....	53
BlockdevOptionsQcow.....	55
BlockdevOptionsQcow2.....	56
BlockdevOptionsQuorum.....	61
BlockdevOptionsRaw.....	67
BlockdevOptionsRbd.....	63
BlockdevOptionsReplication.....	64
BlockdevOptionsSheepdog.....	64
BlockdevOptionsSsh.....	57

BlockdevOptionsThrottle.....	68
BlockdevOptionsVVFAT.....	53
BlockdevOptionsVxHS.....	67
BlockdevQcow2Encryption.....	56
BlockdevQcow2EncryptionFormat.....	55
BlockdevQcowEncryption.....	55
BlockdevQcowEncryptionFormat.....	55
BlockdevRef.....	69
BlockdevRefOrNull.....	70
BlockdevSnapshot.....	33
BlockdevSnapshotInternal.....	82
BlockdevSnapshotSync.....	33
BlockDirtyBitmap.....	41
BlockDirtyBitmapAdd.....	41
BlockDirtyBitmapSha256.....	42
BlockDirtyInfo.....	22
BlockErrorAction.....	76
BlockInfo.....	22
BlockIOThrottle.....	44
BlockJobInfo.....	31
BlockJobType.....	31
BlockMeasureInfo.....	23
BlockStats.....	27

C

ChardevBackend.....	92
ChardevBackendInfo.....	87
ChardevCommon.....	89
ChardevFile.....	90
ChardevHostdev.....	90
ChardevInfo.....	87
ChardevMux.....	91
ChardevReturn.....	93
ChardevRingbuf.....	92
ChardevSocket.....	90
ChardevSpiceChannel.....	91
ChardevSpicePort.....	92
ChardevStdio.....	91
ChardevUdp.....	91
ChardevVC.....	92
COLOMessage.....	146
COLOMode.....	147
CommandInfo.....	160
CommandLineOptionInfo.....	195
CommandLineParameterInfo.....	195
CommandLineParameterType.....	195
CpuDefinitionInfo.....	187
CpuInfo.....	164
CpuInfoArch.....	164
CpuInfoMIPS.....	165
CpuInfoOther.....	165
CpuInfoPPC.....	165
CpuInfoSPARC.....	165

CpuInfoTricore	165
CpuInfoX86	165
CpuInstanceProperties	204
CpuModelBaselineInfo	191
CpuModelCompareInfo	190
CpuModelCompareResult	189
CpuModelExpansionInfo	188
CpuModelExpansionType	188
CpuModelInfo	188

D

DataFormat	88
DevicePropertyInfo	180
DirtyBitmapStatus	21
DriveBackup	33
DriveMirror	39
DummyForceArrays	197
DumpGuestMemoryCapability	184
DumpGuestMemoryFormat	182
DumpQueryResult	183
DumpStatus	183

E

EventInfo	163
-----------------	-----

F

FailoverStatus	147
FdsetFdInfo	193
FdsetInfo	193
FloppyDriveType	82

G

GICCapability	203
GuestPanicAction	9
GuestPanicInformation	9
GuestPanicInformationHyperV	9
GuestPanicInformationType	9
GuidInfo	205

H

HostMemPolicy	198
HotpluggableCPU	204

I

ImageCheck	17
ImageInfo	16
ImageInfoSpecific	16
ImageInfoSpecificQcow2	15
ImageInfoSpecificQcow2Encryption	15
ImageInfoSpecificQcow2EncryptionBase	15
ImageInfoSpecificVmdk	16
InetAddress	3
InetAddressBase	3
InputAxis	132
InputBtnEvent	132
InputButton	131
InputEvent	132
InputKeyEvent	132
InputMoveEvent	132
IoOperationType	2
IOThreadInfo	166
IscsiHeaderDigest	62
IscsiTransport	62

J

JSONType	157
----------------	-----

K

KeyValue	131
KvmInfo	162

L

LostTickPolicy	161
----------------------	-----

M

MachineInfo	186
MapEntry	18
Memdev	199
MemoryDeviceInfo	200
MigrateSetParameters	142
MigrationCapability	139
MigrationCapabilityStatus	140
MigrationInfo	136
MigrationParameter	141
MigrationParameters	143
MigrationStats	134
MigrationStatus	135
MirrorSyncMode	30
MouseInfo	125

N

NameInfo	162
NetClientDriver	102
Netdev	102
NetdevBridgeOptions	101
NetdevDumpOptions	101
NetdevHubPortOptions	101
NetdevL2TPv3Options	100
NetdevNetmapOptions	101
NetdevNoneOptions	97
NetdevSocketOptions	99
NetdevTapOptions	98
NetdevUserOptions	97
NetdevVdeOptions	100
NetdevVhostUserOptions	102
NetFilterDirection	104
NetLegacy	103
NetLegacyNicOptions	97
NetLegacyOptions	104
NetLegacyOptionsType	103
NetworkAddressFamily	3
NewImageMode	32
NFSServer	65
NFSTransport	64
NumaCpuOptions	198
NumaDistOptions	198
NumaNodeOptions	198
NumaOptions	197
NumaOptionsType	197

O

ObjectPropertyInfo	177
ObjectTypeInfo	179
OnOffAuto	2
OnOffSplit	2

P

PCDIMMDeviceInfo	200
PciBridgeInfo	169
PciBusInfo	168
PciDeviceClass	169
PciDeviceId	169
PciDeviceInfo	169
PciInfo	170
PciMemoryRange	168
PciMemoryRegion	168
PreallocMode	79

Q

QapiErrorClass	1
Qcow2OverlapCheckFlags	54
Qcow2OverlapCheckMode	54
Qcow2OverlapChecks	55
QCryptoBlockCreateOptions	13
QCryptoBlockCreateOptionsLUKS	12
QCryptoBlockFormat	12
QCryptoBlockInfo	14
QCryptoBlockInfoBase	13
QCryptoBlockInfoLUKS	14
QCryptoBlockInfoLUKSSlot	13
QCryptoBlockInfoQCow	14
QCryptoBlockOpenOptions	13
QCryptoBlockOptionsBase	12
QCryptoBlockOptionsLUKS	12
QCryptoBlockOptionsQCow	12
QCryptoCipherAlgorithm	11
QCryptoCipherMode	11
QCryptoHashAlgorithm	10
QCryptoIVGenAlgorithm	11
QCryptoSecretFormat	10
QCryptoTLSCredsEndpoint	10
QKeyCode	125
QuorumOpType	85
QuorumReadPattern	61

R

ReplayMode	202
ReplicationMode	64
ReplicationStatus	150
RockerOfDpaFlow	110
RockerOfDpaFlowAction	109
RockerOfDpaFlowKey	108
RockerOfDpaFlowMask	109
RockerOfDpaGroup	111
RockerPort	107
RockerPortAutoneg	107
RockerPortDuplex	107
RockerSwitch	106
RunState	5
RxFilterInfo	105
RxState	104

S

SchemaInfo	156
SchemaInfoAlternate	158
SchemaInfoAlternateMember	159
SchemaInfoArray	157
SchemaInfoBuiltin	156
SchemaInfoCommand	159
SchemaInfoEnum	157
SchemaInfoEvent	159
SchemaInfoObject	157
SchemaInfoObjectMember	158
SchemaInfoObjectVariant	158

SchemaMetaType	156
SnapshotInfo	15
SocketAddress	5
SocketAddressLegacy	4
SocketAddressType	4
SpiceBasicInfo	115
SpiceChannel	116
SpiceInfo	117
SpiceQueryMouseMode	116
SpiceServerInfo	116
StatusInfo	6
String	2
StrOrNull	3

T

TargetInfo	194
ThrottleLimits	46
TPMInfo	113
TpmModel	112
TPMPassthroughOptions	113
TpmType	113
TpmTypeOptions	113
TraceEventInfo	154
TraceEventState	153
TransactionAction	151
TransactionProperties	152

U

UnixSocketAddress	4
UuidInfo	163

V

VersionInfo	160
VersionTriple	159
VncBasicInfo	119
VncClientInfo	120
VncInfo	120
VncInfo2	122
VncPrimaryAuth	121
VncServerInfo	120
VncServerInfo2	122
VncVencryptSubAuth	121
VsockSocketAddress	4

W

WatchdogExpirationAction	8
--------------------------------	---

X

X86CPUFeatureWordInfo	197
X86CPURegister32	196
XBZRLECacheStats	135