NAME

qemu-qmp-ref − QEMU QMP Reference Manual

Contents

- *PciBusInfo (Object)*
- *PciBridgeInfo (Object)*
- *PciDeviceClass (Object)*
- *PciDeviceId (Object)*
- *PciDeviceInfo (Object)*
- *PciInfo (Object)*
- *query−pci (Command)*

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 someway, 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.

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)

QMP ERRORS

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 be 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

COMMON DATA TYPES

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

OffAutoPCIBAR (Enum)
An enumeration of options for specifying a PCI BAR

Values

off	The specified feature is disabled
auto	The PCI BAR for the feature is automatically selected
bar0	PCI BAR0 is used for the feature
bar1	PCI BAR1 is used for the feature
bar2	PCI BAR2 is used for the feature
bar3	PCI BAR3 is used for the feature
bar4	PCI BAR4 is used for the feature
bar5	PCI BAR5 is used for the feature

Since
2.12

PCIELinkSpeed (Enum)
An enumeration of PCIe link speeds in units of GT/s

Values

2_5	2.5GT/s
5	5.0GT/s
8	8.0GT/s
16	16.0GT/s

Since
4.0

PCIELinkWidth (Enum)
An enumeration of PCIe link width

Values

1	x1
2	x2
4	x4
8	x8
12	x12
16	x16
32	x32

Since
4.0

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

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

GrabToggleKeys (Enum)
Keys to toggle input−linux between host and guest.

Values

ctrl−ctrl
Not documented

alt−alt Not documented

shift−shift
Not documented

meta−meta
Not documented

scrolllock
Not documented

ctrl−scrolllock
Not documented

Since
4.0

HumanReadableText (Object)

Members

human−readable−text: string
Formatted output intended for humans.

Since
6.2

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

keep−alive: boolean (optional)
enable keep−alive when connecting to this socket. Not supported for passive sockets. (Since 4.2)

mptcp: **boolean** (optional) (**If: HAVE_IPPROTO_MPTCP**)
enable multi−path TCP. (Since 6.1)

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

abstract: **boolean** (optional) (**If: CONFIG_LINUX**)
if true, this is a Linux abstract socket address. **path** will be prefix ed by a null byte, and optionally padded with null bytes. Defaults to false. (Since 5.1)

tight: **boolean** (optional) (**If: CONFIG_LINUX**)
if false, pad an abstract socket address with enough null bytes to make it fill struct sockaddr_un member sun_path. Defaults to true. (Since 5.1)

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

InetSocketAddressWrapper (Object)
Members
data: InetSocketAddress
Not documented

Since
1.3

UnixSocketAddressWrapper (Object)
Members
data: UnixSocketAddress
Not documented

Since
1.3

VsockSocketAddressWrapper (Object)
Members
data: VsockSocketAddress
Not documented

Since
2.8

StringWrapper (Object)
Members
data: String
Not documented

Since
1.3

SocketAddressLegacy (Object)
Captures the address of a socket, which could also be a named file descriptor

Members
type: SocketAddressType
Not documented

The members of InetSocketAddressWrapper when type is "inet"

The members of UnixSocketAddressWrapper when type is "unix"

The members of VsockSocketAddressWrapper when type is "vsock"

The members of StringWrapper when type is "fd"

Note

This type is deprecated in favor of SocketAddress. The difference between SocketAddressLegacy and SocketAddress is that the latter is has fewer {} on the wire.

Since

1.3

SocketAddressType (Enum)

Available SocketAddress types

Values

inet Internet address

unix Unix domain socket

vsock VMCI address

fd decimal is for file descriptor number, otherwise a file descriptor name. Named file descriptors are permitted in monitor commands, in combination with the 'getfd' command. Decimal file descriptors are permitted at startup or other contexts where no monitor context is active.

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

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)

ShutdownCause (Enum)
An enumeration of reasons for a Shutdown.

Values

none No shutdown request pending

host−error
An error prevents further use of guest

host−qmp−quit
Reaction to the QMP command 'quit'

host−qmp−system−reset
Reaction to the QMP command 'system_reset'

host−signal
Reaction to a signal, such as SIGINT

host−ui
Reaction to a UI event, like window close

guest−shutdown
Guest shutdown/suspend request, via ACPI or other hardware−specific means

guest−reset
Guest reset request, and command line turns that into a shutdown

guest−panic
Guest panicked, and command line turns that into a shutdown

subsystem−reset
Partial guest reset that does not trigger QMP events and ignores −−no−reboot. This is useful for sanitizing hypercalls on s390 that are used during kexec/kdump/boot

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

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

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)

reason: ShutdownCause
The **ShutdownCause** which resulted in the SHUTDOWN. (since 4.0)

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

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

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)

reason: ShutdownCause

The **ShutdownCause** of the RESET. (since 4.0)

Since

0.12

Example

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

STOP (Event)

Emitted when the virtual machine is stopped

Since

0.12

Example

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

RESUME (Event)

Emitted when the virtual machine resumes execution

Since

0.12

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: WatchdogAction
action that has been taken

Note
If action is "reset", "shutdown", or "pause" the WATCHDOG event is followed respectively by the RESET, SHUTDOWN, or STOP events

Note
This event is rate−limited.

Since
0.13

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

WatchdogAction (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 **powerdown**, which tries to set to system status and notify guest
poweroff	system poweroff, the emulator program exits
pause	system pauses, similar to **stop**
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

RebootAction (Enum)
Possible QEMU actions upon guest reboot

Values

reset	Reset the VM
shutdown	Shutdown the VM and exit, according to the shutdown action

Since
6.0

ShutdownAction (Enum)
Possible QEMU actions upon guest shutdown

Values

poweroff
Shutdown the VM and exit

pause pause the VM#

Since
6.0

PanicAction (Enum)
Values

none Continue VM execution

pause Pause the VM

shutdown
Shutdown the VM and exit, according to the shutdown action

Since
6.0

watchdog−set−action (Command)
Set watchdog action

Arguments

action: WatchdogAction
Not documented

Since
2.11

set−action (Command)
Set the actions that will be taken by the emulator in response to guest events.

Arguments

reboot: RebootAction (optional)
RebootAction action taken on guest reboot.

shutdown: ShutdownAction (optional)
ShutdownAction action taken on guest shutdown.

panic: PanicAction (optional)
PanicAction action taken on guest panic.

watchdog: WatchdogAction (optional)
WatchdogAction action taken when watchdog timer expires .

Returns
Nothing on success.

Since
6.0

Example

```
-> { "execute": "set-action",
     "arguments": { "reboot": "shutdown",
                    "shutdown" : "pause",
                    "panic": "pause",
                    "watchdog": "inject-nmi" } }
<- { "return": {} }
```

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" } }
```

GUEST_CRASHLOADED (Event)
Emitted when guest OS crash loaded is detected

Arguments
action: GuestPanicAction
action that has been taken, currently always "run"

info: GuestPanicInformation (optional)
information about a panic

Since
5.0

Example
```
      <- { "event": "GUEST_CRASHLOADED",
           "data": { "action": "run" } }
```

GuestPanicAction (Enum)
An enumeration of the actions taken when guest OS panic is detected

Values
pause system pauses

poweroff
Not documented

run Not documented

Since
2.1 (poweroff since 2.8, run since 5.0)

GuestPanicInformationType (Enum)
An enumeration of the guest panic information types

Values
hyper−v
hyper−v guest panic information type

s390 s390 guest panic information type (Since: 2.12)

Since
2.9

GuestPanicInformation (Object)
Information about a guest panic

Members

type: GuestPanicInformationType
Crash type that defines the hypervisor specific information

The members of GuestPanicInformationHyperV when type is "hyper−v"

The members of GuestPanicInformationS390 when type is "s390"

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

S390CrashReason (Enum)
Reason why the CPU is in a crashed state.

Values

unknown
no crash reason was set

disabled−wait
the CPU has entered a disabled wait state

extint−loop
clock comparator or cpu timer interrupt with new PSW enabled for external interrupts

pgmint−loop
program interrupt with BAD new PSW

opint−loop
operation exception interrupt with invalid code at the program interrupt new PSW

Since
2.12

GuestPanicInformationS390 (Object)
S390 specific guest panic information (PSW)

Members

core: int
core id of the CPU that crashed

psw−mask: int
control fields of guest PSW

psw−addr: int
guest instruction address

reason: S390CrashReason
guest crash reason

Since
2.12

MEMORY_FAILURE (Event)
Emitted when a memory failure occurs on host side.

Arguments

recipient: MemoryFailureRecipient
recipient is defined as **MemoryFailureRecipient**.

action: MemoryFailureAction
action that has been taken. action is defined as **MemoryFailureAction**.

flags: MemoryFailureFlags
flags for MemoryFailureAction. action is defined as **MemoryFailureFlags**.

Since
5.2

Example

```
<- { "event": "MEMORY_FAILURE",
     "data": { "recipient": "hypervisor",
               "action": "fatal",
               "flags": { 'action−required': false } }
```

MemoryFailureRecipient (Enum)
Hardware memory failure occurs, handled by recipient.

Values

hypervisor
memory failure at QEMU process address space. (none guest memory, but used by QEMU itself).

guest memory failure at guest memory,

Since
5.2

MemoryFailureAction (Enum)
Actions taken by QEMU in response to a hardware memory failure.

Values

ignore the memory failure could be ignored. This will only be the case for action−optional failures.

inject memory failure occurred in guest memory, the guest enabled MCE handling mechanism, and QEMU could inject the MCE into the guest successfully.

fatal the failure is unrecoverable. This occurs for action−required failures if the recipient is the hypervisor; QEMU will exit.

reset the failure is unrecoverable but confined to the guest. This occurs if the recipient is a guest guest which is not ready to handle memory failures.

Since
5.2

MemoryFailureFlags (Object)
Additional information on memory failures.

Members

action−required: boolean
whether a memory failure event is action−required or action−optional (e.g. a failure during memory scrub).

recursive: boolean
whether the failure occurred while the previous failure was still in progress.

Since
5.2

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

aes−128	AES with 128 bit / 16 byte keys
aes−192	AES with 192 bit / 24 byte keys
aes−256	AES with 256 bit / 32 byte keys
des	DES with 56 bit / 8 byte keys. Do not use except in VNC. (since 6.1)
3des	3DES(EDE) with 192 bit / 24 byte keys (since 2.9)

cast5−128
Cast5 with 128 bit / 16 byte keys

serpent−128
Serpent with 128 bit / 16 byte keys

serpent−192
Serpent with 192 bit / 24 byte keys

serpent−256
Serpent with 256 bit / 32 byte keys

twofish−128
Twofish with 128 bit / 16 byte keys

twofish−192
Twofish with 192 bit / 24 byte keys

twofish−256
Twofish with 256 bit / 32 byte keys

Since
2.6

QCryptoCipherMode (Enum)
The supported modes for content encryption ciphers

Values

ecb	Electronic Code Book
cbc	Cipher Block Chaining
xts	XEX with tweaked code book and ciphertext stealing
ctr	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 2ˆ32, except where compatibility with pre−existing Linux dm−crypt volumes is required.

Values

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

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−256'.

cipher−mode: QCryptoCipherMode (optional)
the cipher mode for data encryption Currently defaults to 'xts'

ivgen−alg: QCryptoIVGenAlgorithm (optional)
the initialization vector generator Currently defaults to 'plain64'

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−alg: QCryptoCipherAlgorithm
the cipher algorithm for data encryption

cipher−mode: QCryptoCipherMode
the cipher mode for data encryption

ivgen−alg: QCryptoIVGenAlgorithm
the initialization vector generator

ivgen−hash−alg: QCryptoHashAlgorithm (optional)
the initialization vector generator hash

hash−alg: 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

QCryptoBlockInfo (Object)
Information about the block encryption options

Members
The members of QCryptoBlockInfoBase

The members of QCryptoBlockInfoLUKS when format is "luks"

Since
2.7

QCryptoBlockLUKSKeyslotState (Enum)
Defines state of keyslots that are affected by the update

Values
active The slots contain the given password and marked as active

inactive
The slots are erased (contain garbage) and marked as inactive

Since
5.1

QCryptoBlockAmendOptionsLUKS (Object)
This struct defines the update parameters that activate/de−activate set of keyslots

Members
state: QCryptoBlockLUKSKeyslotState
the desired state of the keyslots

new−secret: string (optional)
The ID of a QCryptoSecret object providing the password to be written into added active keyslots

old−secret: string (optional)
Optional (for deactivation only) If given will deactivate all keyslots that match password located in QCryptoSecret with this ID

iter−time: int (optional)
Optional (for activation only) Number of milliseconds to spend in PBKDF passphrase processing for the newly activated keyslot. Currently defaults to 2000.

keyslot: int (optional)
Optional. ID of the keyslot to activate/deactivate. For keyslot activation, keyslot should not be active already (this is unsafe to update an active keyslot), but possible if 'force' parameter is given. If keyslot is not given, first free keyslot will be written.

For keyslot deactivation, this parameter specifies the exact keyslot to deactivate

secret: string (optional)
Optional. The ID of a QCryptoSecret object providing the password to use to retrieve current master key. Defaults to the same secret that was used to open the image

Since 5.1

QCryptoBlockAmendOptions (Object)

The options that are available for all encryption formats when amending encryption settings

Members

The members of QCryptoBlockOptionsBase

The members of QCryptoBlockAmendOptionsLUKS when format is "luks"

Since

5.1

SecretCommonProperties (Object)

Properties for objects of classes derived from secret−common.

Members

loaded: boolean (optional)
if true, the secret is loaded immediately when applying this option and will probably fail when processing the next option. Don't use; only provided for compatibility. (default: false)

format: QCryptoSecretFormat (optional)
the data format that the secret is provided in (default: raw)

keyid: string (optional)
the name of another secret that should be used to decrypt the provided data. If not present, the data is assumed to be unencrypted.

iv: string (optional)
the random initialization vector used for encryption of this particular secret. Should be a base64 encrypted string of the 16−byte IV. Mandatory if **keyid** is given. Ignored if **keyid** is absent.

Features

deprecated
Member **loaded** is deprecated. Setting true doesn't make sense, and false is already the default.

Since

2.6

SecretProperties (Object)

Properties for secret objects.

Either **data** or **file** must be provided, but not both.

Members

data: string (optional)
the associated with the secret from

file: string (optional)
the filename to load the data associated with the secret from

The members of SecretCommonProperties

Since

2.6

SecretKeyringProperties (Object)

Properties for secret_keyring objects.

Members

serial: int
serial number that identifies a key to get from the kernel

The members of SecretCommonProperties

Since
5.1

TlsCredsProperties (Object)
Properties for objects of classes derived from tls−creds.

Members

verify−peer: boolean (optional)
if true the peer credentials will be verified once the handshake is completed. This is a no−op for anonymous credentials. (default: true)

dir: string (optional)
the path of the directory that contains the credential files

endpoint: QCryptoTLSCredsEndpoint (optional)
whether the QEMU network backend that uses the credentials will be acting as a client or as a server (default: client)

priority: string (optional)
a gnutls priority string as described at *https://gnutls.org/manual/html_node/Priority−Strings.html*

Since
2.5

TlsCredsAnonProperties (Object)
Properties for tls−creds−anon objects.

Members

loaded: boolean (optional)
if true, the credentials are loaded immediately when applying this option and will ignore options that are processed later. Don't use; only provided for compatibility. (default: false)

The members of TlsCredsProperties

Features

deprecated
Member **loaded** is deprecated. Setting true doesn't make sense, and false is already the default.

Since
2.5

TlsCredsPskProperties (Object)
Properties for tls−creds−psk objects.

Members

loaded: boolean (optional)
if true, the credentials are loaded immediately when applying this option and will ignore options that are processed later. Don't use; only provided for compatibility. (default: false)

username: string (optional)
the username which will be sent to the server. For clients only. If absent, "qemu" is sent and the property will read back as an empty string.

The members of TlsCredsProperties

Features

deprecated
Member **loaded** is deprecated. Setting true doesn't make sense, and false is already the default.

Since

3.0

TlsCredsX509Properties (Object)

Properties for tls−creds−x509 objects.

Members

loaded: boolean (optional)

if true, the credentials are loaded immediately when applying this option and will ignore options that are processed later. Don't use; only provided for compatibility. (default: false)

sanity−check: boolean (optional)

if true, perform some sanity checks before using the credentials (default: true)

passwordid: string (optional)

For the server−key.pem and client−key.pem files which contain sensitive private keys, it is possible to use an encrypted version by providing the **passwordid** parameter. This provides the ID of a previously created secret object containing the password for decryption.

The members of TlsCredsProperties

Features

deprecated

Member **loaded** is deprecated. Setting true doesn't make sense, and false is already the default.

Since

2.5

BLOCK DEVICES

Block core (VM unrelated)

Background jobs

JobType (Enum)

Type of a background job.

Values

commit

block commit job type, see "block−commit"

stream block stream job type, see "block−stream"

mirror drive mirror job type, see "drive−mirror"

backup

drive backup job type, see "drive−backup"

create image creation job type, see "blockdev−create" (since 3.0)

amend image options amend job type, see "x−blockdev−amend" (since 5.1)

snapshot−load

snapshot load job type, see "snapshot−load" (since 6.0)

snapshot−save

snapshot save job type, see "snapshot−save" (since 6.0)

snapshot−delete

snapshot delete job type, see "snapshot−delete" (since 6.0)

Since

1.7

JobStatus (Enum)

Indicates the present state of a given job in its lifetime.

Values

undefined
Erroneous, default state. Should not ever be visible.

created
The job has been created, but not yet started.

running
The job is currently running.

paused The job is running, but paused. The pause may be requested by either the QMP user or by internal processes.

ready The job is running, but is ready for the user to signal completion. This is used for long−running jobs like mirror that are designed to run indefinitely.

standby
The job is ready, but paused. This is nearly identical to **paused**. The job may return to**r eady** or otherwise be canceled.

waiting
The job is waiting for other jobs in the transaction to converge to the waiting state. This status will likely not be visible for the last job in a transaction.

pending
The job has finished its work, but has finalization steps that it needs to make prior to completing. These changes will require manual intervention via **job−finalize** if auto−finalize was set to false. These pending changes may still fail.

aborting
The job is in the process of being aborted, and will finish with an error. The job will afterwards report that it is **concluded**. This status may not be visible to the management process.

concluded
The job has finished all work. If auto−dismiss was set to false, the job will remain in the query list until it is dismissed via **job−dismiss**.

null The job is in the process of being dismantled. This state should not ever be visible externally.

Since
2.12

JobVerb (Enum)
Represents command verbs that can be applied to a job.

Values

cancel see **job−cancel**

pause see **job−pause**

resume see **job−resume**

set−speed
see **block−job−set−speed**

complete
see **job−complete**

dismiss
see **job−dismiss**

finalize see **job−finalize**

Since
2.12

JOB_STATUS_CHANGE (Event)

Emitted when a job transitions to a different status.

Arguments

id: string

The job identifier

status: JobStatus

The new job status

Since

3.0

job−pause (Command)

Pause an active job.

This command returns immediately after marking the active job for pausing. Pausing an already paused job is an error.

The job will pause as soon as possible, which means transitioning into the PAUSED state if it was RUNNING, or into STANDBY if it was READY. The corresponding JOB_STATUS_CHANGE event will be emitted.

Cancelling a paused job automatically resumes it.

Arguments

id: string

The job identifier.

Since

3.0

job−resume (Command)

Resume a paused job.

This command returns immediately after resuming a paused job. Resuming an already running job is an error.

id : The job identifier.

Arguments

id: string

Not documented

Since

3.0

job−cancel (Command)

Instruct an active background job to cancel at the next opportunity. This command returns immediately after marking the active job for cancellation.

The job will cancel as soon as possible and then emit a JOB_STATUS_CHANGE event. Usually, the status will change to ABORTING, but it is possible that a job successfully completes (e.g. because it was almost done and there was no opportunity to cancel earlier than completing the job) and transitions to PENDING instead.

Arguments

id: string

The job identifier.

Since
3.0

job−complete (Command)
Manually trigger completion of an active job in the READY state.

Arguments
id: string
The job identifier.

Since
3.0

job−dismiss (Command)
Deletes a job that is in the CONCLUDED state. This command only needs to be run explicitly for jobs that don't have automatic dismiss enabled.

This command will refuse to operate on any job that has not yet reached its terminal state, JOB_STATUS_CONCLUDED. For jobs that make use of JOB_READY event, job−cancel or job−complete will still need to be used as appropriate.

Arguments
id: string
The job identifier.

Since
3.0

job−finalize (Command)
Instructs all jobs in a transaction (or a single job if it is not part of any transaction) to finalize any graph changes and do any necessary cleanup. This command requires that all involved jobs are in the PENDING state.

For jobs in a transaction, instructing one job to finalize will force ALL jobs in the transaction to finalize, so it is only necessary to instruct a single member job to finalize.

Arguments
id: string
The identifier of any job in the transaction, or of a job that is not part of any transaction.

Since
3.0

JobInfo (Object)
Information about a job.

Members
id: string
The job identifier

type: JobType
The kind of job that is being performed

status: JobStatus
Current job state/status

current−progress: int
Progress made until now. The unit is arbitrary and the value can only meaningfully be used for the ratio of **current−progress** to **total−progress**. The value is monotonically increasing.

total−progress: int
Estimated **current−progress** value at the completion of the job. This value can arbitrarily change while the job is running, in both directions.

error: string (optional)
If this field is present, the job failed; if it is still missing in the CONCLUDED state, this indicates successful completion.

The value is a human−readable error message to describe the reason for the job failure. It should not be parsed by applications.

Since
3.0

query−jobs (Command)
Return information about jobs.

Returns
a list with a **JobInfo** for each active job

Since
3.0

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

icount: int (optional)
Current instruction count. Appears when execution record/replay is enabled. Used for "time−traveling" to match the moment in the recorded execution with the snapshots. This counter may be obtained through **query−replay** command (since 5.2)

Since
1.3

ImageInfoSpecificQCow2EncryptionBase (Object)
Members

format: BlockdevQcow2EncryptionFormat
The encryption format

Since
2.10

ImageInfoSpecificQCow2Encryption (Object)
Members

The members of ImageInfoSpecificQCow2EncryptionBase

The members of QCryptoBlockInfoLUKS when format is "luks"

Since
2.10

ImageInfoSpecificQCow2 (Object)
Members

compat: string
compatibility level

data−file: string (optional)
the filename of the external data file that is stored in the image and used as a default for opening the image (since: 4.0)

data−file−raw: boolean (optional)
True if the external data file must stay valid as a standalone (read−only) raw image without looking at qcow2 metadata (since: 4.0)

extended−l2: boolean (optional)
true if the image has extended L2 entries; only valid for compat >= 1.1 (since 5.2)

lazy−refcounts: boolean (optional)
on or off; only valid for compat >= 1.1

corrupt: boolean (optional)
true if the image has been marked corrupt; only valid for compat >= 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)

bitmaps: array of Qcow2BitmapInfo (optional)
A list of qcow2 bitmap details (since 4.0)

compression−type: Qcow2CompressionType
the image cluster compression method (since 5.1)

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

ImageInfoSpecificRbd (Object)
Members

encryption−format: RbdImageEncryptionFormat (optional)
Image encryption format

Since
6.1

ImageInfoSpecificKind (Enum)

Values

luks	Since 2.7
rbd	Since 6.1
qcow2	Not documented
vmdk	Not documented

Since
1.7

ImageInfoSpecificQCow2Wrapper (Object)

Members

data: ImageInfoSpecificQCow2
Not documented

Since
1.7

ImageInfoSpecificVmdkWrapper (Object)

Members

data: ImageInfoSpecificVmdk
Not documented

Since
6.1

ImageInfoSpecificLUKSWrapper (Object)

Members

data: QCryptoBlockInfoLUKS
Not documented

Since
2.7

ImageInfoSpecificRbdWrapper (Object)

Members

data: ImageInfoSpecificRbd
Not documented

Since
6.1

ImageInfoSpecific (Object)
A discriminated record of image format specific information structures.

Members

type: ImageInfoSpecificKind
Not documented

The members of ImageInfoSpecificQCow2Wrapper when type is "qcow2"

The members of ImageInfoSpecificVmdkWrapper when type is "vmdk"

The members of ImageInfoSpecificLUKSWrapper when type is "luks"

The members of ImageInfoSpecificRbdWrapper when type is "rbd"

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
virtual (guest) offset of the first byte described by this entry

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

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)

zero: boolean
whether the virtual blocks read as zeroes

depth: int
number of layers (0 = top image, 1 = top image's backing file, ..., n − 1 = bottom image (where n is the number of images in the chain)) before reaching one for which the range is allocated

present: boolean
true if this layer provides the data, false if adding a backing layer could impact this region (since 6.1)

offset: int (optional)
if present, the image file stores the data for this range in raw format at the given (host) offset

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 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

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)

dirty−bitmaps: array of BlockDirtyInfo (optional)
dirty bitmaps information (only present if node has one or more dirty bitmaps) (Since 4.2)

Since
0.14

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

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)

recording: boolean
true if the bitmap is recording new writes from the guest. Replaces **active** and **disabled** statuses. (since 4.0)

busy: boolean
true if the bitmap is in−use by some operation (NBD or jobs) and cannot be modified via QMP or used by another operation. Replaces **locked** and **frozen** statuses. (since 4.0)

persistent: boolean
true if the bitmap was stored on disk, is scheduled to be stored on disk, or both. (since 4.0)

inconsistent: boolean (optional)
true if this is a persistent bitmap that was improperly stored. Implies **persistent** to be true; **recording** and **busy** to be false. This bitmap cannot be used. To remove it, use **block−dirty−bitmap−remove**. (Since 4.0)

Since
1.3

Qcow2BitmapInfoFlags (Enum)
An enumeration of flags that a bitmap can report to the user.

Values

in−use This flag is set by any process actively modifying the qcow2 file, and cleared when the updated bitmap is flushed to the qcow2 image. The presence of this flag in an offline image means that the bitmap was not saved correctly after its last usage, and may contain inconsistent data.

auto The bitmap must reflect all changes of the virtual disk by any application that would write to this qcow2 file.

Since
4.0

Qcow2BitmapInfo (Object)
Qcow2 bitmap information.

Members

name: string
the name of the bitmap

granularity: int
granularity of the bitmap in bytes

flags: array of Qcow2BitmapInfoFlags
flags of the bitmap

Since
4.0

BlockLatencyHistogramInfo (Object)
Block latency histogram.

Members

boundaries: array of int
list of interval boundary values in nanoseconds, all greater than zero and in ascending order. For example, the list [10, 50, 100] produces the following histogram intervals: [0, 10), [10, 50), [50, 100), [100, +inf).

bins: array of int
list of io request counts corresponding to histogram intervals. len(**bins**) = len(**boundaries**) + 1 For the example above, **bins** may be something like [3, 1, 5, 2], and corresponding histogram looks like:

```
5|           *
4|           *
3| *         *
2| *         *    *
1| *    *    *    *
 +------------------
     10   50   100
```

Since
4.0

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)

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

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 further bitmap creation, may require additional space and is not covered here.

Members
required: int
Size required for a new image file, in bytes, when copying just allocated guest−visible contents.

fully−allocated: int
Image file size, in bytes, once data has been written to all sectors, when copying just guest−visible contents.

bitmaps: int (optional)
Additional size required if all the top−level bitmap metadata in the source image were to be copied to the destination, present only when source and destination both support persistent bitmaps. (since 5.1)

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

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.

unmap_bytes: int
The number of bytes unmapped by the device (Since 4.2)

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)

unmap_operations: int
The number of unmap operations performed by the device (Since 4.2)

rd_total_time_ns: int
Total time spent on reads in nanoseconds (since 0.15).

wr_total_time_ns: int
Total time spent on writes in nanoseconds (since 0.15).

flush_total_time_ns: int
Total time spent on cache flushes in nanoseconds (since 0.15).

unmap_total_time_ns: int
Total time spent on unmap operations in nanoseconds (Since 4.2)

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).

unmap_merged: int
Number of unmap requests that have been merged into another request (Since 4.2)

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)

failed_unmap_operations: int
The number of failed unmap operations performed by the device (Since 4.2)

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)

invalid_unmap_operations: int
The number of invalid unmap operations performed by the device (Since 4.2)

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)

rd_latency_histogram: BlockLatencyHistogramInfo (optional)
BlockLatencyHistogramInfo. (Since 4.0)

wr_latency_histogram: BlockLatencyHistogramInfo (optional)
BlockLatencyHistogramInfo. (Since 4.0)

flush_latency_histogram: BlockLatencyHistogramInfo (optional)
BlockLatencyHistogramInfo. (Since 4.0)

Since
0.14

BlockStatsSpecificFile (Object)
File driver statistics

Members

discard−nb−ok: int
The number of successful discard operations performed by the driver.

discard−nb−failed: int
The number of failed discard operations performed by the driver.

discard−bytes−ok: int
The number of bytes discarded by the driver.

Since
4.2

BlockStatsSpecificNvme (Object)
NVMe driver statistics

Members

completion−errors: int
The number of completion errors.

aligned−accesses: int
The number of aligned accesses performed by the driver.

unaligned−accesses: int
The number of unaligned accesses performed by the driver.

Since
5.2

BlockStatsSpecific (Object)
Block driver specific statistics

Members
driver: BlockdevDriver
Not documented

The members of BlockStatsSpecificFile when driver is "file"

The members of **BlockStatsSpecificFile** when **driver** is **"host_device"** (**If: HAVE_HOST_BLOCK_DEVICE**)

The members of BlockStatsSpecificNvme when driver is "nvme"

Since
4.2

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)

qdev: string (optional)
The qdev ID, or if no ID is assigned, the QOM path of the block device. (since 3.0)

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

driver−specific: BlockStatsSpecific (optional)
Optional driver−specific stats. (Since 4.2)

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

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

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
      },
      "qdev": "/machine/unattached/device[23]"
   },
   {
      "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
               },
               "qdev": "/machine/unattached/device[24]"
            },
            {
               "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
               },
               "qdev": "/machine/unattached/device[16]"
            },
            {
               "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). The backup, mirror and commit block jobs retry the failing request later and may still complete successfully. The stream block job continues to stream and will complete with an 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)

bitmap only copy data described by the dirty bitmap. (since: 4.2) Behavior on completion is determined by the BitmapSyncMode.

Since
1.3

BitmapSyncMode (Enum)
An enumeration of possible behaviors for the synchronization of a bitmap when used for data copy operations.

Values

on−success
The bitmap is only synced when the operation is successful. This is the behavior always used for 'INCREMENTAL' backups.

never The bitmap is never synchronized with the operation, and is treated solely as a read−only manifest of blocks to copy.

always The bitmap is always synchronized with the operation, regardless of whether or not the operation was successful.

Since
4.2

MirrorCopyMode (Enum)
An enumeration whose values tell the mirror block job when to trigger writes to the target.

Values

background
copy data in background only.

write−blocking
when data is written to the source, write it (synchronously) to the target as well. In addition, data is copied in background just like in **background** mode.

Since
3.0

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
Estimated **offset** value at the completion of the job. This value can arbitrarily change while the job is running, in both directions.

offset: int
Progress made until now. The unit is arbitrary and the value can only meaningfully be used for the ratio of **offset** to **len**. The value is monotonically increasing.

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.

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)

status: JobStatus
Current job state/status (since 2.12)

auto−finalize: boolean
Job will finalize itself when PENDING, moving to the CONCLUDED state. (since 2.12)

auto−dismiss: boolean
Job will dismiss itself when CONCLUDED, moving to the NULL state and disappearing from the query list. (since 2.12)

error: string (optional)
Error information if the job did not complete successfully. Not set if the job completed successfully. (since 2.12.1)

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_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

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 take a snapshot of.

node−name: string (optional)
graph node name to generate the snapshot from (Since 2.0)

snapshot−file: string
the target of the new overlay image. If the file exists, or if it is a device, the overlay will be created in the existing file/device. Otherwise, a new file will be created.

snapshot−node−name: string (optional)
the graph node name of the new image (Since 2.0)

format: string (optional)
the format of the overlay image, default is 'qcow2'.

mode: NewImageMode (optional)
whether and how QEMU should create a new image, default is 'absolute−paths'.

BlockdevSnapshot (Object)
Members

node: string
device or node name that will have a snapshot taken.

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

Since
2.5

BackupPerf (Object)

Optional parameters for backup. These parameters don't affect functionality, but may significantly affect performance.

Members

use−copy−range: boolean (optional)
Use copy offloading. Default false.

max−workers: int (optional)
Maximum number of parallel requests for the sustained background copying process. Doesn't influence copy−before−write operations. Default 64.

max−chunk: int (optional)
Maximum request length for the sustained background copying process. Doesn't influence copy−before−write operations. 0 means unlimited. If max−chunk is non−zero then it should not be less than job cluster size which is calculated as maximum of target image cluster size and 64k. Default 0.

Since

6.0

BackupCommon (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.

sync: MirrorSyncMode
what parts of the disk image should be copied to the destination (all the disk, only the sectors allocated in the topmost image, from a dirty bitmap, or only new I/O).

speed: int (optional)
the maximum speed, in bytes per second. The default is 0, for unlimited.

bitmap: string (optional)
The name of a dirty bitmap to use. Must be present if sync is "bitmap" or "incremental". Can be present if sync is "full" or "top". Must not be present otherwise. (Since 2.4 (drive−backup), 3.1 (blockdev−backup))

bitmap−mode: BitmapSyncMode (optional)
Specifies the type of data the bitmap should contain after the operation concludes. Must be present if a bitmap was provided, Must NOT be present otherwise. (Since 4.2)

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**).

auto−finalize: boolean (optional)
When false, this job will wait in a PENDING state after it has finished its work, waiting for **block−job−finalize** before making any block graph changes. When true, this job will automatically perform its abort or commit actions. Defaults to true. (Since 2.12)

auto−dismiss: boolean (optional)
When false, this job will wait in a CONCLUDED state after it has completely ceased all work, and awaits **block−job−dismiss**. When true, this job will automatically disappear from the query list without user intervention. Defaults to true. (Since 2.12)

filter−node−name: string (optional)
the node name that should be assigned to the filter driver that the backup job inserts into the graph above node specified by **drive**. If this option is not given, a node name is autogenerated. (Since: 4.2)

x−perf: BackupPerf (optional)
Performance options. (Since 6.0)

Features

unstable
Member **x−perf** is experimental.

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

4.2

DriveBackup (Object)

Members

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

mode: NewImageMode (optional)
whether and how QEMU should create a new image, default is 'absolute−paths'.

The members of BackupCommon

Since

1.6

BlockdevBackup (Object)

Members

target: string
the device name or node−name of the backup target node.

The members of BackupCommon

Since

2.3

blockdev−snapshot−sync (Command)

Takes 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

Example

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

blockdev−snapshot (Command)

Takes a snapshot of a block device.

Take 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.

Features

allow−write−only−overlay
If present, the check whether this operation is safe was relaxed so that it can be used to change backing file of a destination of a blockdev−mirror. (since 5.0)

Since

2.5

Example

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

<- { "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'.

If top == base, that is an error. If top has no overlays on top of it, or if it is in use by a writer, the job will not be completed by itself. The 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.

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−node: string (optional)

The node name of the backing image to write data into. If not specified, this is the deepest backing image. (since: 3.1)

base: string (optional)

Same as **base−node**, except that it is a file name rather than a node name. This must be the exact filename string that was used to open the node; other strings, even if addressing the same file, are not accepted

top−node: string (optional)

The node 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. (since: 3.1)

top: string (optional)

Same as **top−node**, except that it is a file name rather than a node name. This must be the exact filename string that was used to open the node; other strings, even if addressing the same file, are not accepted

backing−file: string (optional)

The backing file string to write into the overlay image of 'top'. If 'top' does not have an overlay image, or if 'top' is in use by a writer, 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)

speed: int (optional)

the maximum speed, in bytes per second

on−error: BlockdevOnError (optional)

the action to take on an error. 'ignore' means that the request should be retried. (default: report; Since: 5.0)

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)

auto−finalize: boolean (optional)
When false, this job will wait in a PENDING state after it has finished its work, waiting for **block−job−finalize** before making any block graph changes. When true, this job will automatically perform its abort or commit actions. Defaults to true. (Since 3.1)

auto−dismiss: boolean (optional)
When false, this job will wait in a CONCLUDED state after it has completely ceased all work, and awaits **block−job−dismiss**. When true, this job will automatically disappear from the query list without user intervention. Defaults to true. (Since 3.1)

Features

deprecated
Members **base** and **top** are deprecated. Use **base−node** and **top−node** instead.

Returns

- Nothing on success
- If **device** does not exist, DeviceNotFound
- Any other error returns a GenericError.

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

Features

deprecated
This command is deprecated. Use **blockdev−backup** instead.

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

Arguments

flat: boolean (optional)

Omit the nested data about backing image ("backing−image" key) if true. Default is false (Since 5.0)

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
                      }
                   } } ] }
```

XDbgBlockGraphNodeType (Enum)

Values

block−backend
corresponds to BlockBackend

block−job
corresponds to BlockJob

block−driver
corresponds to BlockDriverState

Since
4.0

XDbgBlockGraphNode (Object)

Members

id: int Block graph node identifier. This **id** is generated only for x−debug−query−block−graph and does not relate to any other identifiers in Qemu.

type: XDbgBlockGraphNodeType
Type of graph node. Can be one of block−backend, block−job or block−driver−state.

name: string
Human readable name of the node. Corresponds to node−name for block−driver−state nodes; is not guaranteed to be unique in the whole graph (with block−jobs and block−backends).

Since
4.0

BlockPermission (Enum)

Enum of base block permissions.

Values

consistent−read
A user that has the "permission" of consistent reads is guaranteed that their view of the contents of the block device is complete and self−consistent, representing the contents of a disk at a specific point. For most block devices (including their backing files) this is true, but the property cannot be maintained in a few situations like for intermediate nodes of a commit block job.

write This permission is required to change the visible disk contents.

write−unchanged
This permission (which is weaker than BLK_PERM_WRITE) is both enough and required for writes to the block node when the caller promises that the visible disk content doesn't change. As the BLK_PERM_WRITE permission is strictly stronger, either is sufficient to perform an unchanging write.

resize This permission is required to change the size of a block node.

graph−mod
This permission is required to change the node that this BdrvChild points to.

Since
4.0

XDbgBlockGraphEdge (Object)
Block Graph edge description for x−debug−query−block−graph.

Members

parent: int
parent id

child: int
child id

name: string
name of the relation (examples are 'file' and 'backing')

perm: array of BlockPermission
granted permissions for the parent operating on the child

shared−perm: array of BlockPermission
permissions that can still be granted to other users of the child while it is still attached to this parent

Since
4.0

XDbgBlockGraph (Object)
Block Graph − list of nodes and list of edges.

Members

nodes: array of XDbgBlockGraphNode
Not documented

edges: array of XDbgBlockGraphEdge
Not documented

Since
4.0

x−debug−query−block−graph (Command)
Get the block graph.

Features

unstable
This command is meant for debugging.

Since
4.0

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. By default, **device** is replaced, although implicitly created filters on it are kept. (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)

copy−mode: MirrorCopyMode (optional)
when to copy data to the destination; defaults to 'background' (Since: 3.0)

auto−finalize: boolean (optional)
When false, this job will wait in a PENDING state after it has finished its work, waiting for **block−job−finalize** before making any block graph changes. When true, this job will automatically perform its abort or commit actions. Defaults to true. (Since 3.1)

auto−dismiss: boolean (optional)
When false, this job will wait in a CONCLUDED state after it has completely ceased all work, and awaits **block−job−dismiss**. When true, this job will automatically disappear from the query list without user intervention. Defaults to true. (Since 3.1)

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 (must be less than 1024 bytes)

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)

disabled: boolean (optional)
the bitmap is created in the disabled state, which means that it will not track drive changes. The bitmap may be enabled with block−dirty−bitmap−enable. Default is false. (Since: 4.0)

Since
2.4

BlockDirtyBitmapMergeSource (Alternate)

Members

local: string
name of the bitmap, attached to the same node as target bitmap.

external: BlockDirtyBitmap
bitmap with specified node

Since
4.1

BlockDirtyBitmapMerge (Object)

Members

node: string
name of device/node which the **target** bitmap is tracking

target: string
name of the destination dirty bitmap

bitmaps: array of BlockDirtyBitmapMergeSource
name(s) of the source dirty bitmap(s) at **node** and/or fully specified BlockDirtyBitmap elements. The latter are supported since 4.1.

Since
4.0

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": {} }
```

block−dirty−bitmap−enable (Command)

Enables a dirty bitmap so that it will begin tracking disk changes.

Returns

- nothing on success
- If **node** is not a valid block device, DeviceNotFound
- If **name** is not found, GenericError with an explanation

Since

4.0

Example

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

block−dirty−bitmap−disable (Command)

Disables a dirty bitmap so that it will stop tracking disk changes.

Returns

- nothing on success
- If **node** is not a valid block device, DeviceNotFound
- If **name** is not found, GenericError with an explanation

Since

4.0

Example

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

block−dirty−bitmap−merge (Command)

Merge dirty bitmaps listed in **bitmaps** to the **target** dirty bitmap. Dirty bitmaps in **bitmaps** will be unchanged, except if it also appears as the **target** bitmap. Any bits already set in **target** will still be set after the merge, i.e., this operation does not clear the target. On error, **target** is unchanged.

The resulting bitmap will count as dirty any clusters that were dirty in any of the source bitmaps. This can be used to achieve backup checkpoints, or in simpler usages, to copy bitmaps.

Returns

- nothing on success

- If **node** is not a valid block device, DeviceNotFound
- If any bitmap in **bitmaps** or **target** is not found, GenericError
- If any of the bitmaps have different sizes or granularities, GenericError

Since
4.0

Example
```
-> { "execute": "block-dirty-bitmap-merge",
     "arguments": { "node": "drive0", "target": "bitmap0",
                    "bitmaps": ["bitmap1"] } }
<- { "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.

Features
unstable
This command is meant for debugging.

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. By default, **device** is replaced, although implicitly created filters on it are kept.

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)

copy−mode: MirrorCopyMode (optional)
when to copy data to the destination; defaults to 'background' (Since: 3.0)

auto−finalize: boolean (optional)
When false, this job will wait in a PENDING state after it has finished its work, waiting for **block−job−finalize** before making any block graph changes. When true, this job will automatically perform its abort or commit actions. Defaults to true. (Since 3.1)

auto−dismiss: boolean (optional)
When false, this job will wait in a CONCLUDED state after it has completely ceased all work, and awaits **block−job−dismiss**. When true, this job will automatically disappear from the query list without user intervention. Defaults to true. (Since 3.1)

Returns
nothing on success.

Since
2.6

Example

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

BlockIOThrottle (Object)
A set of parameters describing block throttling.

Members

device: string (optional)
Block device name

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)

Features

deprecated
Member **device** is deprecated. Use **id** instead.

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

ThrottleGroupProperties (Object)
Properties for throttle−group objects.

Members

limits: ThrottleLimits (optional)
limits to apply for this throttle group

x−iops−total: int (optional)
Not documented

x−iops−total−max: int (optional)
Not documented

x−iops−total−max−length: int (optional)
Not documented

x−iops−read: int (optional)
Not documented

x−iops−read−max: int (optional)
Not documented

x−iops−read−max−length: int (optional)
Not documented

x−iops−write: int (optional)
Not documented

x−iops−write−max: int (optional)
Not documented

x−iops−write−max−length: int (optional)
Not documented

x−bps−total: int (optional)
Not documented

x−bps−total−max: int (optional)
Not documented

x−bps−total−max−length: int (optional)
Not documented

x−bps−read: int (optional)
Not documented

x−bps−read−max: int (optional)
Not documented

x−bps−read−max−length: int (optional)
Not documented

x−bps−write: int (optional)
Not documented

x−bps−write−max: int (optional)
Not documented

x−bps−write−max−length: int (optional)
Not documented

x−iops−size: int (optional)
Not documented

Features

unstable
All members starting with x− are aliases for the same key without x− in the **limits** object. This is not a stable interface and may be removed or changed incompatibly in the future. Use **limits** for a supported stable interface.

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. This can be used to stream a subset of the backing file chain instead of flattening the entire image. When streaming completes the image file will have the base file as its backing file, unless that node was changed while the job was running. In that case, base's parent's backing (or filtered, whichever exists) child (i.e., base at the beginning of the job) will be the new backing file.

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

In case **device** is a filter node, block−stream modifies the first non−filter overlay node below it to point to the new backing node instead of modifying **device** itself.

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** or **bottom** is also set.

base−node: string (optional)
the node name of the backing file. It cannot be set if **base** or **bottom** is also set. (Since 2.8)

bottom: string (optional)
the last node in the chain that should be streamed into top. It cannot be set if **base** or **base−node** is also set. It cannot be filter node. (Since 6.0)

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.

filter−node−name: string (optional)
the node name that should be assigned to the filter driver that the stream job inserts into the graph above **device**. If this option is not given, a node name is autogenerated. (Since: 6.0)

auto−finalize: boolean (optional)
When false, this job will wait in a PENDING state after it has finished its work, waiting for **block−job−finalize** before making any block graph changes. When true, this job will automatically perform its abort or commit actions. Defaults to true. (Since 3.1)

auto−dismiss: boolean (optional)
When false, this job will wait in a CONCLUDED state after it has completely ceased all work, and awaits **block−job−dismiss**. When true, this job will automatically disappear from the query list without user intervention. Defaults to true. (Since 3.1)

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.

Note that if you issue 'block−job−cancel' after 'drive−mirror' has indicated (via the event BLOCK_JOB_READY) that the source and destination are synchronized, then the event triggered by this command changes to BLOCK_JOB_COMPLETED, to indicate that the mirroring has ended and the destination now has a point−in−time copy tied to the time of the cancellation.

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)

If true, and the job has already emitted the event BLOCK_JOB_READY, abandon the job immediately (even if it is paused) instead of waiting for the destination to complete its final synchronization (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 or if the job is already paused.

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 or if the job is not paused.

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 rerror/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

block−job−dismiss (Command)

For jobs that have already concluded, remove them from the block−job−query list. This command only needs to be run for jobs which were started with QEMU 2.12+ job lifetime management semantics.

This command will refuse to operate on any job that has not yet reached its terminal state, JOB_STATUS_CONCLUDED. For jobs that make use of the BLOCK_JOB_READY event, block−job−cancel or block−job−complete will still need to be used as appropriate.

Arguments

id: string

The job identifier.

Returns

Nothing on success

Since

2.12

block−job−finalize (Command)

Once a job that has manual=true reaches the pending state, it can be instructed to finalize any graph changes and do any necessary cleanup via this command. For jobs in a transaction, instructing one job to finalize will force ALL jobs in the transaction to finalize, so it is only necessary to instruct a single member job to finalize.

Arguments

id: string
The job identifier.

Returns

Nothing on success

Since

2.12

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)

io_uring (**If: CONFIG_LINUX_IO_URING**)
Use linux io_uring (since 5.0)

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

throttle
Since 2.11

nvme Since 2.12

copy−on−read
Since 3.0

blklogwrites
Since 3.0

blkreplay
Since 4.2

compress
Since 5.0

copy−before−write
Since 6.2

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 (**If: HAVE_HOST_BLOCK_DEVICE**)
Not documented

host_device (**If: HAVE_HOST_BLOCK_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

preallocate
Not documented

qcow Not documented

qcow2 Not documented

qed Not documented

quorum
Not documented

raw Not documented

rbd Not documented

replication (**If: CONFIG_REPLICATION**)
Not documented

ssh Not documented

vdi Not documented

vhdx Not documented

vmdk Not documented

vpc Not documented

vvfat Not documented

Since
2.9

BlockdevOptionsFile (Object)
Driver specific block device options for the file backend.

Members

filename: string
path to the image file

pr−manager: string (optional)
the id for the object that will handle persistent reservations for this device (default: none, forward the commands via SG_IO; since 2.11)

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

aio−max−batch: int (optional)
maximum number of requests to batch together into a single submission in the AIO backend. The smallest value between this and the aio−max−batch value of the IOThread object is chosen. 0 means that the AIO backend will handle it automatically. (default: 0, since 6.2)

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)

drop−cache: **boolean** (optional) (**If: CONFIG_LINUX**)
invalidate page cache during live migration. This prevents stale data on the migration destination with cache.direct=off. Currently only supported on Linux hosts. (default: on, since: 4.0)

x−check−cache−dropped: boolean (optional)
whether to check that page cache was dropped on live migration. May cause noticeable delays if the image file is large, do not use in production. (default: off) (since: 3.0)

Features

dynamic−auto−read−only
If present, enabled auto−read−only means that the driver will open the image read−only at first, dynamically reopen the image file read−write when the first writer is attached to the node and reopen read−only when the last writer is detached. This allows giving QEMU write permissions only on demand when an operation actually needs write access.

unstable
Member x−check−cache−dropped is meant for debugging.

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)

read−zeroes: boolean (optional)
if true, reads from the device produce zeroes; if false, the buffer is left unchanged. (default: false; since: 4.1)

Since
2.9

BlockdevOptionsNVMe (Object)
Driver specific block device options for the NVMe backend.

Members

device: string
PCI controller address of the NVMe device in format hhhh:bb:ss.f (host:bus:slot.function)

namespace: int
namespace number of the device, starting from 1.

Note that the PCI **device** must have been unbound from any host kernel driver before instructing QEMU to add the blockdev.

Since
2.12

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'

bitmap−directory: boolean (optional)
since 3.0

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 vectors
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

BlockdevOptionsPreallocate (Object)

Filter driver intended to be inserted between format and protocol node and do preallocation in protocol node on write.

Members

prealloc−align: int (optional)

on preallocation, align file length to this number, default 1048576 (1M)

prealloc−size: int (optional)

how much to preallocate, default 134217728 (128M)

The members of BlockdevOptionsGenericFormat

Since

6.0

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)

l2−cache−entry−size: int (optional)
the size of each entry in the L2 cache in bytes. It must be a power of two between 512 and the cluster size. The default value is the cluster size (since 2.12)

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 600 on supporting platforms, and 0 on other platforms. 0 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)

data−file: BlockdevRef (optional)
reference to or definition of the external data file. This may only be specified for images that require an external data file. If it is not specified for such an image, the data file name is loaded from the image file. (since 4.0)

The members of BlockdevOptionsGenericCOWFormat

Since
2.9

SshHostKeyCheckMode (Enum)
Values

none Don't check the host key at all

hash Compare the host key with a given hash

known_hosts
Check the host key against the known_hosts file

Since
2.12

SshHostKeyCheckHashType (Enum)
Values

md5 The given hash is an md5 hash

sha1 The given hash is an sha1 hash

sha256 The given hash is an sha256 hash

Since
2.12

SshHostKeyHash (Object)
Members

type: SshHostKeyCheckHashType
The hash algorithm used for the hash

hash: string
The expected hash value

Since
2.12

SshHostKeyCheck (Object)
Members

mode: SshHostKeyCheckMode
Not documented

The members of SshHostKeyHash when mode is "hash"

Since
2.12

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

host−key−check: SshHostKeyCheck (optional)
Defines how and what to check the host key against (default: known_hosts)

Since
2.9

BlkdebugEvent (Enum)
Trigger events supported by blkdebug.

Values

l1_shrink_write_table
write zeros to the l1 table to shrink image. (since 2.11)

l1_shrink_free_l2_clusters
discard the l2 tables. (since 2.11)

cor_write
a write due to copy−on−read (since 2.11)

cluster_alloc_space
an allocation of file space for a cluster (since 4.1)

none triggers once at creation of the blkdebug node (since 4.1)

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

BlkdebugIOType (Enum)
Kinds of I/O that blkdebug can inject errors in.

Values

read .bdrv_co_preadv()

write .bdrv_co_pwritev()

write−zeroes
.bdrv_co_pwrite_zeroes()

discard
.bdrv_co_pdiscard()

flush .bdrv_co_flush_to_disk()

block−status
.bdrv_co_block_status()

Since
4.1

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"

iotype: BlkdebugIOType (optional)
the type of I/O operations on which this error should be injected; defaults to "all read, write, write−zeroes, discard, and flush operations" (since: 4.1)

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

take−child−perms: array of BlockPermission (optional)
Permissions to take on **image** in addition to what is necessary anyway (which depends on how the blkdebug node is used). Defaults to none. (since 5.0)

unshare−child−perms: array of BlockPermission (optional)
Permissions not to share on **image** in addition to what cannot be shared anyway (which depends on how the blkdebug node is used). Defaults to none. (since 5.0)

Since
2.9

BlockdevOptionsBlklogwrites (Object)
Driver specific block device options for blklogwrites.

Members

file: BlockdevRef
block device

log: BlockdevRef
block device used to log writes to **file**

log−sector−size: int (optional)
sector size used in logging writes to **file**, determines granularity of offsets and sizes of writes (default: 512)

log−append: boolean (optional)
append to an existing log (default: false)

log−super−update−interval: int (optional)
interval of write requests after which the log super block is updated to disk (default: 4096)

Since
3.0

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

BlockdevOptionsBlkreplay (Object)
Driver specific block device options for blkreplay.

Members

image: BlockdevRef
disk image which should be controlled with blkreplay

Since
4.2

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

RbdAuthMode (Enum)
Values

cephx	Not documented
none	Not documented

Since
3.0

RbdImageEncryptionFormat (Enum)
Values

luks	Not documented
luks2	Not documented

Since
6.1

RbdEncryptionOptionsLUKSBase (Object)
Members

key−secret: string
ID of a QCryptoSecret object providing a passphrase for unlocking the encryption

Since
6.1

RbdEncryptionCreateOptionsLUKSBase (Object)
Members

cipher−alg: QCryptoCipherAlgorithm (optional)
The encryption algorithm

The members of RbdEncryptionOptionsLUKSBase

Since
6.1

RbdEncryptionOptionsLUKS (Object)
Members

The members of RbdEncryptionOptionsLUKSBase

Since
6.1

RbdEncryptionOptionsLUKS2 (Object)
Members

The members of RbdEncryptionOptionsLUKSBase

Since
6.1

RbdEncryptionCreateOptionsLUKS (Object)
Members

The members of RbdEncryptionCreateOptionsLUKSBase

Since
6.1

RbdEncryptionCreateOptionsLUKS2 (Object)
Members
The members of RbdEncryptionCreateOptionsLUKSBase

Since
6.1

RbdEncryptionOptions (Object)
Members
format: RbdImageEncryptionFormat
Not documented

The members of RbdEncryptionOptionsLUKS when format is "luks"

The members of RbdEncryptionOptionsLUKS2 when format is "luks2"

Since
6.1

RbdEncryptionCreateOptions (Object)
Members
format: RbdImageEncryptionFormat
Not documented

The members of RbdEncryptionCreateOptionsLUKS when format is "luks"

The members of RbdEncryptionCreateOptionsLUKS2 when format is "luks2"

Since
6.1

BlockdevOptionsRbd (Object)
Members
pool: string
Ceph pool name.

namespace: string (optional)
Rados namespace name in the Ceph pool. (Since 5.0)

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.

encrypt: RbdEncryptionOptions (optional)
Image encryption options. (Since 6.1)

user: string (optional)
Ceph id name.

auth−client−required: array of RbdAuthMode (optional)
Acceptable authentication modes. This maps to Ceph configuration option "auth_client_required". (Since 3.0)

key−secret: string (optional)
ID of a QCryptoSecret object providing a key for cephx authentication. This maps to Ceph configuration option "key". (Since 3.0)

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

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

If
CONFIG_REPLICATION

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

If
CONFIG_REPLICATION

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

x−dirty−bitmap: string (optional)
A metadata context name such as "qemu:dirty−bitmap:NAME" or "qemu:allocation−depth" to query in place of the traditional "base:allocation" block status (see NBD_OPT_LIST_META_CONTEXT in the NBD protocol; and yes, naming this option x−context would have made more sense) (since 3.0)

reconnect−delay: int (optional)
On an unexpected disconnect, the nbd client tries to connect again until succeeding or encountering a serious error. During the first**r econnect−delay** seconds, all requests are paused and will be rerun on a successful reconnect. After that time, any delayed requests and all future requests before a successful reconnect will immediately fail. Default 0 (Since 4.2)

Features
unstable
Member **x−dirty−bitmap** is experimental.

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

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

BlockdevOptionsCor (Object)
Driver specific block device options for the copy−on−read driver.

Members
bottom: string (optional)
The name of a non−filter node (allocation−bearing layer) that limits the COR operations in the backing chain (inclusive), so that no data below this node will be copied by this filter. If option is absent, the limit is not applied, so that data from all backing layers may be copied.

The members of BlockdevOptionsGenericFormat

Since
6.0

BlockdevOptionsCbw (Object)
Driver specific block device options for the copy−before−write driver, which does so called copy−before−write operations: when data is written to the filter, the filter first reads corresponding blocks from its file child and copies them to **target** child. After successfully copying, the write request is propagated to file

child. If copying fails, the original write request is failed too and no data is written to file child.

Members

target: BlockdevRef
The target for copy−before−write operations.

The members of BlockdevOptionsGenericFormat

Since
6.2

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. Valid node names start with an alphabetic character and may contain only alphanumeric characters, '−', '.' and '_'. Their maximum length is 31 characters.

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). Note that some block drivers support only read−only access, either generally or in certain configurations. In this case, the default value does not work and the option must be specified explicitly.

auto−read−only: boolean (optional)
if true and **read−only** is false, QEMU may automatically decide not to open the image read−write as requested, but fall back to read−only instead (and switch between the modes later), e.g. depending on whether the image file is writable or whether a writing user is attached to the node (default: false, since 3.1)

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 BlockdevOptionsBlklogwrites when driver is "blklogwrites"

The members of BlockdevOptionsBlkverify when driver is "blkverify"

The members of BlockdevOptionsBlkreplay when driver is "blkreplay"

The members of BlockdevOptionsGenericFormat when driver is "bochs"

The members of BlockdevOptionsGenericFormat when driver is "cloop"

The members of BlockdevOptionsGenericFormat when driver is "compress"

The members of BlockdevOptionsCbw when driver is "copy−before−write"

The members of BlockdevOptionsCor when driver is "copy−on−read"

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"** (**If: HAVE_HOST_BLOCK_DEVICE**)

The members of **BlockdevOptionsFile** when **driver** is **"host_device"** (**If: HAVE_HOST_BLOCK_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 BlockdevOptionsNVMe when driver is "nvme"

The members of BlockdevOptionsGenericFormat when driver is "parallels"

The members of BlockdevOptionsPreallocate when driver is "preallocate"

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"** (**If: CONFIG_REPLICATION**)

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"
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.

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−reopen (Command)

Reopens one or more block devices using the given set of options. Any option not specified will be reset to its default value regardless of its previous status. If an option cannot be changed or a particular driver does not support reopening then the command will return an error. All devices in the list are reopened in one transaction, so if one of them fails then the whole transaction is cancelled.

The command receives a list of block devices to reopen. For each one of them, the top−level **node−name** option (from BlockdevOptions) must be specified and is used to select the block device to be reopened. Other **node−name** options must be either omitted or set to the current name of the appropriate node. This command won't change any node name and any attempt to do it will result in an error.

In the case of options that refer to child nodes, the behavior of this command depends on the value:

1. A set of options (BlockdevOptions): the child is reopened with the specified set of options.
2. A reference to the current child: the child is reopened using its existing set of options.
3. A reference to a different node: the current child is replaced with the specified one.
4. NULL: the current child (if any) is detached.

Options (1) and (2) are supported in all cases. Option (3) is supported for **file** and **backing**, and option (4) for **backing** only.

Unlike with blockdev−add, the **backing** option must always be present unless the node being reopened does not have a backing file and its image does not have a default backing file name as part of its metadata.

Arguments

options: array of BlockdevOptions

Not documented

Since

6.1

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": {} }
```

BlockdevCreateOptionsFile (Object)

Driver specific image creation options for file.

Members

filename: string
Filename for the new image file

size: int
Size of the virtual disk in bytes

preallocation: PreallocMode (optional)
Preallocation mode for the new image (default: off; allowed values: off, falloc (if CONFIG_POSIX_FALLOCATE), full (if CONFIG_POSIX))

nocow: boolean (optional)
Turn off copy−on−write (valid only on btrfs; default: off)

extent−size−hint: int (optional)
Extent size hint to add to the image file; 0 for not adding an extent size hint (default: 1 MB, since 5.1)

Since
2.12

BlockdevCreateOptionsGluster (Object)

Driver specific image creation options for gluster.

Members

location: BlockdevOptionsGluster
Where to store the new image file

size: int
Size of the virtual disk in bytes

preallocation: PreallocMode (optional)
Preallocation mode for the new image (default: off; allowed values: off, falloc (if CONFIG_GLUSTERFS_FALLOCATE), full (if CONFIG_GLUSTERFS_ZEROFILL))

Since
2.12

BlockdevCreateOptionsLUKS (Object)

Driver specific image creation options for LUKS.

Members

file: BlockdevRef
Node to create the image format on

size: int
Size of the virtual disk in bytes

preallocation: PreallocMode (optional)
Preallocation mode for the new image (since: 4.2) (default: off; allowed values: off, metadata, falloc, full)

The members of QCryptoBlockCreateOptionsLUKS

Since
2.12

BlockdevCreateOptionsNfs (Object)
Driver specific image creation options for NFS.

Members

location: BlockdevOptionsNfs
Where to store the new image file

size: int
Size of the virtual disk in bytes

Since
2.12

BlockdevCreateOptionsParallels (Object)
Driver specific image creation options for parallels.

Members

file: BlockdevRef
Node to create the image format on

size: int
Size of the virtual disk in bytes

cluster−size: int (optional)
Cluster size in bytes (default: 1 MB)

Since
2.12

BlockdevCreateOptionsQcow (Object)
Driver specific image creation options for qcow.

Members

file: BlockdevRef
Node to create the image format on

size: int
Size of the virtual disk in bytes

backing−file: string (optional)
File name of the backing file if a backing file should be used

encrypt: QCryptoBlockCreateOptions (optional)
Encryption options if the image should be encrypted

Since
2.12

BlockdevQcow2Version (Enum)

Values

v2 The original QCOW2 format as introduced in qemu 0.10 (version 2)

v3 The extended QCOW2 format as introduced in qemu 1.1 (version 3)

Since

2.12

Qcow2CompressionType (Enum)

Compression type used in qcow2 image file

Values

zlib zlib compression, see <*http://zlib.net/*>

zstd (**If: CONFIG_ZSTD**)
zstd compression, see <*http://github.com/facebook/zstd*>

Since

5.1

BlockdevCreateOptionsQcow2 (Object)

Driver specific image creation options for qcow2.

Members

file: BlockdevRef
Node to create the image format on

data−file: BlockdevRef (optional)
Node to use as an external data file in which all guest data is stored so that only metadata remains in the qcow2 file (since: 4.0)

data−file−raw: boolean (optional)
True if the external data file must stay valid as a standalone (read−only) raw image without looking at qcow2 metadata (default: false; since: 4.0)

extended−l2: boolean (optional)
True to make the image have extended L2 entries (default: false; since 5.2)

size: int
Size of the virtual disk in bytes

version: BlockdevQcow2Version (optional)
Compatibility level (default: v3)

backing−file: string (optional)
File name of the backing file if a backing file should be used

backing−fmt: BlockdevDriver (optional)
Name of the block driver to use for the backing file

encrypt: QCryptoBlockCreateOptions (optional)
Encryption options if the image should be encrypted

cluster−size: int (optional)
qcow2 cluster size in bytes (default: 65536)

preallocation: PreallocMode (optional)
Preallocation mode for the new image (default: off; allowed values: off, falloc, full, metadata)

lazy−refcounts: boolean (optional)
True if refcounts may be updated lazily (default: off)

refcount−bits: int (optional)
Width of reference counts in bits (default: 16)

compression−type: Qcow2CompressionType (optional)
The image cluster compression method (default: zlib, since 5.1)

Since
2.12

BlockdevCreateOptionsQed (Object)
Driver specific image creation options for qed.

Members

file: BlockdevRef
Node to create the image format on

size: int
Size of the virtual disk in bytes

backing−file: string (optional)
File name of the backing file if a backing file should be used

backing−fmt: BlockdevDriver (optional)
Name of the block driver to use for the backing file

cluster−size: int (optional)
Cluster size in bytes (default: 65536)

table−size: int (optional)
L1/L2 table size (in clusters)

Since
2.12

BlockdevCreateOptionsRbd (Object)
Driver specific image creation options for rbd/Ceph.

Members

location: BlockdevOptionsRbd
Where to store the new image file. This location cannot point to a snapshot.

size: int
Size of the virtual disk in bytes

cluster−size: int (optional)
RBD object size

encrypt: RbdEncryptionCreateOptions (optional)
Image encryption options. (Since 6.1)

Since
2.12

BlockdevVmdkSubformat (Enum)
Subformat options for VMDK images

Values

monolithicSparse
Single file image with sparse cluster allocation

monolithicFlat
Single flat data image and a descriptor file

twoGbMaxExtentSparse
Data is split into 2GB (per virtual LBA) sparse extent files, in addition to a descriptor file

twoGbMaxExtentFlat
Data is split into 2GB (per virtual LBA) flat extent files, in addition to a descriptor file

streamOptimized
Single file image sparse cluster allocation, optimized for streaming over network.

Since
4.0

BlockdevVmdkAdapterType (Enum)
Adapter type info for VMDK images

Values

ide Not documented

buslogic
Not documented

lsilogic Not documented

legacyESX
Not documented

Since
4.0

BlockdevCreateOptionsVmdk (Object)
Driver specific image creation options for VMDK.

Members

file: BlockdevRef
Where to store the new image file. This refers to the image file for monolithcSparse and streamOptimized format, or the descriptor file for other formats.

size: int
Size of the virtual disk in bytes

extents: array of BlockdevRef (optional)
Where to store the data extents. Required for monolithcFlat, twoGbMaxExtentSparse and twoGbMaxExtentFlat formats. For monolithicFlat, only one entry is required; for twoGbMaxExtent* formats, the number of entries required is calculated as extent_number = virtual_size / 2GB. Providing more extents than will be used is an error.

subformat: BlockdevVmdkSubformat (optional)
The subformat of the VMDK image. Default: "monolithicSparse".

backing−file: string (optional)
The path of backing file. Default: no backing file is used.

adapter−type: BlockdevVmdkAdapterType (optional)
The adapter type used to fill in the descriptor. Default: ide.

hwversion: string (optional)
Hardware version. The meaningful options are "4" or "6". Default: "4".

toolsversion: string (optional)
VMware guest tools version. Default: "2147483647" (Since 6.2)

zeroed−grain: boolean (optional)
Whether to enable zeroed−grain feature for sparse subformats. Default: false.

Since
4.0

BlockdevCreateOptionsSsh (Object)
Driver specific image creation options for SSH.

Members

location: BlockdevOptionsSsh
Where to store the new image file

size: int
Size of the virtual disk in bytes

Since
2.12

BlockdevCreateOptionsVdi (Object)
Driver specific image creation options for VDI.

Members

file: BlockdevRef
Node to create the image format on

size: int
Size of the virtual disk in bytes

preallocation: PreallocMode (optional)
Preallocation mode for the new image (default: off; allowed values: off, metadata)

Since
2.12

BlockdevVhdxSubformat (Enum)

Values

dynamic
Growing image file

fixed Preallocated fixed−size image file

Since
2.12

BlockdevCreateOptionsVhdx (Object)
Driver specific image creation options for vhdx.

Members

file: BlockdevRef
Node to create the image format on

size: int
Size of the virtual disk in bytes

log−size: int (optional)
Log size in bytes, must be a multiple of 1 MB (default: 1 MB)

block−size: int (optional)
Block size in bytes, must be a multiple of 1 MB and not larger than 256 MB (default: automatically choose a block size depending on the image size)

subformat: BlockdevVhdxSubformat (optional)
vhdx subformat (default: dynamic)

block−state−zero: boolean (optional)
Force use of payload blocks of type 'ZERO'. Non−standard, but default. Do not set to 'off' when using 'qemu−img convert' with subformat=dynamic.

Since
2.12

BlockdevVpcSubformat (Enum)

Values

dynamic
Growing image file

fixed Preallocated fixed−size image file

Since
2.12

BlockdevCreateOptionsVpc (Object)
Driver specific image creation options for vpc (VHD).

Members

file: BlockdevRef
Node to create the image format on

size: int
Size of the virtual disk in bytes

subformat: BlockdevVpcSubformat (optional)
vhdx subformat (default: dynamic)

force−size: boolean (optional)
Force use of the exact byte size instead of rounding to the next size that can be represented in CHS geometry (default: false)

Since
2.12

BlockdevCreateOptions (Object)
Options for creating an image format on a given node.

Members

driver: BlockdevDriver
block driver to create the image format

The members of BlockdevCreateOptionsFile when driver is "file"

The members of BlockdevCreateOptionsGluster when driver is "gluster"

The members of BlockdevCreateOptionsLUKS when driver is "luks"

The members of BlockdevCreateOptionsNfs when driver is "nfs"

The members of BlockdevCreateOptionsParallels when driver is "parallels"

The members of BlockdevCreateOptionsQcow when driver is "qcow"

The members of BlockdevCreateOptionsQcow2 when driver is "qcow2"

The members of BlockdevCreateOptionsQed when driver is "qed"

The members of BlockdevCreateOptionsRbd when driver is "rbd"

The members of BlockdevCreateOptionsSsh when driver is "ssh"

The members of BlockdevCreateOptionsVdi when driver is "vdi"

The members of BlockdevCreateOptionsVhdx when driver is "vhdx"

The members of BlockdevCreateOptionsVmdk when driver is "vmdk"

The members of BlockdevCreateOptionsVpc when driver is "vpc"

Since
2.12

blockdev−create (Command)
Starts a job to create an image format on a given node. The job is automatically finalized, but a manual job−dismiss is required.

Arguments

job−id: string
Identifier for the newly created job.

options: BlockdevCreateOptions
Options for the image creation.

Since
3.0

BlockdevAmendOptionsLUKS (Object)
Driver specific image amend options for LUKS.

Members

The members of QCryptoBlockAmendOptionsLUKS

Since
5.1

BlockdevAmendOptionsQcow2 (Object)
Driver specific image amend options for qcow2. For now, only encryption options can be amended

encrypt Encryption options to be amended

Members

encrypt: QCryptoBlockAmendOptions (optional)
Not documented

Since
5.1

BlockdevAmendOptions (Object)
Options for amending an image format

Members

driver: BlockdevDriver
Block driver of the node to amend.

The members of BlockdevAmendOptionsLUKS when driver is "luks"

The members of BlockdevAmendOptionsQcow2 when driver is "qcow2"

Since
5.1

x−blockdev−amend (Command)
Starts a job to amend format specific options of an existing open block device The job is automatically finalized, but a manual job−dismiss is required.

Arguments

job−id: string
Identifier for the newly created job.

node−name: string
Name of the block node to work on

options: BlockdevAmendOptions
Options (driver specific)

force: boolean (optional)
Allow unsafe operations, format specific For luks that allows erase of the last active keyslot (permanent loss of data), and replacement of an active keyslot (possible loss of data if IO error happens)

Features

unstable
This command is experimental.

Since
5.1

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 (optional)
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. The node name is not present if the drive is empty. (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

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: JobType
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: JobType
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: JobType
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 } }
```

BLOCK_JOB_PENDING (Event)
Emitted when a block job is awaiting explicit authorization to finalize graph changes via **block−job−finalize**. If this job is part of a transaction, it will not emit this event until the transaction has converged first.

Arguments
type: JobType
job type

id: string
The job identifier.

Since
2.12

Example
```
<- { "event": "BLOCK_JOB_WAITING",
     "data": { "device": "drive0", "type": "mirror" },
     "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 data.

full preallocate all data by writing it to the device to ensure disk space is really available. This data may or may not be zero, depending on the image format and storage. **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.

Features

unstable
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.

FIXME Removing children from a quorum node means introducing gaps in the child indices. This cannot be represented in the 'children' list of BlockdevOptionsQuorum, as returned by

.bdrv_refresh_filename().

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": {} }
```

x−blockdev−set−iothread (Command)

Move **node** and its children into the **iothread**. If**iothr ead** is null then move **node** and its children into the main loop.

The node must not be attached to a BlockBackend.

Arguments

node−name: string

the name of the block driver node

iothread: StrOrNull

the name of the IOThread object or null for the main loop

force: boolean (optional)

true if the node and its children should be moved when a BlockBackend is already attached

Features

unstable

This command is experimental and intended for test cases that need control over IOThreads only.

Since

2.12

Example

```
1. Move a node into an IOThread
-> { "execute": "x-blockdev-set-iothread",
     "arguments": { "node-name": "disk1",
                    "iothread": "iothread0" } }
<- { "return": {} }

2. Move a node into the main loop
-> { "execute": "x-blockdev-set-iothread",
     "arguments": { "node-name": "disk1",
                                   "iothread": null } }
      <- { "return": {} }
```

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 } }
```

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, and rbd.

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,
                   "icount": 220414
     }
   }
```

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

PRManagerInfo (Object)
Information about a persistent reservation manager

Members
id: string
the identifier of the persistent reservation manager

connected: boolean
true if the persistent reservation manager is connected to the underlying storage or helper

Since
3.0

query−pr−managers (Command)
Returns a list of information about each persistent reservation manager.

Returns
a list of **PRManagerInfo** for each persistent reservation manager

Since
3.0

eject (Command)
Ejects the medium from a removable drive.

Arguments
device: string (optional)
Block device name

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.

Features
deprecated
Member **device** is deprecated. Use **id** instead.

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

Example

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

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

Features

deprecated

Member **device** is deprecated. Use **id** instead.

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

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

Features

deprecated
Member **device** is deprecated. Use **id** instead.

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": {} }
```

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

id: string
The name or QOM path of the guest device

Since
2.12

Example

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

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

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

id: string
The name or QOM path of the guest device

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

Since
2.12

Example

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

-> { "execute": "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, blockdev−remove−medium, blockdev−insert−medium and blockdev−close−tray).

Arguments

device: string (optional)
Block device name

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'

Features

deprecated
Member **device** is deprecated. Use **id** instead.

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": {} }
```

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 } }
```

PR_MANAGER_STATUS_CHANGED (Event)
Emitted whenever the connected status of a persistent reservation manager changes.

Arguments
id: string
The id of the PR manager object

connected: boolean
true if the PR manager is connected to a backend

Since
3.0

Example
```
<- { "event": "PR_MANAGER_STATUS_CHANGED",
     "data": { "id": "pr-helper0",
               "connected": true
     },
     "timestamp": { "seconds": 1519840375, "microseconds": 450486 } }
```

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": "virtio-blk-pci0/virtio-backend",
                    "bps": 0,
                    "bps_rd": 0,
                    "bps_wr": 0,
                    "iops": 512,
                    "iops_rd": 0,
                    "iops_wr": 0,
                    "bps_max": 0,
                    "bps_rd_max": 0,
                    "bps_wr_max": 0,
                    "iops_max": 0,
                    "iops_rd_max": 0,
                    "iops_wr_max": 0,
                    "bps_max_length": 0,
                    "iops_size": 0 } }
<- { "return": {} }

-> { "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": {} }
```

block−latency−histogram−set (Command)

Manage read, write and flush latency histograms for the device.

If only **id** parameter is specified, remove all present latency histograms for the device. Otherwise, add/reset some of (or all) latency histograms.

Arguments

id: string

The name or QOM path of the guest device.

boundaries: array of int (optional)

list of interval boundary values (see description in BlockLatencyHistogramInfo definition). If specified, all latency histograms are removed, and empty ones created for all io types with intervals corresponding to **boundaries** (except for io types, for which specific boundaries are set through the following parameters).

boundaries−read: array of int (optional)

list of interval boundary values for read latency histogram. If specified, old read latency histogram is removed, and empty one created with intervals corresponding to **boundaries−read**. The parameter has higher priority then **boundaries**.

boundaries−write: array of int (optional)
list of interval boundary values for write latency histogram.

boundaries−flush: array of int (optional)
list of interval boundary values for flush latency histogram.

Returns
error if device is not found or any boundary arrays are invalid.

Since
4.0

Example
```
set new histograms for all io types with intervals
[0, 10), [10, 50), [50, 100), [100, +inf):

-> { "execute": "block-latency-histogram-set",
     "arguments": { "id": "drive0",
                    "boundaries": [10, 50, 100] } }
<- { "return": {} }
```

Example
```
set new histogram only for write, other histograms will remain
not changed (or not created):

-> { "execute": "block-latency-histogram-set",
     "arguments": { "id": "drive0",
                    "boundaries-write": [10, 50, 100] } }
<- { "return": {} }
```

Example
```
set new histograms with the following intervals:
  read, flush: [0, 10), [10, 50), [50, 100), [100, +inf)
  write: [0, 1000), [1000, 5000), [5000, +inf)

-> { "execute": "block-latency-histogram-set",
     "arguments": { "id": "drive0",
                    "boundaries": [10, 50, 100],
                    "boundaries-write": [1000, 5000] } }
<- { "return": {} }
```

Example
```
remove all latency histograms:

-> { "execute": "block-latency-histogram-set",
     "arguments": { "id": "drive0" } }
<- { "return": {} }
```

Block device exports

NbdServerOptions (Object)
Keep this type consistent with the nbd−server−start arguments. The only intended difference is using SocketAddress instead of SocketAddressLegacy.

Members

addr: SocketAddress
Address on which to listen.

tls−creds: string (optional)
ID of the TLS credentials object (since 2.6).

tls−authz: string (optional)
ID of the QAuthZ authorization object used to validate the client's x509 distinguished name. This object is is only resolved at time of use, so can be deleted and recreated on the fly while the NBD server is active. If missing, it will default to denying access (since 4.0).

max−connections: int (optional)
The maximum number of connections to allow at the same time, 0 for unlimited. (since 5.2; default: 0)

Since
4.2

nbd−server−start (Command)
Start an NBD server listening on the given host and port. Block devices can then be exported using **nbd−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".

Keep this type consistent with the NbdServerOptions type. The only intended difference is using SocketAddressLegacy instead of SocketAddress.

Arguments

addr: SocketAddressLegacy
Address on which to listen.

tls−creds: string (optional)
ID of the TLS credentials object (since 2.6).

tls−authz: string (optional)
ID of the QAuthZ authorization object used to validate the client's x509 distinguished name. This object is is only resolved at time of use, so can be deleted and recreated on the fly while the NBD server is active. If missing, it will default to denying access (since 4.0).

max−connections: int (optional)
The maximum number of connections to allow at the same time, 0 for unlimited. (since 5.2; default: 0)

Returns
error if the server is already running.

Since
1.3

BlockExportOptionsNbdBase (Object)
An NBD block export (common options shared between nbd−server−add and the NBD branch of block−export−add).

Members

name: string (optional)
Export name. If unspecified, the **device** parameter is used as the export name. (Since 2.12)

description: string (optional)
Free−form description of the export, up to 4096 bytes. (Since 5.0)

Since
5.0

BlockExportOptionsNbd (Object)
An NBD block export (distinct options used in the NBD branch of block−export−add).

Members

bitmaps: array of string (optional)
Also export each of the named dirty bitmaps reachable from **device**, so the NBD client can use NBD_OPT_SET_META_CONTEXT with the metadata context name

"qemu:dirty−bitmap:BITMAP" to inspect each bitmap.

allocation−depth: boolean (optional)
Also export the allocation depth map for **device**, so the NBD client can use NBD_OPT_SET_META_CONTEXT with the metadata context name "qemu:allocation−depth" to inspect allocation details. (since 5.2)

The members of BlockExportOptionsNbdBase

Since
5.2

BlockExportOptionsVhostUserBlk (Object)
A vhost−user−blk block export.

Members

addr: SocketAddress
The vhost−user socket on which to listen. Both 'unix' and 'fd' SocketAddress types are supported. Passed fds must be UNIX domain sockets.

logical−block−size: int (optional)
Logical block size in bytes. Defaults to 512 bytes.

num−queues: int (optional)
Number of request virtqueues. Must be greater than 0. Defaults to 1.

Since
5.2

FuseExportAllowOther (Enum)
Possible allow_other modes for FUSE exports.

Values

off	Do not pass allow_other as a mount option.
on	Pass allow_other as a mount option.
auto	Try mounting with allow_other first, and if that fails, retry without allow_other.

Since
6.1

BlockExportOptionsFuse (Object)
Options for exporting a block graph node on some (file) mountpoint as a raw image.

Members

mountpoint: string
Path on which to export the block device via FUSE. This must point to an existing regular file.

growable: boolean (optional)
Whether writes beyond the EOF should grow the block node accordingly. (default: false)

allow−other: FuseExportAllowOther (optional)
If this is off, only qemu's user is allowed access to this export. That cannot be changed even with chmod or chown. Enabling this option will allow other users access to the export with the FUSE mount option "allow_other". Note that using allow_other as a non−root user requires user_allow_other to be enabled in the global fuse.conf configuration file. In auto mode (the default), the FUSE export driver will first attempt to mount the export with allow_other, and if that fails, try again without. (since 6.1; default: auto)

Since
6.0

If

CONFIG_FUSE

NbdServerAddOptions (Object)
An NBD block export, per legacy nbd−server−add command.

Members

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).

bitmap: string (optional)
Also export a single dirty bitmap reachable from **device**, so the NBD client can use NBD_OPT_SET_META_CONTEXT with the metadata context name "qemu:dirty−bitmap:BITMAP" to inspect the bitmap (since 4.0).

The members of BlockExportOptionsNbdBase

Since
5.0

nbd−server−add (Command)
Export a block node to QEMU's embedded NBD server.

The export name will be used as the id for the resulting block export.

Arguments
The members of NbdServerAddOptions

Features

deprecated
This command is deprecated. Use **block−export−add** instead.

Returns
error if the server is not running, or export with the same name already exists.

Since
1.3

BlockExportRemoveMode (Enum)
Mode for removing a block export.

Values

safe Remove export if there are no existing connections, fail otherwise.

hard Drop all connections immediately and remove export.

Potential additional modes to be added in the future:

hide: Just hide export from new clients, leave existing connections as is. Remove export after all clients are disconnected.

soft: Hide export from new clients, answer with ESHUTDOWN for all further requests from existing clients.

Since
2.12

nbd−server−remove (Command)
Remove NBD export by name.

Arguments

name: string
Block export id.

mode: BlockExportRemoveMode (optional)
Mode of command operation. See **BlockExportRemoveMode** description. Default is 'safe'.

Features

deprecated
This command is deprecated. Use **block−export−del** instead.

Returns

error if

- the server is not running
- export is not found
- mode is 'safe' and there are existing connections

Since
2.12

nbd−server−stop (Command)
Stop QEMU's embedded NBD server, and unregister all devices previously added via **nbd−server−add**.

Since
1.3

BlockExportType (Enum)
An enumeration of block export types

Values

nbd NBD export

vhost−user−blk
vhost−user−blk export (since 5.2)

fuse (**If: CONFIG_FUSE**)
FUSE export (since: 6.0)

Since
4.2

BlockExportOptions (Object)
Describes a block export, i.e. how single node should be exported on an external interface.

Members

id: string
A unique identifier for the block export (across all export types)

node−name: string
The node name of the block node to be exported (since: 5.2)

writable: boolean (optional)
True if clients should be able to write to the export (default false)

writethrough: boolean (optional)
If true, caches are flushed after every write request to the export before completion is signalled. (since: 5.2; default: false)

iothread: string (optional)
The name of the iothread object where the export will run. The default is to use the thread currently associated with the block node. (since: 5.2)

fixed−iothread: boolean (optional)
True prevents the block node from being moved to another thread while the export is active. If true and **iothread** is given, export creation fails if the block node cannot be moved to the iothread. The default is false. (since: 5.2)

type: BlockExportType
Not documented

The members of BlockExportOptionsNbd when type is "nbd"

The members of BlockExportOptionsVhostUserBlk when type is "vhost−user−blk"

The members of **BlockExportOptionsFuse** when **type** is **"fuse"** (**If: CONFIG_FUSE**)

Since
4.2

block−export−add (Command)
Creates a new block export.

Arguments
The members of BlockExportOptions

Since
5.2

block−export−del (Command)
Request to remove a block export. This drops the user's reference to the export, but the export may still stay around after this command returns until the shutdown of the export has completed.

Arguments
id: string
Block export id.

mode: BlockExportRemoveMode (optional)
Mode of command operation. See **BlockExportRemoveMode** description. Default is 'safe'.

Returns
Error if the export is not found or **mode** is 'safe' and the export is still in use (e.g. by existing client connections)

Since
5.2

BLOCK_EXPORT_DELETED (Event)
Emitted when a block export is removed and its id can be reused.

Arguments
id: string
Block export id.

Since
5.2

BlockExportInfo (Object)
Information about a single block export.

Members
id: string
The unique identifier for the block export

type: BlockExportType
The block export type

node−name: string
The node name of the block node that is exported

shutting−down: boolean
True if the export is shutting down (e.g. after a block−export−del command, but before the shutdown has completed)

Since
5.2

query−block−exports (Command)
Returns
A list of BlockExportInfo describing all block exports

Since
5.2

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

query−chardev (Command)
Returns information about current character devices.

Returns
a list of **ChardevInfo**

Since
0.14

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,serv
            "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)

tls−authz: string (optional)
the ID of the QAuthZ authorization object against which the client's x509 distinguished name will be validated. This object is only resolved at time of use, so can be deleted and recreated on the fly while the chardev server is active. If missing, it will default to denying access (since 4.0)

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

wait: boolean (optional)
wait for incoming connection on server sockets (default: false). Silently ignored with server: false. This use is deprecated.

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)

websocket: boolean (optional)
enable websocket protocol on server sockets (default: false) (Since: 3.1)

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 ˆC) be delivered to qemu. Default: true.

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

If
CONFIG_SPICE

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

If
CONFIG_SPICE

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

ChardevQemuVDAgent (Object)
Configuration info for qemu vdagent implementation.

Members

mouse: boolean (optional)
enable/disable mouse, default is enabled.

clipboard: boolean (optional)
enable/disable clipboard, default is disabled.

The members of ChardevCommon

Since
6.1

If
CONFIG_SPICE_PROTOCOL

ChardevBackendKind (Enum)

Values

pipe Since 1.5

udp	Since 1.5
mux	Since 1.5
msmouse	Since 1.5
wctablet	Since 2.9
braille	Since 1.5
testdev	Since 2.2
stdio	Since 1.5
console	Since 1.5
spicevmc (**If: CONFIG_SPICE**)	Since 1.5
spiceport (**If: CONFIG_SPICE**)	Since 1.5
qemu−vdagent (**If: CONFIG_SPICE_PROTOCOL**)	Since 6.1
vc	v1.5
ringbuf	Since 1.6
memory	Since 1.5
file	Not documented
serial	Not documented
parallel	Not documented
socket	Not documented
pty	Not documented
null	Not documented

Since
1.4

ChardevFileWrapper (Object)
Members
data: ChardevFile
Not documented

Since
1.4

ChardevHostdevWrapper (Object)
Members
data: ChardevHostdev
Not documented

Since
1.4

ChardevSocketWrapper (Object)
Members
data: ChardevSocket
Not documented

Since
1.4

ChardevUdpWrapper (Object)
Members
data: ChardevUdp
Not documented

Since
1.5

ChardevCommonWrapper (Object)
Members
data: ChardevCommon
Not documented

Since
2.6

ChardevMuxWrapper (Object)
Members
data: ChardevMux
Not documented

Since
1.5

ChardevStdioWrapper (Object)
Members
data: ChardevStdio
Not documented

Since
1.5

ChardevSpiceChannelWrapper (Object)
Members
data: ChardevSpiceChannel
Not documented

Since
1.5

If
CONFIG_SPICE

ChardevSpicePortWrapper (Object)
Members
data: ChardevSpicePort
Not documented

Since
1.5

If
CONFIG_SPICE

ChardevQemuVDAgentWrapper (Object)
Members
data: ChardevQemuVDAgent
Not documented

Since
6.1

If
CONFIG_SPICE_PROTOCOL

ChardevVCWrapper (Object)
Members
data: ChardevVC
Not documented

Since
1.5

ChardevRingbufWrapper (Object)
Members
data: ChardevRingbuf
Not documented

Since
1.5

ChardevBackend (Object)
Configuration info for the new chardev backend.

Members
type: ChardevBackendKind
Not documented

The members of ChardevFileWrapper when type is "file"

The members of ChardevHostdevWrapper when type is "serial"

The members of ChardevHostdevWrapper when type is "parallel"

The members of ChardevHostdevWrapper when type is "pipe"

The members of ChardevSocketWrapper when type is "socket"

The members of ChardevUdpWrapper when type is "udp"

The members of ChardevCommonWrapper when type is "pty"

The members of ChardevCommonWrapper when type is "null"

The members of ChardevMuxWrapper when type is "mux"

The members of ChardevCommonWrapper when type is "msmouse"

The members of ChardevCommonWrapper when type is "wctablet"

The members of ChardevCommonWrapper when type is "braille"

The members of ChardevCommonWrapper when type is "testdev"

The members of ChardevStdioWrapper when type is "stdio"

The members of ChardevCommonWrapper when type is "console"

The members of **ChardevSpiceChannelWrapper** when **type** is **"spicevmc"** (**If: CONFIG_SPICE**)

The members of **ChardevSpicePortWrapper** when **type** is **"spiceport"** (**If: CONFIG_SPICE**)

The members of **ChardevQemuVDAgentWrapper** when **type** is **"qemu−vdagent"** (**If: CONFIG_SPICE_PROTOCOL**)

The members of ChardevVCWrapper when type is "vc"

The members of ChardevRingbufWrapper when type is "ringbuf"

The members of ChardevRingbufWrapper when type is "memory"

Since
1.4

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

Note

This event is rate−limited.

Since

2.1

Example

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

DUMP GUEST MEMORY

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

win−dmp
Windows full crashdump format, can be used instead of ELF converting (since 2.13)

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
final dump status

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"] }
```

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

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.

Additional arguments depend on the type.

Arguments
The members of Netdev

Since
0.14

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

Example

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

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

domainname: string (optional)
guest−visible domain name of the virtual nameserver (since 3.0)

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

tftp−server−name: string (optional)
RFC2132 "TFTP server name" string (Since 3.1)

Since
1.2

NetdevTapOptions (Object)
Used to configure a host TAP network interface backend.

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)
Socket netdevs are used to establish a network connection to another QEMU virtual machine via a TCP socket.

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)
Configure an Ethernet over L2TPv3 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 coookies

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 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

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

netdev: string (optional)
used to connect hub to a netdev instead of a device (since 2.12)

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

NetdevVhostVDPAOptions (Object)
Vhost−vdpa network backend

vDPA device is a device that uses a datapath which complies with the virtio specifications with a vendor specific control path.

Members

vhostdev: string (optional)
path of vhost−vdpa device (default:'/dev/vhost−vdpa−0')

queues: int (optional)
number of queues to be created for multiqueue vhost−vdpa (default: 1)

Since
5.1

NetClientDriver (Enum)
Available netdev drivers.

Values

none	Not documented

nic	Not documented
user	Not documented
tap	Not documented
l2tpv3	Not documented
socket	Not documented
vde	Not documented
bridge	Not documented
hubport	Not documented
netmap	Not documented
vhost−user	Not documented
vhost−vdpa	Not documented

Since
2.7

vhost−vdpa since 5.1

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 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 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"

The members of NetdevVhostVDPAOptions when type is "vhost−vdpa"

Since
1.2

'l2tpv3' − since 2.1

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 } }
   }
```

AnnounceParameters (Object)
Parameters for self−announce timers

Members
initial: int
Initial delay (in ms) before sending the first GARP/RARP announcement

max: int
Maximum delay (in ms) between GARP/RARP announcement packets

rounds: int
Number of self−announcement attempts

step: int
Delay increase (in ms) after each self−announcement attempt

interfaces: array of string (optional)
An optional list of interface names, which restricts the announcement to the listed interfaces. (Since 4.1)

id: string (optional)
A name to be used to identify an instance of announce−timers and to allow it to modified later. Not for use as part of the migration parameters. (Since 4.1)

Since
4.0

announce−self (Command)
Trigger generation of broadcast RARP frames to update network switches. This can be useful when network bonds fail−over the active slave.

Arguments
The members of AnnounceParameters

Example

```
-> { "execute": "announce-self",
     "arguments": {
         "initial": 50, "max": 550, "rounds": 10, "step": 50,
         "interfaces": ["vn2", "vn3"], "id": "bob" } }
<- { "return": {} }
```

Since
4.0

FAILOVER_NEGOTIATED (Event)
Emitted when VIRTIO_NET_F_STANDBY was enabled during feature negotiation. Failover primary devices which were hidden (not hotplugged when requested) before will now be hotplugged by the virtio−net standby device.

device−id: QEMU device id of the unplugged device

Arguments
device−id: string
Not documented

Since
4.2

Example

```
<- { "event": "FAILOVER_NEGOTIATED",
     "data": "net1" }
```

RDMA DEVICE

RDMA_GID_STATUS_CHANGED (Event)
Emitted when guest driver adds/deletes GID to/from device

Arguments
netdev: string
RoCE Network Device name

gid−status: boolean
Add or delete indication

subnet−prefix: int
Subnet Prefix

interface−id: int
Not documented
interface−id : Interface ID

Since
4.0

Example

```
<- {"timestamp": {"seconds": 1541579657, "microseconds": 986760},
    "event": "RDMA_GID_STATUS_CHANGED",
    "data":
        {"netdev": "bridge0",
        "interface-id": 15880512517475447892,
        "gid-status": true,
        "subnet-prefix": 33022}}
```

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 eumeration of port duplex states.

Values

half	half duplex
full	full duplex

Since
2.4

RockerPortAutoneg (Enum)
An eumeration 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}
   ]}
```

TPM (TRUSTED PLATFORM MODULE) DEVICES

TpmModel (Enum)
An enumeration of TPM models

Values
tpm−tis
TPM TIS model

tpm−crb
TPM CRB model (since 2.12)

tpm−spapr
TPM SPAPR model (since 5.0)

Since
1.5

If
CONFIG_TPM

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", "tpm-crb", "tpm-spapr" ] }
```

If
CONFIG_TPM

TpmType (Enum)
An enumeration of TPM types

Values
passthrough
TPM passthrough type

emulator
Software Emulator TPM type Since: 2.11

Since
1.5

If
CONFIG_TPM

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", "emulator" ] }
```

If
CONFIG_TPM

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

If
CONFIG_TPM

TPMEmulatorOptions (Object)
Information about the TPM emulator type

Members
chardev: string
Name of a unix socket chardev

Since
2.11

If
  CONFIG_TPM

TPMPassthroughOptionsWrapper (Object)
Members
  data: TPMPassthroughOptions
    Not documented

Since
  1.5

If
  CONFIG_TPM

TPMEmulatorOptionsWrapper (Object)
Members
  data: TPMEmulatorOptions
    Not documented

Since
  2.11

If
  CONFIG_TPM

TpmTypeOptions (Object)
  A union referencing different TPM backend types' configuration options

Members
  type: TpmType

- 'passthrough' The configuration options for the TPM passthrough type
- 'emulator' The configuration options for TPM emulator backend type

  The members of TPMPassthroughOptionsWrapper when type is "passthrough"

  The members of TPMEmulatorOptionsWrapper when type is "emulator"

Since
  1.5

If
  CONFIG_TPM

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

If
  CONFIG_TPM

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"
       }
     ]
   }
```

If

CONFIG_TPM

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

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

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

device: string (optional)
ID of the display device that should be dumped. If this parameter is missing, the primary display will be used. (Since 2.12)

head: int (optional)
head to use in case the device supports multiple heads. If this parameter is missing, head #0 will be used. Also note that the head can only be specified in conjunction with the device ID. (Since 2.12)

Returns
Nothing on success

Since
0.14

Example

```
-> { "execute": "screendump",
     "arguments": { "filename": "/tmp/image" } }
<- { "return": {} }
```

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

If
CONFIG_SPICE

SpiceServerInfo (Object)
Information about a SPICE server

Members
auth: string (optional)
authentication method

The members of SpiceBasicInfo

Since
2.1

If
CONFIG_SPICE

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

If
CONFIG_SPICE

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

If
CONFIG_SPICE

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

If
CONFIG_SPICE

query−spice (Command)
Returns information about the current SPICE server

Returns
SpiceInfo

Since
0.14

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 ... ]
         ]
      }
   }
```

If
   CONFIG_SPICE

SPICE_CONNECTED (Event)
   Emitted when a SPICE client establishes a connection

Arguments
   server: SpiceBasicInfo
      server information

   client: SpiceBasicInfo
      client information

Since
   0.14

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"}
   }}
```

If
   CONFIG_SPICE

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

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}
   }}
```

If
CONFIG_SPICE

SPICE_DISCONNECTED (Event)
Emitted when the SPICE connection is closed

Arguments

server: SpiceBasicInfo
server information

client: SpiceBasicInfo
client information

Since
0.14

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"}
   }}
```

If
CONFIG_SPICE

SPICE_MIGRATE_COMPLETED (Event)
Emitted when SPICE migration has completed

Since
1.3

Example

```
<- { "timestamp": {"seconds": 1290688046, "microseconds": 417172},
     "event": "SPICE_MIGRATE_COMPLETED" }
```

If
CONFIG_SPICE

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

If
CONFIG_VNC

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

If
CONFIG_VNC

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

If
CONFIG_VNC

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
- 'vencrypt+plain' if VEncrypt is used with plain text authentication
- 'vencrypt+tls+none' if VEncrypt is used with TLS and no authentication
- 'vencrypt+tls+vnc' if VEncrypt is used with TLS and VNC authentication
- 'vencrypt+tls+plain' if VEncrypt is used with TLS and plain text auth
- 'vencrypt+x509+none' if VEncrypt is used with x509 and no auth
- 'vencrypt+x509+vnc' if VEncrypt is used with x509 and VNC auth
- 'vencrypt+x509+plain' if VEncrypt is used with x509 and plain text auth
- 'vencrypt+tls+sasl' if VEncrypt is used with TLS and SASL auth
- 'vencrypt+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

If
CONFIG_VNC

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
vencrypt	Not documented
sasl	Not documented

Since
2.3

If
CONFIG_VNC

VncVencryptSubAuth (Enum)
vnc sub authentication method with vencrypt.

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

If
CONFIG_VNC

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

If
CONFIG_VNC

VncInfo2 (Object)
Information about a vnc server

Members

id: string
vnc server name.

server: array of VncServerInfo2
A list of **VncBasincInfo** 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

If
CONFIG_VNC

query−vnc (Command)
Returns information about the current VNC server

Returns
VncInfo

Since
0.14

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"
            }
         ]
      }
   }
```

If
CONFIG_VNC

query−vnc−servers (Command)
Returns a list of vnc servers. The list can be empty.

Returns
a list of **VncInfo2**

Since
2.3

If
  CONFIG_VNC

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.

If
  CONFIG_VNC

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

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 } }
```

If
  CONFIG_VNC

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

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 } }
```

If

CONFIG_VNC

VNC_DISCONNECTED (Event)

Emitted when the connection is closed

Arguments

server: VncServerInfo

server information

client: VncClientInfo

client information

Since

0.13

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 } }
```

If

CONFIG_VNC

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

query−mice (Command)

Returns information about each active mouse device

Returns

a list of **MouseInfo** for each device

Since

0.14

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
pause	since 2.0
ro	since 2.4
kp_comma	since 2.4
kp_equals	since 2.6
power	since 2.6
hiragana	since 2.9
henkan	since 2.9
yen	since 2.9
sleep	since 2.10
wake	since 2.10
audionext	since 2.10
audioprev	since 2.10
audiostop	since 2.10
audioplay	since 2.10
audiomute	since 2.10

volumeup	since 2.10
volumedown	since 2.10
mediaselect	since 2.10
mail	since 2.10
calculator	since 2.10
computer	since 2.10
ac_home	since 2.10
ac_back	since 2.10
ac_forward	since 2.10
ac_refresh	since 2.10
ac_bookmarks	since 2.10
muhenkan	since 2.12
katakanahiragana	since 2.12
lang1	since 6.1
lang2	since 6.1
shift	Not documented
shift_r	Not documented
alt	Not documented
alt_r	Not documented
ctrl	Not documented
ctrl_r	Not documented
menu	Not documented
esc	Not documented
1	Not documented
2	Not documented
3	Not documented
4	Not documented
5	Not documented
6	Not documented
7	Not documented

8	Not documented
9	Not documented
0	Not documented
minus	Not documented
equal	Not documented
backspace	Not documented
tab	Not documented
q	Not documented
w	Not documented
e	Not documented
r	Not documented
t	Not documented
y	Not documented
u	Not documented
i	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
spc	Not documented
caps_lock	Not documented
f1	Not documented
f2	Not documented
f3	Not documented
f4	Not documented
f5	Not documented
f6	Not documented
f7	Not documented
f8	Not documented
f9	Not documented
f10	Not documented
num_lock	Not documented
scroll_lock	Not documented
kp_divide	Not documented
kp_multiply	Not documented
kp_subtract	Not documented
kp_add	Not documented
kp_enter	Not documented

kp_decimal	Not documented
sysrq	Not documented
kp_0	Not documented
kp_1	Not documented
kp_2	Not documented
kp_3	Not documented
kp_4	Not documented
kp_5	Not documented
kp_6	Not documented
kp_7	Not documented
kp_8	Not documented
kp_9	Not documented
less	Not documented
f11	Not documented
f12	Not documented
print	Not documented
home	Not documented
pgup	Not documented
pgdn	Not documented
end	Not documented
left	Not documented
up	Not documented
down	Not documented
right	Not documented
insert	Not documented
delete	Not documented
stop	Not documented
again	Not documented
props	Not documented
undo	Not documented
front	Not documented
copy	Not documented
open	Not documented
paste	Not documented
find	Not documented
cut	Not documented
lf	Not documented
help	Not documented

meta_l Not documented

meta_r Not documented

compose
Not documented

'sysrq' was mistakenly added to hack around the fact that the ps2 driver was not generating correct scancodes sequences when 'alt+print' was pressed. This flaw is now fixed and the 'sysrq' key serves no further purpose. Any further use of 'sysrq' will be transparently changed to 'print', so they are effectively synonyms.

Since
1.3

KeyValueKind (Enum)
Values
number
Not documented

qcode Not documented

Since
1.3

IntWrapper (Object)
Members
data: int
Not documented

Since
1.3

QKeyCodeWrapper (Object)
Members
data: QKeyCode
Not documented

Since
1.3

KeyValue (Object)
Represents a keyboard key.

Members
type: KeyValueKind
Not documented

The members of IntWrapper when type is "number"

The members of QKeyCodeWrapper when type is "qcode"

Since
1.3

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

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 −> 0x7ffff

Since
2.0

InputEventKind (Enum)

Values

key	Not documented
btn	Not documented
rel	Not documented
abs	Not documented

Since
2.0

InputKeyEventWrapper (Object)

Members

data: InputKeyEvent
Not documented

Since
2.0

InputBtnEventWrapper (Object)

Members

data: InputBtnEvent
Not documented

Since
2.0

InputMoveEventWrapper (Object)

Members

data: InputMoveEvent
Not documented

Since
2.0

InputEvent (Object)
Input event union.

Members

type: InputEventKind
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

The members of InputKeyEventWrapper when type is "key"

The members of InputBtnEventWrapper when type is "btn"

The members of InputMoveEventWrapper when type is "rel"

The members of InputMoveEventWrapper when type is "abs"

Since
2.0

input−send−event (Command)
Send input event(s) to guest.

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**de vice** is missing, only devices that have no input routing config are admissible. If**de vice** is specified, both input devices with and without input routing config are admissible, but devices with input routing config take precedence.

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.

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": {} }
```

DisplayGTK (Object)

GTK display options.

Members

grab−on−hover: boolean (optional)
Grab keyboard input on mouse hover.

zoom−to−fit: boolean (optional)
Zoom guest display to fit into the host window. When turned off the host window will be resized instead. In case the display device can notify the guest on window resizes (virtio−gpu) this will default to "on", assuming the guest will resize the display to match the window size then. Otherwise it defaults to "off". Since 3.1

Since

2.12

DisplayEGLHeadless (Object)

EGL headless display options.

Members

rendernode: string (optional)
Which DRM render node should be used. Default is the first available node on the host.

Since

3.1

DisplayGLMode (Enum)

Display OpenGL mode.

Values

off	Disable OpenGL (default).
on	Use OpenGL, pick context type automatically. Would better be named 'auto' but is called 'on' for backward compatibility with bool type.
core	Use OpenGL with Core (desktop) Context.
es	Use OpenGL with ES (embedded systems) Context.

Since

3.0

DisplayCurses (Object)
Curses display options.

Members

charset: string (optional)
Font charset used by guest (default: CP437).

Since
4.0

DisplayType (Enum)
Display (user interface) type.

Values

default The default user interface, selecting from the first available of gtk, sdl, cocoa, and vnc.

none No user interface or video output display. The guest will still see an emulated graphics card, but its output will not be displayed to the QEMU user.

gtk (**If: CONFIG_GTK**)
The GTK user interface.

sdl (**If: CONFIG_SDL**)
The SDL user interface.

egl−headless (**If: CONFIG_OPENGL and CONFIG_GBM**)
No user interface, offload GL operations to a local DRI device. Graphical display need to be paired with VNC or Spice. (Since 3.1)

curses (**If: CONFIG_CURSES**)
Display video output via curses. For graphics device models which support a text mode, QEMU can display this output using a curses/ncurses interface. Nothing is displayed when the graphics device is in graphical mode or if the graphics device does not support a text mode. Generally only the VGA device models support text mode.

cocoa (**If: CONFIG_COCOA**)
The Cocoa user interface.

spice−app (**If: CONFIG_SPICE**)
Set up a Spice server and run the default associated application to connect to it. The server will redirect the serial console and QEMU monitors. (Since 4.0)

Since
2.12

DisplayOptions (Object)
Display (user interface) options.

Members

type: DisplayType
Which DisplayType qemu should use.

full−screen: boolean (optional)
Start user interface in fullscreen mode (default: off).

window−close: boolean (optional)
Allow to quit qemu with window close button (default: on).

show−cursor: boolean (optional)
Force showing the mouse cursor (default: off). (since: 5.0)

gl: DisplayGLMode (optional)
Enable OpenGL support (default: off).

The members of **DisplayGTK** when **type** is **"gtk"** (**If: CONFIG_GTK**)

The members of **DisplayCurses** when **type** is **"curses"** (**If: CONFIG_CURSES**)

The members of **DisplayEGLHeadless** when **type** is **"egl−headless"** (**If: CONFIG_OPENGL and CONFIG_GBM**)

Since
2.12

query−display−options (Command)
Returns information about display configuration

Returns
DisplayOptions

Since
3.1

DisplayReloadType (Enum)
Available DisplayReload types.

Values
vnc VNC display

Since
6.0

DisplayReloadOptionsVNC (Object)
Specify the VNC reload options.

Members
tls−certs: boolean (optional)
reload tls certs or not.

Since
6.0

DisplayReloadOptions (Object)
Options of the display configuration reload.

Members
type: DisplayReloadType
Specify the display type.

The members of DisplayReloadOptionsVNC when type is "vnc"

Since
6.0

display−reload (Command)
Reload display configuration.

Arguments
The members of DisplayReloadOptions

Returns
Nothing on success.

Since
6.0

Example

```
-> { "execute": "display-reload",
     "arguments": { "type": "vnc", "tls-certs": true  } }
<- { "return": {} }
```

USER AUTHORIZATION

QAuthZListPolicy (Enum)

The authorization policy result

Values

deny	deny access
allow	allow access

Since

4.0

QAuthZListFormat (Enum)

The authorization policy match format

Values

exact	an exact string match
glob	string with ? and * shell wildcard support

Since

4.0

QAuthZListRule (Object)

A single authorization rule.

Members

match: string
a string or glob to match against a user identity

policy: QAuthZListPolicy
the result to return if **match** evaluates to true

format: QAuthZListFormat (optional)
the format of the **match** rule (default 'exact')

Since

4.0

AuthZListProperties (Object)

Properties for authz−list objects.

Members

policy: QAuthZListPolicy (optional)
Default policy to apply when no rule matches (default: deny)

rules: array of QAuthZListRule (optional)
Authorization rules based on matching user

Since

4.0

AuthZListFileProperties (Object)

Properties for authz−listfile objects.

Members

filename: string
File name to load the configuration from. The file must contain valid JSON for AuthZListProperties.

refresh: boolean (optional)
If true, inotify is used to monitor the file, automatically reloading changes. If an error occurs during reloading, all authorizations will fail until the file is next successfully loaded. (default: true if the binary was built with CONFIG_INOTIFY1, false otherwise)

Since
4.0

AuthZPAMProperties (Object)
Properties for authz−pam objects.

Members

service: string
PAM service name to use for authorization

Since
4.0

AuthZSimpleProperties (Object)
Properties for authz−simple objects.

Members

identity: string
Identifies the allowed user. Its format depends on the network service that authorization object is associated with. For authorizing based on TLS x509 certificates, the identity must be the x509 distinguished name.

Since
4.0

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)

multifd−bytes: int
The number of bytes sent through multifd (since 3.0)

pages−per−second: int
the number of memory pages transferred per second (Since 4.0)

Since
0.14

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)

encoding−rate: number
rate of encoded bytes (since 5.1)

overflow: int
number of overflows

Since
1.2

CompressionStats (Object)
Detailed migration compression statistics

Members

pages: int
amount of pages compressed and transferred to the target VM

busy: int
count of times that no free thread was available to compress data

busy−rate: number
rate of thread busy

compressed−size: int
amount of bytes after compression

compression−rate: number
rate of compressed size

Since
3.1

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)

postcopy−paused
during postcopy but paused. (since 3.0)

postcopy−recover
trying to recover from a paused postcopy. (since 3.0)

completed
migration is finished.

failed some error occurred during migration process.

colo 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)

pre−switchover
Paused before device serialisation. (since 2.11)

device During device serialisation when pause−before−switchover is enabled (since 2.11)

wait−unplug
wait for device unplug request by guest OS to be completed. (since 4.2)

Since
2.3

VfioStats (Object)
Detailed VFIO devices migration statistics

Members
transferred: int
amount of bytes transferred to the target VM by VFIO devices

Since
5.2

MigrationInfo (Object)
Information about current migration process.

Members
status: MigrationStatus (optional)
MigrationStatus describing the current migration status. If this field is not returned, no migration process has been initiated

ram: MigrationStats (optional)
MigrationStats containing detailed migration status, only returned if status is 'active' or 'completed'(since 1.2)

disk: MigrationStats (optional)
MigrationStats containing detailed disk migration status, only returned if status is 'active' and it is a block migration

xbzrle−cache: XBZRLECacheStats (optional)
XBZRLECacheStats containing detailed XBZRLE migration statistics, only returned if XBZRLE feature is on and status is 'active' or 'completed' (since 1.2)

total−time: int (optional)
total amount of milliseconds since migration started. If migration has ended, it returns the total migration time. (since 1.2)

downtime: int (optional)
only present when migration finishes correctly total downtime in milliseconds for the guest. (since 1.3)

expected−downtime: int (optional)
only present while migration is active expected downtime in milliseconds for the guest in last walk of the dirty bitmap. (since 1.3)

setup−time: int (optional)
amount of setup time in milliseconds *before* the iterations begin but *after* 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)

cpu−throttle−percentage: int (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)

postcopy−blocktime: int (optional)
total time when all vCPU were blocked during postcopy live migration. This is only present when the postcopy−blocktime migration capability is enabled. (Since 3.0)

postcopy−vcpu−blocktime: array of int (optional)
list of the postcopy blocktime per vCPU. This is only present when the postcopy−blocktime migration capability is enabled. (Since 3.0)

compression: CompressionStats (optional)
migration compression statistics, only returned if compression feature is on and status is 'active' or 'completed' (Since 3.1)

socket−address: array of SocketAddress (optional)
Only used for tcp, to know what the real port is (Since 4.0)

vfio: VfioStats (optional)
VfioStats containing detailed VFIO devices migration statistics, only returned if VFIO device is present, migration is supported by all VFIO devices and status is 'active' or 'completed' (since 5.2)

blocked−reasons: array of string (optional)
A list of reasons an outgoing migration is blocked. Present and non−empty when migration is blocked. (since 6.0)

Since
0.14

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

Example

```
1. Before the first migration

-> { "execute": "query-migrate" }
<- { "return": {} }

2. Migration is done and has succeeded

-> { "execute": "query-migrate" }
<- { "return": {
        "status": "completed",
        "total-time":12345,
        "setup-time":12345,
        "downtime":12345,
        "ram":{
          "transferred":123,
          "remaining":123,
          "total":246,
          "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",
         "total-time":12345,
         "setup-time":12345,
         "expected-downtime":12345,
         "ram":{
            "transferred":123,
            "remaining":123,
            "total":246,
            "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",
         "total-time":12345,
         "setup-time":12345,
         "expected-downtime":12345,
         "ram":{
            "total":1057024,
            "remaining":1053304,
            "transferred":3720,
            "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",
         "total-time":12345,
         "setup-time":12345,
         "expected-downtime":12345,
         "ram":{
            "total":1057024,
            "remaining":1053304,
            "transferred":3720,
            "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,
            "encoding-rate":80.1,
            "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. The capacity must have the same setting on both source and target or migration will not even start. 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)

pause−before−switchover
Pause outgoing migration before serialising device state and before disabling block IO (since 2.11)

multifd
Use more than one fd for migration (since 4.0)

dirty−bitmaps
If enabled, QEMU will migrate named dirty bitmaps. (since 2.12)

postcopy−blocktime
Calculate downtime for postcopy live migration (since 3.0)

late−block−activate
If enabled, the destination will not activate block devices (and thus take locks) immediately at the end of migration. (since 3.0)

x−ignore−shared
If enabled, QEMU will not migrate shared memory (since 4.0)

validate−uuid
Send the UUID of the source to allow the destination to ensure it is the same. (since 4.2)

background−snapshot
If enabled, the migration stream will be a snapshot of the VM exactly at the point when the migration procedure starts. The VM RAM is saved with running VM. (since 6.0)

Features

unstable
Members **x−colo** and **x−ignore−shared** are experimental.

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"}
   ]}
```

MultiFDCompression (Enum)
An enumeration of multifd compression methods.

Values

none no compression.

zlib use zlib compression method.

zstd (**If: CONFIG_ZSTD**)
use zstd compression method.

Since
5.0

BitmapMigrationBitmapAliasTransform (Object)
Members

persistent: boolean (optional)
If present, the bitmap will be made persistent or transient depending on this parameter.

Since
6.0

BitmapMigrationBitmapAlias (Object)
Members

name: string
The name of the bitmap.

alias: string
An alias name for migration (for example the bitmap name on the opposite site).

transform: BitmapMigrationBitmapAliasTransform (optional)
Allows the modification of the migrated bitmap. (since 6.0)

Since
5.2

BitmapMigrationNodeAlias (Object)
Maps a block node name and the bitmaps it has to aliases for dirty bitmap migration.

Members

node−name: string
A block node name.

alias: string
An alias block node name for migration (for example the node name on the opposite site).

bitmaps: array of BitmapMigrationBitmapAlias
Mappings for the bitmaps on this node.

Since
5.2

MigrationParameter (Enum)
Migration parameters enumeration

Values

announce−initial
Initial delay (in milliseconds) before sending the first announce (Since 4.0)

announce−max
Maximum delay (in milliseconds) between packets in the announcement (Since 4.0)

announce−rounds
Number of self−announce packets sent after migration (Since 4.0)

announce−step
Increase in delay (in milliseconds) between subsequent packets in the announcement (Since 4.0)

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.

compress−wait−thread
Controls behavior when all compression threads are currently busy. If true (default), wait for a free compression thread to become available; otherwise, send the page uncompressed. (Since 3.1)

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.

throttle−trigger−threshold
The ratio of bytes_dirty_period and bytes_xfer_period to trigger throttling. It is expressed as percentage. The default value is 50. (Since 5.0)

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)

cpu−throttle−tailslow
Make CPU throttling slower at tail stage At the tail stage of throttling, the Guest is very sensitive to CPU percentage while the **cpu−throttle** −increment is excessive usually at tail stage. If this parameter is true, we will compute the ideal CPU percentage used by the Guest, which may exactly make the dirty rate match the dirty rate threshold. Then we will choose a smaller throttle increment between the one specified by **cpu−throttle−increment** and the one generated by ideal CPU percentage. Therefore, it is compatible to traditional throttling, meanwhile the throttle increment won't be excessive at tail stage. The default value is false. (Since 5.1)

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)

tls−authz
: ID of the 'authz' object subclass that provides access control checking of the TLS x509 certificate distinguished name. This object is only resolved at time of use, so can be deleted and recreated on the fly while the migration server is active. If missing, it will default to denying access (Since 4.0)

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)

multifd−channels
: Number of channels used to migrate data in parallel. This is the same number that the number of sockets used for migration. The default value is 2 (since 4.0)

xbzrle−cache−size
: cache size to be used by XBZRLE migration. It needs to be a multiple of the target page size and a power of 2 (Since 2.11)

max−postcopy−bandwidth
: Background transfer bandwidth during postcopy. Defaults to 0 (unlimited). In bytes per second. (Since 3.0)

max−cpu−throttle
: maximum cpu throttle percentage. Defaults to 99. (Since 3.1)

multifd−compression
: Which compression method to use. Defaults to none. (Since 5.0)

multifd−zlib−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. Defaults to 1. (Since 5.0)

multifd−zstd−level
: Set the compression level to be used in live migration, the compression level is an integer between 0 and 20, where 0 means no compression, 1 means the best compression speed, and 20 means best compression ratio which will consume more CPU. Defaults to 1. (Since 5.0)

block−bitmap−mapping
: Maps block nodes and bitmaps on them to aliases for the purpose of dirty bitmap migration. Such aliases may for example be the corresponding names on the opposite site. The mapping must be one−to−one, but not necessarily complete: On the source, unmapped bitmaps and all bitmaps on unmapped nodes will be ignored. On the destination, encountering an unmapped alias in the incoming migration stream will result in a report, and all further bitmap migration data will then be discarded. Note that the destination does not know about bitmaps it does not receive, so there is no limitation or requirement regarding the number of bitmaps received, or how they are named, or on which nodes they are placed. By default (when this parameter has never been set), bitmap names are mapped to themselves. Nodes are mapped to their block device name if there is one, and to their node name otherwise. (Since 5.2)

Features

unstable

Member **x−checkpoint−delay** is experimental.

Since

2.4

MigrateSetParameters (Object)

Members

announce−initial: int (optional)

Initial delay (in milliseconds) before sending the first announce (Since 4.0)

announce−max: int (optional)

Maximum delay (in milliseconds) between packets in the announcement (Since 4.0)

announce−rounds: int (optional)

Number of self−announce packets sent after migration (Since 4.0)

announce−step: int (optional)

Increase in delay (in milliseconds) between subsequent packets in the announcement (Since 4.0)

compress−level: int (optional)

compression level

compress−threads: int (optional)

compression thread count

compress−wait−thread: boolean (optional)

Controls behavior when all compression threads are currently busy. If true (default), wait for a free compression thread to become available; otherwise, send the page uncompressed. (Since 3.1)

decompress−threads: int (optional)

decompression thread count

throttle−trigger−threshold: int (optional)

The ratio of bytes_dirty_period and bytes_xfer_period to trigger throttling. It is expressed as percentage. The default value is 50. (Since 5.0)

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)

cpu−throttle−tailslow: boolean (optional)

Make CPU throttling slower at tail stage At the tail stage of throttling, the Guest is very sensitive to CPU percentage while the **cpu−throttle** −increment is excessive usually at tail stage. If this parameter is true, we will compute the ideal CPU percentage used by the Guest, which may exactly make the dirty rate match the dirty rate threshold. Then we will choose a smaller throttle increment between the one specified by **cpu−throttle−increment** and the one generated by ideal CPU percentage. Therefore, it is compatible to traditional throttling, meanwhile the throttle increment won't be excessive at tail stage. The default value is false. (Since 5.1)

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)

multifd−channels: int (optional)
Number of channels used to migrate data in parallel. This is the same number that the number of sockets used for migration. The default value is 2 (since 4.0)

xbzrle−cache−size: int (optional)
cache size to be used by XBZRLE migration. It needs to be a multiple of the target page size and a power of 2 (Since 2.11)

max−postcopy−bandwidth: int (optional)
Background transfer bandwidth during postcopy. Defaults to 0 (unlimited). In bytes per second. (Since 3.0)

max−cpu−throttle: int (optional)
maximum cpu throttle percentage. The default value is 99. (Since 3.1)

multifd−compression: MultiFDCompression (optional)
Which compression method to use. Defaults to none. (Since 5.0)

multifd−zlib−level: int (optional)
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. Defaults to 1. (Since 5.0)

multifd−zstd−level: int (optional)
Set the compression level to be used in live migration, the compression level is an integer between 0 and 20, where 0 means no compression, 1 means the best compression speed, and 20 means best compression ratio which will consume more CPU. Defaults to 1. (Since 5.0)

block−bitmap−mapping: array of BitmapMigrationNodeAlias (optional)
Maps block nodes and bitmaps on them to aliases for the purpose of dirty bitmap migration. Such aliases may for example be the corresponding names on the opposite site. The mapping must be one−to−one, but not necessarily complete: On the source, unmapped bitmaps and all bitmaps on unmapped nodes will be ignored. On the destination, encountering an unmapped alias in the incoming migration stream will result in a report, and all further bitmap migration data will then be discarded. Note that the destination does not know about bitmaps it does not receive, so there is no limitation or requirement regarding the number of bitmaps received, or how they are named, or on which nodes they are placed. By default (when this parameter has never been set), bitmap names are mapped to themselves. Nodes are mapped to their block device name if there is one, and to their node name otherwise. (Since 5.2)

tls−authz: StrOrNull (optional)
Not documented

Features

unstable
Member **x−checkpoint−delay** is experimental.

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

announce−initial: int (optional)
Initial delay (in milliseconds) before sending the first announce (Since 4.0)

announce−max: int (optional)
Maximum delay (in milliseconds) between packets in the announcement (Since 4.0)

announce−rounds: int (optional)
Number of self−announce packets sent after migration (Since 4.0)

announce−step: int (optional)
Increase in delay (in milliseconds) between subsequent packets in the announcement (Since 4.0)

compress−level: int (optional)
compression level

compress−threads: int (optional)
compression thread count

compress−wait−thread: boolean (optional)
Controls behavior when all compression threads are currently busy. If true (default), wait for a free compression thread to become available; otherwise, send the page uncompressed. (Since 3.1)

decompress−threads: int (optional)
decompression thread count

throttle−trigger−threshold: int (optional)
The ratio of bytes_dirty_period and bytes_xfer_period to trigger throttling. It is expressed as percentage. The default value is 50. (Since 5.0)

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)

cpu−throttle−tailslow: boolean (optional)
Make CPU throttling slower at tail stage At the tail stage of throttling, the Guest is very sensitive to CPU percentage while the **cpu−throttle** −increment is excessive usually at tail stage. If this parameter is true, we will compute the ideal CPU percentage used by the Guest, which may exactly make the dirty rate match the dirty rate threshold. Then we will choose a smaller throttle increment between the one specified by **cpu−throttle−increment** and the one generated by ideal CPU percentage. Therefore, it is compatible to traditional throttling, meanwhile the throttle increment won't be excessive at tail stage. The default value is false. (Since 5.1)

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.

tls−authz: string (optional)
ID of the 'authz' object subclass that provides access control checking of the TLS x509 certificate distinguished name. (Since 4.0)

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)

multifd−channels: int (optional)
Number of channels used to migrate data in parallel. This is the same number that the number of sockets used for migration. The default value is 2 (since 4.0)

xbzrle−cache−size: int (optional)
cache size to be used by XBZRLE migration. It needs to be a multiple of the target page size and a power of 2 (Since 2.11)

max−postcopy−bandwidth: int (optional)
Background transfer bandwidth during postcopy. Defaults to 0 (unlimited). In bytes per second. (Since 3.0)

max−cpu−throttle: int (optional)
maximum cpu throttle percentage. Defaults to 99. (Since 3.1)

multifd−compression: MultiFDCompression (optional)
Which compression method to use. Defaults to none. (Since 5.0)

multifd−zlib−level: int (optional)
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. Defaults to 1. (Since 5.0)

multifd−zstd−level: int (optional)
Set the compression level to be used in live migration, the compression level is an integer between 0 and 20, where 0 means no compression, 1 means the best compression speed, and 20 means best compression ratio which will consume more CPU. Defaults to 1. (Since 5.0)

block−bitmap−mapping: array of BitmapMigrationNodeAlias (optional)
Maps block nodes and bitmaps on them to aliases for the purpose of dirty bitmap migration. Such aliases may for example be the corresponding names on the opposite site. The mapping must be one−to−one, but not necessarily complete: On the source, unmapped bitmaps and all bitmaps on unmapped nodes will be ignored. On the destination, encountering an unmapped alias in the incoming migration stream will result in a report, and all further bitmap migration data will then be discarded. Note that the destination does not know about bitmaps it does not receive, so there is no limitation or requirement regarding the number of bitmaps received, or how they are named, or on which nodes they are placed. By default (when this parameter has never been set), bitmap names are mapped to themselves. Nodes are mapped to their block device name if there is one, and to their node name otherwise. (Since 5.2)

Features
unstable
Member **x−checkpoint−delay** is experimental.

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

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 on both source and destination 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 current mode.

Values

none COLO is disabled.

primary
COLO node in primary side.

secondary
COLO node in 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

COLO_EXIT (Event)
Emitted when VM finishes COLO mode due to some errors happening or at the request of users.

Arguments

mode: COLOMode
report COLO mode when COLO exited.

reason: COLOExitReason
describes the reason for the COLO exit.

Since
3.1

Example

```
<- { "timestamp": {"seconds": 2032141960, "microseconds": 417172},
     "event": "COLO_EXIT", "data": {"mode": "primary", "reason": "request" } }
```

COLOExitReason (Enum)
The reason for a COLO exit.

Values

none failover has never happened. This state does not occur in the COLO_EXIT event, and is only visible in the result of query−colo−status.

request
COLO exit is due to an external request.

error COLO exit is due to an internal error.

processing
COLO is currently handling a failover (since 4.0).

Since
3.1

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.

Features

unstable
This command is experimental.

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

Example

```
-> { "execute": "migrate_cancel" }
<- { "return": {} }
```

migrate−continue (Command)
Continue migration when it's in a paused state.

Arguments

state: MigrationStatus
The state the migration is currently expected to be in

Returns
nothing on success

Since
2.11

Example

```
-> { "execute": "migrate−continue" , "arguments":
     { "state": "pre−switchover" } }
<- { "return": {} }
```

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

resume: boolean (optional)
resume one paused migration, default "off". (since 3.0)

Returns
nothing on success

Since
0.14

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.

live: boolean (optional)

Optional argument to ask QEMU to treat this command as part of a live migration. Default to true. (since 2.11)

Returns

Nothing on success

Since

1.1

Example

```
-> { "execute": "xen-save-devices-state",
     "arguments": { "filename": "/tmp/save" } }
<- { "return": {} }
```

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": {} }
```

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": {} }
```

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

If
CONFIG_REPLICATION

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

If
CONFIG_REPLICATION

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

If
: **CONFIG_REPLICATION**

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

If
CONFIG_REPLICATION

COLOStatus (Object)
The result format for 'query−colo−status'.

Members

mode: COLOMode
COLO running mode. If COLO is running, this field will return 'primary' or 'secondary'.

last−mode: COLOMode
COLO last running mode. If COLO is running, this field will return same like mode field, after failover we can use this field to get last colo mode. (since 4.0)

reason: COLOExitReason
describes the reason for the COLO exit.

Since
3.1

query−colo−status (Command)
Query COLO status while the vm is running.

Returns
A **COLOStatus** object showing the status.

Example

```
-> { "execute": "query−colo−status" }
<- { "return": { "mode": "primary", "reason": "request" } }
```

Since
3.1

migrate−recover (Command)
Provide a recovery migration stream URI.

Arguments

uri: string
the URI to be used for the recovery of migration stream.

Returns
nothing.

Example

```
-> { "execute": "migrate−recover",
     "arguments": { "uri": "tcp:192.168.1.200:12345" } }
<- { "return": {} }
```

Since
3.0

migrate−pause (Command)
Pause a migration. Currently it only supports postcopy.

Returns
nothing.

Example

```
-> { "execute": "migrate-pause" }
<- { "return": {} }
```

Since
3.0

UNPLUG_PRIMARY (Event)
Emitted from source side of a migration when migration state is WAIT_UNPLUG. Device was unplugged by guest operating system. Device resources in QEMU are kept on standby to be able to re−plug it in case of migration failure.

Arguments

device−id: string
QEMU device id of the unplugged device

Since
4.2

Example

```
{"event": "UNPLUG_PRIMARY", "data": {"device-id": "hostdev0"} }
```

DirtyRateVcpu (Object)
Dirty rate of vcpu.

Members

id: int vcpu index.

dirty−rate: int
dirty rate.

Since
6.2

DirtyRateStatus (Enum)
An enumeration of dirtyrate status.

Values

unstarted
the dirtyrate thread has not been started.

measuring
the dirtyrate thread is measuring.

measured
the dirtyrate thread has measured and results are available.

Since
5.2

DirtyRateMeasureMode (Enum)
An enumeration of mode of measuring dirtyrate.

Values

page−sampling
calculate dirtyrate by sampling pages.

dirty−ring
calculate dirtyrate by dirty ring.

dirty−bitmap
calculate dirtyrate by dirty bitmap.

Since
6.2

DirtyRateInfo (Object)
Information about current dirty page rate of vm.

Members

dirty−rate: int (optional)
an estimate of the dirty page rate of the VM in units of MB/s, present only when estimating the rate has completed.

status: DirtyRateStatus
status containing dirtyrate query status includes 'unstarted' or 'measuring' or 'measured'

start−time: int
start time in units of second for calculation

calc−time: int
time in units of second for sample dirty pages

sample−pages: int
page count per GB for sample dirty pages the default value is 512 (since 6.1)

mode: DirtyRateMeasureMode
mode containing method of calculate dirtyrate includes 'page−sampling' and 'dirty−ring' (Since 6.2)

vcpu−dirty−rate: array of DirtyRateVcpu (optional)
dirtyrate for each vcpu if dirty−ring mode specified (Since 6.2)

Since
5.2

calc−dirty−rate (Command)
start calculating dirty page rate for vm

Arguments

calc−time: int
time in units of second for sample dirty pages

sample−pages: int (optional)
page count per GB for sample dirty pages the default value is 512 (since 6.1)

mode: DirtyRateMeasureMode (optional)
mechanism of calculating dirtyrate includes 'page−sampling' and 'dirty−ring' (Since 6.1)

Since
5.2

Example

```
{"command": "calc−dirty−rate", "data": {"calc−time": 1,
                                          'sample−pages': 512} }
```

query−dirty−rate (Command)
query dirty page rate in units of MB/s for vm

Since
5.2

snapshot−save (Command)

Save a VM snapshot

Arguments

job−id: string
identifier for the newly created job

tag: string
name of the snapshot to create

vmstate: string
block device node name to save vmstate to

devices: array of string
list of block device node names to save a snapshot to

Applications should not assume that the snapshot save is complete when this command returns. The job commands / events must be used to determine completion and to fetch details of any errors that arise.

Note that execution of the guest CPUs may be stopped during the time it takes to save the snapshot. A future version of QEMU may ensure CPUs are executing continuously.

It is strongly recommended that **devices** contain all writable block device nodes if a consistent snapshot is required.

If **tag** already exists, an error will be reported

Returns

nothing

Example

```
-> { "execute": "snapshot−save",
     "data": {
        "job−id": "snapsave0",
        "tag": "my−snap",
        "vmstate": "disk0",
        "devices": ["disk0", "disk1"]
     }
   }
<- { "return": { } }
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "created", "id": "snapsave0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "running", "id": "snapsave0"}}
<- {"event": "STOP"}
<- {"event": "RESUME"}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "waiting", "id": "snapsave0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "pending", "id": "snapsave0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "concluded", "id": "snapsave0"}}
-> {"execute": "query−jobs"}
<- {"return": [{"current−progress": 1,
                "status": "concluded",
                "total−progress": 1,
                "type": "snapshot−save",
                "id": "snapsave0"}]}
```

Since
6.0

snapshot−load (Command)
Load a VM snapshot

Arguments

job−id: string
identifier for the newly created job

tag: string
name of the snapshot to load.

vmstate: string
block device node name to load vmstate from

devices: array of string
list of block device node names to load a snapshot from

Applications should not assume that the snapshot load is complete when this command returns. The job commands / events must be used to determine completion and to fetch details of any errors that arise.

Note that execution of the guest CPUs will be stopped during the time it takes to load the snapshot.

It is strongly recommended that **devices** contain all writable block device nodes that can have changed since the original **snapshot−save** command execution.

Returns
nothing

Example

```
-> { "execute": "snapshot-load",
     "data": {
        "job-id": "snapload0",
        "tag": "my-snap",
        "vmstate": "disk0",
        "devices": ["disk0", "disk1"]
     }
   }
<- { "return": { } }
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "created", "id": "snapload0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "running", "id": "snapload0"}}
<- {"event": "STOP"}
<- {"event": "RESUME"}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "waiting", "id": "snapload0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "pending", "id": "snapload0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "concluded", "id": "snapload0"}}
-> {"execute": "query-jobs"}
<- {"return": [{"current-progress": 1,
                "status": "concluded",
                "total-progress": 1,
                "type": "snapshot-load",
                "id": "snapload0"}]}
```

Since
6.0

snapshot−delete (Command)
Delete a VM snapshot

Arguments
job−id: string
identifier for the newly created job

tag: string
name of the snapshot to delete.

devices: array of string
list of block device node names to delete a snapshot from

Applications should not assume that the snapshot delete is complete when this command returns. The job commands / events must be used to determine completion and to fetch details of any errors that arise.

Returns
nothing

Example

```
-> { "execute": "snapshot-delete",
     "data": {
        "job-id": "snapdelete0",
        "tag": "my-snap",
        "devices": ["disk0", "disk1"]
     }
   }
<- { "return": { } }
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "created", "id": "snapdelete0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "running", "id": "snapdelete0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "waiting", "id": "snapdelete0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "pending", "id": "snapdelete0"}}
<- {"event": "JOB_STATUS_CHANGE",
    "data": {"status": "concluded", "id": "snapdelete0"}}
-> {"execute": "query-jobs"}
<- {"return": [{"current-progress": 1,
                "status": "concluded",
                "total-progress": 1,
                "type": "snapshot-delete",
                "id": "snapdelete0"}]}
```

Since
6.0

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

TransactionActionKind (Enum)

Values

abort Since 1.6

block−dirty−bitmap−add

Since 2.5

block−dirty−bitmap−remove

Since 4.2

block−dirty−bitmap−clear

Since 2.5

block−dirty−bitmap−enable

Since 4.0

block−dirty−bitmap−disable

Since 4.0

block−dirty−bitmap−merge

Since 4.0

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

Features

deprecated

Member **drive−backup** is deprecated. Use member **blockdev−backup** instead.

Since

1.1

AbortWrapper (Object)

Members

data: Abort

Not documented

Since
1.6

BlockDirtyBitmapAddWrapper (Object)
Members
data: BlockDirtyBitmapAdd
Not documented

Since
2.5

BlockDirtyBitmapWrapper (Object)
Members
data: BlockDirtyBitmap
Not documented

Since
2.5

BlockDirtyBitmapMergeWrapper (Object)
Members
data: BlockDirtyBitmapMerge
Not documented

Since
4.0

BlockdevBackupWrapper (Object)
Members
data: BlockdevBackup
Not documented

Since
2.3

BlockdevSnapshotWrapper (Object)
Members
data: BlockdevSnapshot
Not documented

Since
2.5

BlockdevSnapshotInternalWrapper (Object)
Members
data: BlockdevSnapshotInternal
Not documented

Since
1.7

BlockdevSnapshotSyncWrapper (Object)
Members
data: BlockdevSnapshotSync
Not documented

Since
1.1

DriveBackupWrapper (Object)
Members

data: DriveBackup
Not documented

Since
1.6

TransactionAction (Object)
A discriminated record of operations that can be performed with **transaction**.

Members

type: TransactionActionKind
Not documented

The members of AbortWrapper when type is "abort"

The members of BlockDirtyBitmapAddWrapper when type is "block−dirty−bitmap−add"

The members of BlockDirtyBitmapWrapper when type is "block−dirty−bitmap−remove"

The members of BlockDirtyBitmapWrapper when type is "block−dirty−bitmap−clear"

The members of BlockDirtyBitmapWrapper when type is "block−dirty−bitmap−enable"

The members of BlockDirtyBitmapWrapper when type is "block−dirty−bitmap−disable"

The members of BlockDirtyBitmapMergeWrapper when type is "block−dirty−bitmap−merge"

The members of BlockdevBackupWrapper when type is "blockdev−backup"

The members of BlockdevSnapshotWrapper when type is "blockdev−snapshot"

The members of BlockdevSnapshotInternalWrapper when type is "blockdev−snapshot−internal−sync"

The members of BlockdevSnapshotSyncWrapper when type is "blockdev−snapshot−sync"

The members of DriveBackupWrapper 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, and rbd,

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": {} }
```

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": {} }
```

COMPATIBILITY POLICY

CompatPolicyInput (Enum)
Policy for handling "funny" input.

Values

accept	Accept silently
reject	Reject with an error
crash	abort() the process

Since
6.0

CompatPolicyOutput (Enum)
Policy for handling "funny" output.

Values

accept	Pass on unchanged
hide	Filter out

Since
6.0

CompatPolicy (Object)
Policy for handling deprecated management interfaces.

This is intended for testing users of the management interfaces.

Limitation: covers only syntactic aspects of QMP, i.e. stuff tagged with feature 'deprecated'. We may want to extend it to cover semantic aspects, CLI, and experimental features.

Limitation: deprecated−output policy **hide** is not implemented for enumeration values. They behave the same as with policy **accept**.

Members

deprecated−input: CompatPolicyInput (optional)
how to handle deprecated input (default 'accept')

deprecated−output: CompatPolicyOutput (optional)
how to handle deprecated output (default 'accept')

unstable−input: CompatPolicyInput (optional)
how to handle unstable input (default 'accept') (since 6.2)

unstable−output: CompatPolicyOutput (optional)
how to handle unstable output (default 'accept') (since 6.2)

Since
6.0

QMP MONITOR CONTROL

qmp_capabilities (Command)

Enable QMP capabilities.

Arguments:

Arguments

enable: array of QMPCapability (optional)
An optional list of QMPCapability values to enable. The client must not enable any capability that is not mentioned in the QMP greeting message. If the field is not provided, it means no QMP capabilities will be enabled. (since 2.12)

Example

```
-> { "execute": "qmp_capabilities",
     "arguments": { "enable": [ "oob" ] } }
<- { "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)

The QMP client needs to explicitly enable QMP capabilities, otherwise all the QMP capabilities will be turned off by default.

Since
0.13

QMPCapability (Enum)

Enumeration of capabilities to be advertised during initial client connection, used for agreeing on particular QMP extension behaviors.

Values

oob	QMP ability to support out−of−band requests. (Please refer to qmp−spec.txt for more information on OOB)

Since
2.12

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

query−version (Command)
Returns the current version of QEMU.

Returns
A **VersionInfo** object describing the current version of QEMU.

Since
0.14

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

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

Example

```
-> { "execute": "query-commands" }
<- {
     "return":[
        {
           "name":"query-balloon"
        },
        {
           "name":"system_powerdown"
        }
     ]
   }
```

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

Example

```
-> { "execute": "quit" }
<- { "return": {} }
```

MonitorMode (Enum)

An enumeration of monitor modes.

Values

readline
HMP monitor (human−oriented command line interface)

control QMP monitor (JSON−based machine interface)

Since

5.0

MonitorOptions (Object)

Options to be used for adding a new monitor.

Members

id: string (optional)
Name of the monitor

mode: MonitorMode (optional)
Selects the monitor mode (default: readline in the system emulator, control in qemu−storage−daemon)

pretty: boolean (optional)
Enables pretty printing (QMP only)

chardev: string
Name of a character device to expose the monitor on

Since

5.0

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**.

features: array of string (optional)

names of features associated with the entity, in no particular order. (since 4.1 for object types, 4.2 for commands, 5.0 for the rest)

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 8259 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
members: array of SchemaInfoEnumMember
the enum type's members, in no particular order (since 6.2).

values: array of string
the enumeration type's member names, in no particular order. Redundant with**members**. Just for backward compatibility.

Features
deprecated
Member **values** is deprecated. Use **members** instead.
Values of this type are JSON string on the wire.

Since
2.5

SchemaInfoEnumMember (Object)
An object member.

Members

name: string
the member's name, as defined in the QAPI schema.

features: array of string (optional)
names of features associated with the member, in no particular order.

Since
6.2

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.

features: array of string (optional)
names of features associated with the member, in no particular order. (since 5.0)

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.

allow−oob: boolean (optional)
whether the command allows out−of−band execution, defaults to false (Since: 2.12)

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

QEMU OBJECT MODEL (QOM)

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.

description: string (optional)
if specified, the description of the property.

default−value: value (optional)
the default value, if any (since 5.0)

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

Example

```
-> { "execute": "qom−list",
     "arguments": { "path": "/chardevs" } }
<- { "return": [ { "name": "type", "type": "string" },
                 { "name": "parallel0", "type": "child<chardev−vc>" },
                 { "name": "serial0", "type": "child<chardev−vc>" },
                 { "name": "mon0", "type": "child<chardev−stdio>" } ] }
```

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

Example
```
1. Use absolute path

-> { "execute": "qom-get",
     "arguments": { "path": "/machine/unattached/device[0]",
                    "property": "hotplugged" } }
<- { "return": false }

2. Use partial path

-> { "execute": "qom-get",
     "arguments": { "path": "unattached/sysbus",
                    "property": "type" } }
<- { "return": "System" }
```

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

Example
```
-> { "execute": "qom-set",
     "arguments": { "path": "/machine",
                    "property": "graphics",
                    "value": false } }
<- { "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

qom−list−properties (Command)
List properties associated with a QOM object.

Arguments

typename: string
the type name of an object

Note
objects can create properties at runtime, for example to describe links between different devices and/or objects. These properties are not included in the output of this command.

Returns
a list of ObjectPropertyInfo describing object properties

Since
2.12

CanHostSocketcanProperties (Object)
Properties for can−host−socketcan objects.

Members

if: string
interface name of the host system CAN bus to connect to

canbus: string
object ID of the can−bus object to connect to the host interface

Since
2.12

ColoCompareProperties (Object)
Properties for colo−compare objects.

Members

primary_in: string
name of the character device backend to use for the primary input (incoming packets are redirected to **outdev**)

secondary_in: string
name of the character device backend to use for secondary input (incoming packets are only compared to the input on **primary_in** and then dropped)

outdev: string
name of the character device backend to use for output

iothread: string
name of the iothread to run in

notify_dev: string (optional)
name of the character device backend to be used to communicate with the remote colo−frame (only for Xen COLO)

compare_timeout: int (optional)
the maximum time to hold a packet from **primary_in** for comparison with an incoming packet on **secondary_in** in milliseconds (default: 3000)

expired_scan_cycle: int (optional)
the interval at which colo−compare checks whether packets from **primary** have timed out, in milliseconds (default: 3000)

max_queue_size: int (optional)
the maximum number of packets to keep in the queue for comparing with incoming packets from **secondary_in**. If the queue is full and additional packets are received, the additional packets are dropped. (default: 1024)

vnet_hdr_support: boolean (optional)
if true, vnet header support is enabled (default: false)

Since
2.8

CryptodevBackendProperties (Object)
Properties for cryptodev−backend and cryptodev−backend−builtin objects.

Members

queues: int (optional)
the number of queues for the cryptodev backend. Ignored for cryptodev−backend and must be 1 for cryptodev−backend−builtin. (default: 1)

Since
2.8

CryptodevVhostUserProperties (Object)
Properties for cryptodev−vhost−user objects.

Members

chardev: string
the name of a Unix domain socket character device that connects to the vhost−user server

The members of CryptodevBackendProperties

Since
2.12

DBusVMStateProperties (Object)
Properties for dbus−vmstate objects.

Members

addr: string
the name of the DBus bus to connect to

id−list: string (optional)
a comma separated list of DBus IDs of helpers whose data should be included in the VM state on migration

Since
5.0

NetfilterInsert (Enum)
Indicates where to insert a netfilter relative to a given other filter.

Values

before	insert before the specified filter
behind	insert behind the specified filter

Since
5.0

NetfilterProperties (Object)
Properties for objects of classes derived from netfilter.

Members

netdev: string
id of the network device backend to filter

queue: NetFilterDirection (optional)
indicates which queue(s) to filter (default: all)

status: string (optional)
indicates whether the filter is enabled ("on") or disabled ("off") (default: "on")

position: string (optional)
specifies where the filter should be inserted in the filter list. "head" means the filter is inserted at the head of the filter list, before any existing filters. "tail" means the filter is inserted at the tail of the filter list, behind any existing filters (default). "id=<id>" means the filter is inserted before or behind the filter specified by <id>, depending on the **insert** property. (default: "tail")

insert: NetfilterInsert (optional)
where to insert the filter relative to the filter given in **position**. Ignored if**position** is "head" or "tail". (default: behind)

Since
2.5

FilterBufferProperties (Object)
Properties for filter−buffer objects.

Members

interval: int
a non−zero interval in microseconds. All packets arriving in the given interval are delayed until the end of the interval.

The members of NetfilterProperties

Since
2.5

FilterDumpProperties (Object)
Properties for filter−dump objects.

Members

file: string
the filename where the dumped packets should be stored

maxlen: int (optional)
maximum number of bytes in a packet that are stored (default: 65536)

The members of NetfilterProperties

Since
2.5

FilterMirrorProperties (Object)
Properties for filter−mirror objects.

Members

outdev: string
the name of a character device backend to which all incoming packets are mirrored

vnet_hdr_support: boolean (optional)
if true, vnet header support is enabled (default: false)

The members of NetfilterProperties

Since
2.6

FilterRedirectorProperties (Object)
Properties for filter−redirector objects.

At least one of **indev** or **outdev** must be present. If both are present, they must not refer to the same character device backend.

Members

indev: string (optional)
the name of a character device backend from which packets are received and redirected to the filtered network device

outdev: string (optional)
the name of a character device backend to which all incoming packets are redirected

vnet_hdr_support: boolean (optional)
if true, vnet header support is enabled (default: false)

The members of NetfilterProperties

Since
2.6

FilterRewriterProperties (Object)
Properties for filter−rewriter objects.

Members

vnet_hdr_support: boolean (optional)
if true, vnet header support is enabled (default: false)

The members of NetfilterProperties

Since
2.8

InputBarrierProperties (Object)
Properties for input−barrier objects.

Members

name: string
the screen name as declared in the screens section of barrier.conf

server: string (optional)
hostname of the Barrier server (default: "localhost")

port: string (optional)
TCP port of the Barrier server (default: "24800")

x−origin: string (optional)
x coordinate of the leftmost pixel on the guest screen (default: "0")

y−origin: string (optional)
y coordinate of the topmost pixel on the guest screen (default: "0")

width: string (optional)
the width of secondary screen in pixels (default: "1920")

height: string (optional)
the height of secondary screen in pixels (default: "1080")

Since
4.2

InputLinuxProperties (Object)
Properties for input−linux objects.

Members

evdev: string
the path of the host evdev device to use

grab_all: boolean (optional)
if true, grab is toggled for all devices (e.g. both keyboard and mouse) instead of just one device (default: false)

repeat: boolean (optional)
enables auto−repeat events (default: false)

grab−toggle: GrabToggleKeys (optional)
the key or key combination that toggles device grab (default: ctrl−ctrl)

Since
2.6

IothreadProperties (Object)
Properties for iothread objects.

Members

poll−max−ns: int (optional)
the maximum number of nanoseconds to busy wait for events. 0 means polling is disabled (default: 32768 on POSIX hosts, 0 otherwise)

poll−grow: int (optional)
the multiplier used to increase the polling time when the algorithm detects it is missing events due to not polling long enough. 0 selects a default behaviour (default: 0)

poll−shrink: int (optional)
the divisor used to decrease the polling time when the algorithm detects it is spending too long polling without encountering events. 0 selects a default behaviour (default: 0)

aio−max−batch: int (optional)
maximum number of requests in a batch for the AIO engine, 0 means that the engine will use its default (default:0, since 6.1)

Since
2.0

MemoryBackendProperties (Object)
Properties for objects of classes derived from memory−backend.

Members

merge: boolean (optional)
if true, mark the memory as mergeable (default depends on the machine type)

dump: boolean (optional)
    if true, include the memory in core dumps (default depends on the machine type)

host−nodes: array of int (optional)
    the list of NUMA host nodes to bind the memory to

policy: HostMemPolicy (optional)
    the NUMA policy (default: 'default')

prealloc: boolean (optional)
    if true, preallocate memory (default: false)

prealloc−threads: int (optional)
    number of CPU threads to use for prealloc (default: 1)

share: boolean (optional)
    if false, the memory is private to QEMU; if true, it is shared (default: false)

reserve: boolean (optional)
    if true, reserve swap space (or huge pages) if applicable (default: true) (since 6.1)

size: int
    size of the memory region in bytes

x−use−canonical−path−for−ramblock−id: boolean (optional)
    if true, the canoncial path is used for ramblock−id. Disable this for 4.0 machine types or older to allow migration with newer QEMU versions. (default: false generally, but true for machine types <= 4.0)

Note

prealloc=true and reserve=false cannot be set at the same time. With reserve=true, the behavior depends on the operating system: for example, Linux will not reserve swap space for shared file mappings −− "not applicable". In contrast, reserve=false will bail out if it cannot be configured accordingly.

Since

2.1

MemoryBackendFileProperties (Object)

Properties for memory−backend−file objects.

Members

align: int (optional)
    the base address alignment when QEMU mmap(2)s **mem−path**. Some backend stores specified by **mem−path** require an alignment different than the default one used by QEMU, e.g. the device DAX /dev/dax0.0 requires 2M alignment rather than 4K. In such cases, users can specify the required alignment via this option. 0 selects a default alignment (currently the page size). (default: 0)

discard−data: boolean (optional)
    if true, the file contents can be destroyed when QEMU exits, to avoid unnecessarily flushing data to the backing file. Note that **discard−data** is only an optimization, and QEMU might not discard file contents if it aborts unexpectedly or is terminated using SIGKILL. (default: false)

mem−path: string
    the path to either a shared memory or huge page filesystem mount

pmem: **boolean** (optional) (**If: CONFIG_LIBPMEM**)
    specifies whether the backing file specified by **mem−path** is in host persistent memory that can be accessed using the SNIA NVM programming model (e.g. Intel NVDIMM).

readonly: boolean (optional)
    if true, the backing file is opened read−only; if false, it is opened read−write. (default: false)

The members of MemoryBackendProperties

Since
2.1

MemoryBackendMemfdProperties (Object)
Properties for memory−backend−memfd objects.

The **share** boolean option is true by default with memfd.

Members

hugetlb: boolean (optional)
if true, the file to be created resides in the hugetlbfs filesystem (default: false)

hugetlbsize: int (optional)
the hugetlb page size on systems that support multiple hugetlb page sizes (it must be a power of 2 value supported by the system). 0 selects a default page size. This option is ignored if **hugetlb** is false. (default: 0)

seal: boolean (optional)
if true, create a sealed−file, which will block further resizing of the memory (default: true)

The members of MemoryBackendProperties

Since
2.12

MemoryBackendEpcProperties (Object)
Properties for memory−backend−epc objects.

The **share** boolean option is true by default with epc

The **merge** boolean option is false by default with epc

The **dump** boolean option is false by default with epc

Members

The members of MemoryBackendProperties

Since
6.2

PrManagerHelperProperties (Object)
Properties for pr−manager−helper objects.

Members

path: string
the path to a Unix domain socket for connecting to the external helper

Since
2.11

QtestProperties (Object)
Properties for qtest objects.

Members

chardev: string
the chardev to be used to receive qtest commands on.

log: string (optional)
the path to a log file

Since
6.0

RemoteObjectProperties (Object)
Properties for x−remote−object objects.

Members

fd: string
file descriptor name previously passed via 'getfd' command

devid: string
the id of the device to be associated with the file descriptor

Since
6.0

RngProperties (Object)
Properties for objects of classes derived from rng.

Members

opened: boolean (optional)
if true, the device is opened immediately when applying this option and will probably fail when processing the next option. Don't use; only provided for compatibility. (default: false)

Features

deprecated
Member **opened** is deprecated. Setting true doesn't make sense, and false is already the default.

Since
1.3

RngEgdProperties (Object)
Properties for rng−egd objects.

Members

chardev: string
the name of a character device backend that provides the connection to the RNG daemon

The members of RngProperties

Since
1.3

RngRandomProperties (Object)
Properties for rng−random objects.

Members

filename: string (optional)
the filename of the device on the host to obtain entropy from (default: "/dev/urandom")

The members of RngProperties

Since
1.3

SevGuestProperties (Object)
Properties for sev−guest objects.

Members

sev−device: string (optional)
SEV device to use (default: "/dev/sev")

dh−cert−file: string (optional)
guest owners DH certificate (encoded with base64)

session−file: string (optional)
guest owners session parameters (encoded with base64)

policy: int (optional)
SEV policy value (default: 0x1)

handle: int (optional)
SEV firmware handle (default: 0)

cbitpos: int (optional)
C−bit location in page table entry (default: 0)

reduced−phys−bits: int
number of bits in physical addresses that become unavailable when SEV is enabled

kernel−hashes: boolean (optional)
if true, add hashes of kernel/initrd/cmdline to a designated guest firmware page for measured boot with −kernel (default: false) (since 6.2)

Since
2.12

ObjectType (Enum)
Values

authz−list
Not documented

authz−listfile
Not documented

authz−pam
Not documented

authz−simple
Not documented

can−bus
Not documented

can−host−socketcan (**If: CONFIG_LINUX**)
Not documented

colo−compare
Not documented

cryptodev−backend
Not documented

cryptodev−backend−builtin
Not documented

cryptodev−vhost−user (**If: CONFIG_VHOST_CRYPTO**)
Not documented

dbus−vmstate
Not documented

filter−buffer
Not documented

filter−dump
Not documented

filter−mirror
Not documented

filter−redirector
Not documented

filter−replay
Not documented

filter−rewriter
Not documented

input−barrier
Not documented

input−linux (**If: CONFIG_LINUX**)
Not documented

iothread
Not documented

memory−backend−epc (**If: CONFIG_LINUX**)
Not documented

memory−backend−file
Not documented

memory−backend−memfd (**If: CONFIG_LINUX**)
Not documented

memory−backend−ram
Not documented

pef−guest
Not documented

pr−manager−helper (**If: CONFIG_LINUX**)
Not documented

qtest Not documented

rng−builtin
Not documented

rng−egd
Not documented

rng−random (**If: CONFIG_POSIX**)
Not documented

secret Not documented

secret_keyring (**If: CONFIG_SECRET_KEYRING**)
Not documented

sev−guest
Not documented

s390−pv−guest
Not documented

throttle−group
Not documented

tls−creds−anon
Not documented

tls−creds−psk
Not documented

tls−creds−x509
Not documented

tls−cipher−suites
Not documented

x−remote−object
Not documented

Features

unstable
Member **x−remote−object** is experimental.

Since
6.0

ObjectOptions (Object)
Describes the options of a user creatable QOM object.

Members

qom−type: ObjectType
the class name for the object to be created

id: string
the name of the new object

The members of AuthZListProperties when qom−type is "authz−list"

The members of AuthZListFileProperties when qom−type is "authz−listfile"

The members of AuthZPAMProperties when qom−type is "authz−pam"

The members of AuthZSimpleProperties when qom−type is "authz−simple"

The members of **CanHostSocketcanProperties** when **qom−type** is **"can−host−socketcan"** (**If: CONFIG_LINUX**)

The members of ColoCompareProperties when qom−type is "colo−compare"

The members of CryptodevBackendProperties when qom−type is "cryptodev−backend"

The members of CryptodevBackendProperties when qom−type is "cryptodev−backend−builtin"

The members of **CryptodevVhostUserProperties** when **qom−type** is **"cryptodev−vhost−user"** (**If: CONFIG_VHOST_CRYPTO**)

The members of DBusVMStateProperties when qom−type is "dbus−vmstate"

The members of FilterBufferProperties when qom−type is "filter−buffer"

The members of FilterDumpProperties when qom−type is "filter−dump"

The members of FilterMirrorProperties when qom−type is "filter−mirror"

The members of FilterRedirectorProperties when qom−type is "filter−redirector"

The members of NetfilterProperties when qom−type is "filter−replay"

The members of FilterRewriterProperties when qom−type is "filter−rewriter"

The members of InputBarrierProperties when qom−type is "input−barrier"

The members of **InputLinuxProperties** when **qom−type** is **"input−linux"** (**If: CONFIG_LINUX**)

The members of IothreadProperties when qom−type is "iothread"

The members of **MemoryBackendEpcProperties** when **qom−type** is **"memory−backend−epc"** (**If: CONFIG_LINUX**)

The members of MemoryBackendFileProperties when qom−type is "memory−backend−file"

The members of **MemoryBackendMemfdProperties** when **qom−type** is **"memory−backend−memfd"** (**If: CONFIG_LINUX**)

The members of MemoryBackendProperties when qom−type is "memory−backend−ram"

The members of **PrManagerHelperProperties** when **qom−type** is **"pr−manager−helper"** (**If: CONFIG_LINUX**)

The members of QtestProperties when qom−type is "qtest"

The members of RngProperties when qom−type is "rng−builtin"

The members of RngEgdProperties when qom−type is "rng−egd"

The members of **RngRandomProperties** when **qom−type** is **"rng−random"** (**If: CONFIG_POSIX**)

The members of SecretProperties when qom−type is "secret"

The members of **SecretKeyringProperties** when **qom−type** is **"secret_keyring"** (**If: CONFIG_SECRET_KEYRING**)

The members of SevGuestProperties when qom−type is "sev−guest"

The members of ThrottleGroupProperties when qom−type is "throttle−group"

The members of TlsCredsAnonProperties when qom−type is "tls−creds−anon"

The members of TlsCredsPskProperties when qom−type is "tls−creds−psk"

The members of TlsCredsX509Properties when qom−type is "tls−creds−x509"

The members of TlsCredsProperties when qom−type is "tls−cipher−suites"

The members of RemoteObjectProperties when qom−type is "x−remote−object"

Since
6.0

object−add (Command)
Create a QOM object.

Arguments
The members of ObjectOptions

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",
                    "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": {} }
```

DEVICE INFRASTRUCTURE (QDEV)

device−list−properties (Command)

List properties associated with a device.

Arguments

typename: string
the type name of a device

Returns

a list of ObjectPropertyInfo describing a devices properties

Note

objects can create properties at runtime, for example to describe links between different devices and/or objects. These properties are not included in the output of this command.

Since

1.2

device_add (Command)

Add a device.

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

Features

json−cli
If present, the "−device" command line option supports JSON syntax with a structure identical to the arguments of this command.

Notes

Additional arguments depend on the type.

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. If a guest−side error in the hot removal process is detected, the device will not be removed and a DEVICE_UNPLUG_GUEST_ERROR event is sent. Some errors cannot be detected.

Since
0.14

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)
the device's ID if it has one

path: string
the device's QOM 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 } }
```

DEVICE_UNPLUG_GUEST_ERROR (Event)
Emitted when a device hot unplug fails due to a guest reported error.

Arguments
device: string (optional)
the device's ID if it has one

path: string
the device's QOM path

Since
6.2

Example
```
<- { "event": "DEVICE_UNPLUG_GUEST_ERROR"
     "data": { "device": "core1",
               "path": "/machine/peripheral/core1" },
     },
     "timestamp": { "seconds": 1615570772, "microseconds": 202844 } }
```

MACHINES

SysEmuTarget (Enum)

The comprehensive enumeration of QEMU system emulation ("softmmu") targets. Run "./configure −−help" in the project root directory, and look for the *−softmmu targets near the "−−target−list" option. The individual target constants are not documented here, for the time being.

Values

rx	since 5.0
avr	since 5.1
aarch64	Not documented
alpha	Not documented
arm	Not documented
cris	Not documented
hppa	Not documented
i386	Not documented
m68k	Not documented
microblaze	Not documented
microblazeel	Not documented
mips	Not documented
mips64	Not documented
mips64el	Not documented
mipsel	Not documented
nios2	Not documented
or1k	Not documented
ppc	Not documented
ppc64	Not documented
riscv32	Not documented
riscv64	Not documented
s390x	Not documented
sh4	Not documented
sh4eb	Not documented
sparc	Not documented
sparc64	Not documented
tricore	Not documented
x86_64	Not documented

xtensa Not documented

xtensaeb
Not documented

Notes
The resulting QMP strings can be appended to the "qemu−system−" prefix to produce the corresponding QEMU executable name. This is true even for "qemu−system−x86_64".

Since
3.0

CpuS390State (Enum)
An enumeration of cpu states that can be assumed by a virtual S390 CPU

Values

uninitialized
Not documented

stopped
Not documented

check−stop
Not documented

operating
Not documented

load Not documented

Since
2.12

CpuInfoS390 (Object)
Additional information about a virtual S390 CPU

Members

cpu−state: CpuS390State
the virtual CPU's state

Since
2.12

CpuInfoFast (Object)
Information about a virtual CPU

Members

cpu−index: int
index of the virtual CPU

qom−path: string
path to the CPU object in the QOM tree

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

target: SysEmuTarget
the QEMU system emulation target, which determines which additional fields will be listed (since 3.0)

The members of CpuInfoS390 when target is "s390x"

Since
2.12

query−cpus−fast (Command)
Returns information about all virtual CPUs.

Returns
list of **CpuInfoFast**

Since
2.12

Example

```
-> { "execute": "query-cpus-fast" }
<- { "return": [
        {
            "thread-id": 25627,
            "props": {
                "core-id": 0,
                "thread-id": 0,
                "socket-id": 0
            },
            "qom-path": "/machine/unattached/device[0]",
            "arch":"x86",
            "target":"x86_64",
            "cpu-index": 0
        },
        {
            "thread-id": 25628,
            "props": {
                "core-id": 0,
                "thread-id": 0,
                "socket-id": 1
            },
            "qom-path": "/machine/unattached/device[2]",
            "arch":"x86",
            "target":"x86_64",
            "cpu-index": 1
        }
    ]
}
```

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)

hotpluggable−cpus: boolean
cpu hotplug via −device is supported (since 2.7)

numa−mem−supported: boolean
true if '−numa node,mem' option is supported by the machine type and false otherwise (since 4.1)

deprecated: boolean
if true, the machine type is deprecated and may be removed in future versions of QEMU according to the QEMU deprecation policy (since 4.1)

default−cpu−type: string (optional)
default CPU model typename if none is requested via the −cpu argument. (since 4.2)

default−ram−id: string (optional)
the default ID of initial RAM memory backend (since 5.2)

Since
1.2

query−machines (Command)
Return a list of supported machines

Returns
a list of MachineInfo

Since
1.2

CurrentMachineParams (Object)
Information describing the running machine parameters.

Members
wakeup−suspend−support: boolean
true if the machine supports wake up from suspend

Since
4.0

query−current−machine (Command)
Return information on the current virtual machine.

Returns
CurrentMachineParams

Since
4.0

TargetInfo (Object)
Information describing the QEMU target.

Members
arch: SysEmuTarget
the target architecture

Since
1.2

query−target (Command)
Return information about the target for this QEMU

Returns
TargetInfo

Since
1.2

UuidInfo (Object)
Guest UUID information (Universally Unique Identifier).

Members
UUID: string
the UUID of the guest

Since
0.14

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

Example

```
    -> { "execute": "query-uuid" }
    <- { "return": { "UUID": "550e8400-e29b-41d4-a716-446655440000" } }
```

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

system_reset (Command)
Performs a hard reset of a guest.

Since
0.14

Example

```
    -> { "execute": "system_reset" }
    <- { "return": {} }
```

system_powerdown (Command)
Requests that a guest perform a powerdown operation.

Since
0.14

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": {} }
```

system_wakeup (Command)

Wake up guest from suspend. If the guest has wake−up from suspend support enabled (wakeup−suspend−support flag from query−current−machine), wake−up guest from suspend if the guest is in SUSPENDED state. Return an error otherwise.

Since

1.1

Returns

nothing.

Note

prior to 4.0, this command does nothing in case the guest isn't suspended.

Example

```
-> { "execute": "system_wakeup" }
<- { "return": {} }
```

LostTickPolicy (Enum)

Policy for handling lost ticks in timer devices. Ticks end up getting lost when, for example, the guest is paused.

Values

discard

throw away the missed ticks and continue with future injection normally. The guest OS will see the timer jump ahead by a potentially quite significant amount all at once, as if the intervening chunk of time had simply not existed; needless to say, such a sudden jump can easily confuse a guest OS which is not specifically prepared to deal with it. Assuming the guest OS can deal correctly with the time jump, the time in the guest and in the host should now match.

delay continue to deliver ticks at the normal rate. The guest OS will not notice anything is amiss, as from its point of view time will have continued to flow normally. The time in the guest should now be behind the time in the host by exactly the amount of time during which ticks have been missed.

slew deliver ticks at a higher rate to catch up with the missed ticks. The guest OS will not notice anything is amiss, as from its point of view time will have continued to flow normally. Once the timer has managed to catch up with all the missing ticks, the time in the guest and in the host should match.

Since

2.0

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

Note

prior to 2.1, this command was only supported for x86 and s390 VMs

Example

```
-> { "execute": "inject-nmi" }
<- { "return": {} }
```

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

query−kvm (Command)
Returns information about KVM acceleration

Returns
KvmInfo

Since
0.14

Example

```
-> { "execute": "query-kvm" }
<- { "return": { "enabled": true, "present": true } }
```

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)

hmat−lb
memory latency and bandwidth information (Since: 5.0)

hmat−cache
memory side cache information (Since: 5.0)

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"

The members of NumaHmatLBOptions when type is "hmat−lb"

The members of NumaHmatCacheOptions when type is "hmat−cache"

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.

initiator: int (optional)
defined in ACPI 6.3 Chapter 5.2.27.3 Table 5−145, points to the nodeid which has the memory controller responsible for this NUMA node. This field provides additional information as to the initiator node that is closest (as in directly attached) to this node, and therefore has the best performance (since 5.0)

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

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

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

HmatLBMemoryHierarchy (Enum)
The memory hierarchy in the System Locality Latency and Bandwidth Information Structure of HMAT (Heterogeneous Memory Attribute Table)

For more information about **HmatLBMemoryHierarchy**, see chapter 5.2.27.4: Table 5−146: Field "Flags" of ACPI 6.3 spec.

Values
memory
the structure represents the memory performance

first−level
first level of memory side cache

second−level
second level of memory side cache

third−level
third level of memory side cache

Since
5.0

HmatLBDataType (Enum)
Data type in the System Locality Latency and Bandwidth Information Structure of HMAT (Heterogeneous Memory Attribute Table)

For more information about **HmatLBDataType**, see chapter 5.2.27.4: Table 5−146: Field "Data Type" of ACPI 6.3 spec.

Values
access−latency
access latency (nanoseconds)

read−latency
read latency (nanoseconds)

write−latency
write latency (nanoseconds)

access−bandwidth
access bandwidth (Bytes per second)

read−bandwidth
read bandwidth (Bytes per second)

write−bandwidth
write bandwidth (Bytes per second)

Since
5.0

NumaHmatLBOptions (Object)
Set the system locality latency and bandwidth information between Initiator and Target proximity Domains.

For more information about **NumaHmatLBOptions**, see chapter 5.2.27.4: Table 5−146 of ACPI 6.3 spec.

Members

initiator: int
the Initiator Proximity Domain.

target: int
the Target Proximity Domain.

hierarchy: HmatLBMemoryHierarchy
the Memory Hierarchy. Indicates the performance of memory or side cache.

data−type: HmatLBDataType
presents the type of data, access/read/write latency or hit latency.

latency: int (optional)
the value of latency from **initiator** to **target** proximity domain, the latency unit is "ns(nanosecond)".

bandwidth: int (optional)
the value of bandwidth between **initiator** and **target** proximity domain, the bandwidth unit is "Bytes per second".

Since
5.0

HmatCacheAssociativity (Enum)
Cache associativity in the Memory Side Cache Information Structure of HMAT

For more information of **HmatCacheAssociativity**, see chapter 5.2.27.5: Table 5−147 of ACPI 6.3 spec.

Values

none

None (no memory side cache in this proximity domain,
or cache associativity unknown)

direct Direct Mapped

complex
Complex Cache Indexing (implementation specific)

Since
5.0

HmatCacheWritePolicy (Enum)

Cache write policy in the Memory Side Cache Information Structure of HMAT

For more information of **HmatCacheWritePolicy**, see chapter 5.2.27.5: Table 5−147: Field "Cache Attributes" of ACPI 6.3 spec.

Values

none None (no memory side cache in this proximity domain, or cache write policy unknown)

write−back
Write Back (WB)

write−through
Write Through (WT)

Since

5.0

NumaHmatCacheOptions (Object)

Set the memory side cache information for a given memory domain.

For more information of **NumaHmatCacheOptions**, see chapter 5.2.27.5: Table 5−147: Field "Cache Attributes" of ACPI 6.3 spec.

Members

node−id: int
the memory proximity domain to which the memory belongs.

size: int
the size of memory side cache in bytes.

level: int
the cache level described in this structure.

associativity: HmatCacheAssociativity
the cache associativity, none/direct−mapped/complex(complex cache indexing).

policy: HmatCacheWritePolicy
the write policy, none/write−back/write−through.

line: int
the cache Line size in bytes.

Since

5.0

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

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

Notes
Errors were not reliably returned until 1.1

Example
```
-> { "execute": "pmemsave",
     "arguments": { "val": 10,
                    "size": 100,
                    "filename": "/tmp/physical-mem-dump" } }
<- { "return": {} }
```

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
whether memory merge support is enabled

dump: boolean
whether memory backend's memory is included in a core dump

prealloc: boolean
whether memory was preallocated

share: boolean
whether memory is private to QEMU or shared (since 6.1)

reserve: boolean (optional)
whether swap space (or huge pages) was reserved if applicable. This corresponds to the user configuration and not the actual behavior implemented in the OS to perform the reservation. For example, Linux will never reserve swap space for shared file mappings. (since 6.1)

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"
       }
     ]
   }
```

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

die−id: int (optional)
die number within node/board the CPU belongs to (Since 4.1)

core−id: int (optional)
core number within die the CPU belongs to

thread−id: int (optional)
thread number within core the CPU belongs to

Note
currently there are 5 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)
TODO
Better documentation; currently there is none.

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}
     }
   ]}
```

```
For s390x−virtio−ccw machine type started with −smp 1,maxcpus=2 −cpu qemu
(Since: 2.11):

−> { "execute": "query−hotpluggable−cpus" }
<− {"return": [
     {
        "type": "qemu−s390x−cpu", "vcpus−count": 1,
        "props": { "core−id": 1 }
     },
     {
        "qom−path": "/machine/unattached/device[0]",
        "type": "qemu−s390x−cpu", "vcpus−count": 1,
        "props": { "core−id": 0 }
     }
   ]}
```

set−numa−node (Command)

Runtime equivalent of '−numa' CLI option, available at preconfigure stage to configure numa mapping before initializing machine.

Since 3.0

Arguments

The members of NumaOptions

balloon (Command)

Request the balloon driver to change its balloon size.

Arguments

value: int

the target logical size of the VM in bytes. We can deduce the size of the balloon using this formula:

logical_vm_size = vm_ram_size − balloon_size

From it we have: balloon_size = vm_ram_size − **value**

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

Example

```
−> { "execute": "balloon", "arguments": { "value": 536870912 } }
<− { "return": {} }
```

```
With a 2.5GiB guest this command inflated the ballon to 3GiB.
```

BalloonInfo (Object)

Information about the guest balloon device.

Members

actual: int

the logical size of the VM in bytes Formula used: logical_vm_size = vm_ram_size − balloon_size

Since

0.14

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

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

the logical size of the VM in bytes Formula used: logical_vm_size = vm_ram_size − balloon_size

Note

this event is rate−limited.

Since

1.2

Example

```
<- { "event": "BALLOON_CHANGE",
     "data": { "actual": 944766976 },
     "timestamp": { "seconds": 1267020223, "microseconds": 435656 } }
```

MemoryInfo (Object)

Actual memory information in bytes.

Members

base−memory: int

size of "base" memory specified with command line option −m.

plugged−memory: int (optional)

size of memory that can be hot−unplugged. This field is omitted if target doesn't support memory hotplug (i.e. CONFIG_MEM_DEVICE not defined at build time).

Since
2.11

query−memory−size−summary (Command)
Return the amount of initially allocated and present hotpluggable (if enabled) memory in bytes.

Example

```
-> { "execute": "query-memory-size-summary" }
<- { "return": { "base-memory": 4294967296, "plugged-memory": 0 } }
```

Since
2.11

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

VirtioPMEMDeviceInfo (Object)
VirtioPMEM state information

Members

id: string (optional)
device's ID

memaddr: int
physical address in memory, where device is mapped

size: int
size of memory that the device provides

memdev: string
memory backend linked with device

Since
4.1

VirtioMEMDeviceInfo (Object)
VirtioMEMDevice state information

Members

id: string (optional)
device's ID

memaddr: int
physical address in memory, where device is mapped

requested−size: int
the user requested size of the device

size: int
the (current) size of memory that the device provides

max−size: int
the maximum size of memory that the device can provide

block−size: int
the block size of memory that the device provides

node: int
NUMA node number where device is assigned to

memdev: string
memory backend linked with the region

Since
5.1

SgxEPCDeviceInfo (Object)
Sgx EPC state information

Members

id: string (optional)
device's ID

memaddr: int
physical address in memory, where device is mapped

size: int
size of memory that the device provides

memdev: string
memory backend linked with device

Since
6.2

MemoryDeviceInfoKind (Enum)
Values

dimm Not documented

nvdimm
Not documented

virtio−pmem
Not documented

virtio−mem
Not documented

sgx−epc
Not documented

Since
2.1

PCDIMMDeviceInfoWrapper (Object)
Members
data: PCDIMMDeviceInfo
Not documented

Since
2.1

VirtioPMEMDeviceInfoWrapper (Object)
Members
data: VirtioPMEMDeviceInfo
Not documented

Since
2.1

VirtioMEMDeviceInfoWrapper (Object)
Members
data: VirtioMEMDeviceInfo
Not documented

Since
2.1

SgxEPCDeviceInfoWrapper (Object)
Members
data: SgxEPCDeviceInfo
Not documented

Since
6.2

MemoryDeviceInfo (Object)
Union containing information about a memory device

nvdimm is included since 2.12. virtio−pmem is included since 4.1. virtio−mem is included since 5.1. sgx−epc is included since 6.2.

Members
type: MemoryDeviceInfoKind
Not documented

The members of PCDIMMDeviceInfoWrapper when type is "dimm"

The members of PCDIMMDeviceInfoWrapper when type is "nvdimm"

The members of VirtioPMEMDeviceInfoWrapper when type is "virtio−pmem"

The members of VirtioMEMDeviceInfoWrapper when type is "virtio−mem"

The members of SgxEPCDeviceInfoWrapper when type is "sgx−epc"

Since
2.1

SgxEPC (Object)
Sgx EPC cmdline information

Members
memdev: string
memory backend linked with device

Since
6.2

SgxEPCProperties (Object)
SGX properties of machine types.

Members
sgx−epc: array of SgxEPC
list of ids of memory−backend−epc objects.

Since
6.2

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"
                } ] }
```

MEMORY_DEVICE_SIZE_CHANGE (Event)
Emitted when the size of a memory device changes. Only emitted for memory devices that can actually change the size (e.g., virtio−mem due to guest action).

Arguments
id: string (optional)
device's ID

size: int
the new size of memory that the device provides

qom−path: string
path to the device object in the QOM tree (since 6.2)

Note
this event is rate−limited.

Since
5.1

Example
```
<- { "event": "MEMORY_DEVICE_SIZE_CHANGE",
     "data": { "id": "vm0", "size": 1073741824},
     "timestamp": { "seconds": 1588168529, "microseconds": 201316 } }
```

MEM_UNPLUG_ERROR (Event)
Emitted when memory hot unplug error occurs.

Arguments
device: string
device name

msg: string
Informative message

Features

deprecated
This event is deprecated. Use **DEVICE_UNPLUG_GUEST_ERROR** instead.

Since
2.4

Example

```
<- { "event": "MEM_UNPLUG_ERROR"
     "data": { "device": "dimm1",
               "msg": "acpi: device unplug for unsupported device"
     },
     "timestamp": { "seconds": 1265044230, "microseconds": 450486 } }
```

SMPConfiguration (Object)
Schema for CPU topology configuration. A missing value lets QEMU figure out a suitable value based on the ones that are provided.

Members

cpus: int (optional)
number of virtual CPUs in the virtual machine

sockets: int (optional)
number of sockets in the CPU topology

dies: int (optional)
number of dies per socket in the CPU topology

cores: int (optional)
number of cores per die in the CPU topology

threads: int (optional)
number of threads per core in the CPU topology

maxcpus: int (optional)
maximum number of hotpluggable virtual CPUs in the virtual machine

Since
6.1

x−query−irq (Command)
Query interrupt statistics

Features

unstable
This command is meant for debugging.

Returns
interrupt statistics

Since
6.2

x−query−jit (Command)
Query TCG compiler statistics

Features

unstable
This command is meant for debugging.

Returns
TCG compiler statistics

Since
6.2

If
CONFIG_TCG

x−query−numa (Command)
Query NUMA topology information

Features
unstable
This command is meant for debugging.

Returns
topology information

Since
6.2

x−query−opcount (Command)
Query TCG opcode counters

Features
unstable
This command is meant for debugging.

Returns
TCG opcode counters

Since
6.2

If
CONFIG_TCG

x−query−profile (Command)
Query TCG profiling information

Features
unstable
This command is meant for debugging.

Returns
profile information

Since
6.2

x−query−ramblock (Command)
Query system ramblock information

Features
unstable
This command is meant for debugging.

Returns
system ramblock information

Since
6.2

x−query−rdma (Command)
Query RDMA state

Features
unstable
This command is meant for debugging.

Returns
RDMA state

Since
6.2

x−query−roms (Command)
Query information on the registered ROMS

Features
unstable
This command is meant for debugging.

Returns
registered ROMs

Since
6.2

x−query−usb (Command)
Query information on the USB devices

Features
unstable
This command is meant for debugging.

Returns
USB device information

Since
6.2

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

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 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. The**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

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

CpuModelBaselineInfo (Object)

The result of a CPU model baseline.

Members

model: CpuModelInfo
the baselined CpuModelInfo.

Since

2.8

If

TARGET_S390X

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

If
TARGET_S390X

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.

Note
this command isn't specific to s390x, but is only implemented on this architecture currently.

Since
2.8

If
TARGET_S390X

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.

Note
this command isn't specific to s390x, but is only implemented on this architecture currently.

Since
2.8

If
TARGET_S390X

CpuModelExpansionInfo (Object)
The result of a cpu model expansion.

Members

model: CpuModelInfo
the expanded CpuModelInfo.

Since
2.8

If
TARGET_S390X or TARGET_I386 or TARGET_ARM

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". Arm only supports "full".

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

If

TARGET_S390X or TARGET_I386 or TARGET_ARM

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)

alias−of: string (optional)
Name of CPU model this model is an alias for. The target of the CPU model alias may change depending on the machine type. Management software is supposed to translate CPU model aliases in the VM configuration, because aliases may stop being migration−safe in the future (since 4.1)

deprecated: boolean
If true, this CPU model is deprecated and may be removed in in some future version of QEMU according to the QEMU deprecation policy. (since 5.2)

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

If

TARGET_PPC or TARGET_ARM or TARGET_I386 or TARGET_S390X or TARGET_MIPS

query−cpu−definitions (Command)

Return a list of supported virtual CPU definitions

Returns

a list of CpuDefInfo

Since

1.2

If

TARGET_PPC or TARGET_ARM or TARGET_I386 or TARGET_S390X or TARGET_MIPS

RECORD/REPLAY

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

ReplayInfo (Object)

Record/replay information.

Members

mode: ReplayMode

current mode.

filename: string (optional)

name of the record/replay log file. It is present only in record or replay modes, when the log is recorded or replayed.

icount: int

current number of executed instructions.

Since

5.2

query−replay (Command)

Retrieve the record/replay information. It includes current instruction count which may be used for **replay−break** and **replay−seek** commands.

Returns

record/replay information.

Since
5.2

Example
```
-> { "execute": "query-replay" }
<- { "return": { "mode": "play", "filename": "log.rr", "icount": 220414 } }
```

replay−break (Command)
Set replay breakpoint at instruction count **icount**. Execution stops when the specified instruction is reached. There can be at most one breakpoint. When breakpoint is set, any prior one is removed. The breakpoint may be set only in replay mode and only "in the future", i.e. at instruction counts greater than the current one. The current instruction count can be observed with **query−replay**.

Arguments
icount: int
instruction count to stop at

Since
5.2

Example
```
-> { "execute": "replay-break", "data": { "icount": 220414 } }
```

replay−delete−break (Command)
Remove replay breakpoint which was set with **replay−break**. The command is ignored when there are no replay breakpoints.

Since
5.2

Example
```
-> { "execute": "replay-delete-break" }
```

replay−seek (Command)
Automatically proceed to the instruction count **icount**, when replaying the execution. The command automatically loads nearest snapshot and replays the execution to find the desired instruction. When there is no preceding snapshot or the execution is not replayed, then the command fails. icount for the reference may be obtained with **query−replay** command.

Arguments
icount: int
target instruction count

Since
5.2

Example
```
-> { "execute": "replay-seek", "data": { "icount": 220414 } }
```

YANK FEATURE

YankInstanceType (Enum)
An enumeration of yank instance types. See **YankInstance** for more information.

Values
block−node
Not documented

chardev
Not documented

migration
Not documented

Since
6.0

YankInstanceBlockNode (Object)
Specifies which block graph node to yank. See **YankInstance** for more information.

Members
node−name: string
the name of the block graph node

Since
6.0

YankInstanceChardev (Object)
Specifies which character device to yank. See **YankInstance** for more information.

Members
id: string
the chardev's ID

Since
6.0

YankInstance (Object)
A yank instance can be yanked with the **yank** qmp command to recover from a hanging QEMU.

Currently implemented yank instances:

- nbd block device: Yanking it will shut down the connection to the nbd server without attempting to reconnect.
- socket chardev: Yanking it will shut down the connected socket.
- migration: Yanking it will shut down all migration connections. Unlike **migrate_cancel**, it will not notify the migration process, so migration will go into **failed** state, instead of **cancelled** state. **yank** should be used to recover from hangs.

Members
type: YankInstanceType
Not documented

The members of YankInstanceBlockNode when type is "block−node"

The members of YankInstanceChardev when type is "chardev"

Since
6.0

yank (Command)
Try to recover from hanging QEMU by yanking the specified instances. See **YankInstance** for more information.

Takes a list of **YankInstance** as argument.

Arguments
instances: array of YankInstance
Not documented

Returns
- Nothing on success
- **DeviceNotFound** error, if any of the YankInstances doesn't exist

Example
```
-> { "execute": "yank",
     "arguments": {
```

```
                  "instances": [
                       { "type": "block-node",
                         "node-name": "nbd0" }
                  ] } }
         <- { "return": {} }
```

Since

6.0

query−yank (Command)

Query yank instances. See **YankInstance** for more information.

Returns

list of **YankInstance**

Example

```
         -> { "execute": "query-yank" }
         <- { "return": [
                  { "type": "block-node",
                    "node-name": "nbd0" }
              ] }
```

Since

6.0

MISCELLANEA

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

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

query−name (Command)
Return the name information of a guest.

Returns
NameInfo of the guest

Since
0.14

Example
```
-> { "execute": "query-name" }
<- { "return": { "name": "qemu-name" } }
```

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)

aio−max−batch: int
maximum number of requests in a batch for the AIO engine, 0 means that the engine will use its default (since 6.1)

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
```

```
                  }
              ]
          }
```

stop (Command)
Stop all guest VCPU execution.

Since
0.14

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": {} }
```

cont (Command)
Resume guest VCPU execution.

Since
0.14

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": {} }
```

x−exit−preconfig (Command)
Exit from "preconfig" state

This command makes QEMU exit the preconfig state and proceed with VM initialization using configuration data provided on the command line and via the QMP monitor during the preconfig state. The command is only available during the preconfig state (i.e. when the −−preconfig command line option was in use).

Features
unstable
This command is experimental.
Since 3.0

Returns
nothing

Example

```
-> { "execute": "x-exit-preconfig" }
<- { "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

Features

savevm−monitor−nodes
If present, HMP command savevm only snapshots monitor−owned nodes if they have no parents. This allows the use of 'savevm' with −blockdev. (since 4.2)

Returns
the output of the command as a string

Since
0.14

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" }
```

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

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

Example

```
-> { "execute": "closefd", "arguments": { "fdname": "fd1" } }
<- { "return": {} }
```

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

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

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

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

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

query−fdsets (Command)

Return information describing all fd sets.

Returns

A list of **FdsetInfo**

Since

1.2

Note

The list of fd sets is shared by all monitor connections.

Example

```
-> { "execute": "query-fdsets" }
<- { "return": [
       {
         "fds": [
           {
             "fd": 30,
             "opaque": "rdonly:/path/to/file"
           },
           {
             "fd": 24,
             "opaque": "rdwr:/path/to/file"
           }
         ],
         "fdset-id": 1
       },
       {
```

```
                  "fds": [
                    {
                      "fd": 28
                    },
                    {
                      "fd": 29
                    }
                  ],
                  "fdset-id": 0
                }
              ]
            }
```

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"
         }
      ]
   }
```

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

Example

```
<-   { "event": "RTC_CHANGE",
       "data": { "offset": 78 },
       "timestamp": { "seconds": 1267020223, "microseconds": 435656 } }
```

If

TARGET_ALPHA or TARGET_ARM or TARGET_HPPA or TARGET_I386 or TARGET_MIPS or TARGET_MIPS64 or TARGET_PPC or TARGET_PPC64 or TARGET_S390X or TARGET_SH4 or TARGET_SPARC

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": {} }
```

If

TARGET_I386

SevState (Enum)

An enumeration of SEV state information used during **query−sev**.

Values

uninit The guest is uninitialized.

launch−update
The guest is currently being launched; plaintext data and register state is being imported.

launch−secret
The guest is currently being launched; ciphertext data is being imported.

running
The guest is fully launched or migrated in.

send−update
The guest is currently being migrated out to another machine.

receive−update
The guest is currently being migrated from another machine.

Since

2.12

If

TARGET_I386

SevInfo (Object)

Information about Secure Encrypted Virtualization (SEV) support

Members

enabled: boolean
true if SEV is active

api−major: int
SEV API major version

api−minor: int
SEV API minor version

build−id: int
SEV FW build id

policy: int
SEV policy value

state: SevState
SEV guest state

handle: int
SEV firmware handle

Since

2.12

If

TARGET_I386

query−sev (Command)

Returns information about SEV

Returns

SevInfo

Since

2.12

Example

```
-> { "execute": "query-sev" }
<- { "return": { "enabled": true, "api-major" : 0, "api-minor" : 0,
                 "build-id" : 0, "policy" : 0, "state" : "running",
                 "handle" : 1 } }
```

If

TARGET_I386

SevLaunchMeasureInfo (Object)

SEV Guest Launch measurement information

Members

data: string

the measurement value encoded in base64

Since

2.12

If

TARGET_I386

query−sev−launch−measure (Command)

Query the SEV guest launch information.

Returns

The **SevLaunchMeasureInfo** for the guest

Since

2.12

Example

```
-> { "execute": "query-sev-launch-measure" }
<- { "return": { "data": "4l8LXeNlSPUDlXPJG5966/8%YZ" } }
```

If

TARGET_I386

SevCapability (Object)

The struct describes capability for a Secure Encrypted Virtualization feature.

Members

pdh: string

Platform Diffie−Hellman key (base64 encoded)

cert−chain: string

PDH certificate chain (base64 encoded)

cbitpos: int

C−bit location in page table entry

reduced−phys−bits: int

Number of physical Address bit reduction when SEV is enabled

Since

2.12

If
    TARGET_I386

query−sev−capabilities (Command)
    This command is used to get the SEV capabilities, and is supported on AMD X86 platforms only.

Returns
    SevCapability objects.

Since
    2.12

Example

```
-> { "execute": "query-sev-capabilities" }
<- { "return": { "pdh": "8CCDD8DDD", "cert-chain": "888CCCDDDEE",
                 "cbitpos": 47, "reduced-phys-bits": 5}}
```

If
    TARGET_I386

sev−inject−launch−secret (Command)
    This command injects a secret blob into memory of SEV guest.

Arguments
    packet−header: string
        the launch secret packet header encoded in base64

    secret: string
        the launch secret data to be injected encoded in base64

    gpa: int (optional)
        the guest physical address where secret will be injected.

Since
    6.0

If
    TARGET_I386

SevAttestationReport (Object)
    The struct describes attestation report for a Secure Encrypted Virtualization feature.

Members
    data: string
        guest attestation report (base64 encoded)

Since
    6.1

If
    TARGET_I386

query−sev−attestation−report (Command)
    This command is used to get the SEV attestation report, and is supported on AMD X86 platforms only.

Arguments
    mnonce: string
        a random 16 bytes value encoded in base64 (it will be included in report)

Returns
    SevAttestationReport objects.

Since
    6.1

Example

```
-> { "execute" : "query-sev-attestation-report",
                 "arguments": { "mnonce": "aaaaaaa" } }
<- { "return" : { "data": "aaaaaaaabbbddddd"} }
```

If

TARGET_I386

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": {} }
```

If

TARGET_S390X

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

If

TARGET_ARM

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 } ] }
```

If

TARGET_ARM

SGXInfo (Object)
Information about intel Safe Guard eXtension (SGX) support

Members

sgx: boolean
true if SGX is supported

sgx1: boolean
true if SGX1 is supported

sgx2: boolean
true if SGX2 is supported

flc: boolean
true if FLC is supported

section−size: int
The EPC section size for guest

Since
6.2

If
TARGET_I386

query−sgx (Command)
Returns information about SGX

Returns
SGXInfo

Since
6.2

Example

```
-> { "execute": "query-sgx" }
<- { "return": { "sgx": true, "sgx1" : true, "sgx2" : true,
                 "flc": true, "section-size" : 0 } }
```

If
TARGET_I386

query−sgx−capabilities (Command)
Returns information from host SGX capabilities

Returns
SGXInfo

Since
6.2

Example

```
-> { "execute": "query-sgx-capabilities" }
<- { "return": { "sgx": true, "sgx1" : true, "sgx2" : true,
                 "flc": true, "section-size" : 0 } }
```

If
TARGET_I386

AUDIO

AudiodevPerDirectionOptions (Object)
General audio backend options that are used for both playback and recording.

Members

mixing−engine: boolean (optional)
use QEMU's mixing engine to mix all streams inside QEMU and convert audio formats when not supported by the backend. When set to off, fixed−settings must be also off (default on, since 4.2)

fixed−settings: boolean (optional)
use fixed settings for host input/output. When off, frequency, channels and format must not be specified (default true)

frequency: int (optional)
frequency to use when using fixed settings (default 44100)

channels: int (optional)
number of channels when using fixed settings (default 2)

voices: int (optional)
number of voices to use (default 1)

format: AudioFormat (optional)
sample format to use when using fixed settings (default s16)

buffer−length: int (optional)
the buffer length in microseconds

Since
4.0

AudiodevGenericOptions (Object)
Generic driver−specific options.

Members

in: AudiodevPerDirectionOptions (optional)
options of the capture stream

out: AudiodevPerDirectionOptions (optional)
options of the playback stream

Since
4.0

AudiodevAlsaPerDirectionOptions (Object)
Options of the ALSA backend that are used for both playback and recording.

Members

dev: string (optional)
the name of the ALSA device to use (default 'default')

period−length: int (optional)
the period length in microseconds

try−poll: boolean (optional)
attempt to use poll mode, falling back to non−polling access on failure (default true)

The members of AudiodevPerDirectionOptions

Since
4.0

AudiodevAlsaOptions (Object)
Options of the ALSA audio backend.

Members

in: AudiodevAlsaPerDirectionOptions (optional)
options of the capture stream

out: AudiodevAlsaPerDirectionOptions (optional)
options of the playback stream

threshold: int (optional)
set the threshold (in microseconds) when playback starts

Since
4.0

AudiodevCoreaudioPerDirectionOptions (Object)
Options of the Core Audio backend that are used for both playback and recording.

Members
buffer−count: int (optional)
number of buffers

The members of AudiodevPerDirectionOptions

Since
4.0

AudiodevCoreaudioOptions (Object)
Options of the coreaudio audio backend.

Members
in: AudiodevCoreaudioPerDirectionOptions (optional)
options of the capture stream

out: AudiodevCoreaudioPerDirectionOptions (optional)
options of the playback stream

Since
4.0

AudiodevDsoundOptions (Object)
Options of the DirectSound audio backend.

Members
in: AudiodevPerDirectionOptions (optional)
options of the capture stream

out: AudiodevPerDirectionOptions (optional)
options of the playback stream

latency: int (optional)
add extra latency to playback in microseconds (default 10000)

Since
4.0

AudiodevJackPerDirectionOptions (Object)
Options of the JACK backend that are used for both playback and recording.

Members
server−name: string (optional)
select from among several possible concurrent server instances (default: environment variable $JACK_DEFAULT_SERVER if set, else "default")

client−name: string (optional)
the client name to use. The server will modify this name to create a unique variant, if needed unless **exact−name** is true (default: the guest's name)

connect−ports: string (optional)
if set, a regular expression of JACK client port name(s) to monitor for and automatically connect to

start−server: boolean (optional)
start a jack server process if one is not already present (default: false)

exact−name: boolean (optional)
use the exact name requested otherwise JACK automatically generates a unique one, if needed (default: false)

The members of AudiodevPerDirectionOptions

Since
5.1

AudiodevJackOptions (Object)
Options of the JACK audio backend.

Members

in: AudiodevJackPerDirectionOptions (optional)
options of the capture stream

out: AudiodevJackPerDirectionOptions (optional)
options of the playback stream

Since
5.1

AudiodevOssPerDirectionOptions (Object)
Options of the OSS backend that are used for both playback and recording.

Members

dev: string (optional)
file name of the OSS device (default '/dev/dsp')

buffer−count: int (optional)
number of buffers

try−poll: boolean (optional)
attempt to use poll mode, falling back to non−polling access on failure (default true)

The members of AudiodevPerDirectionOptions

Since
4.0

AudiodevOssOptions (Object)
Options of the OSS audio backend.

Members

in: AudiodevOssPerDirectionOptions (optional)
options of the capture stream

out: AudiodevOssPerDirectionOptions (optional)
options of the playback stream

try−mmap: boolean (optional)
try using memory−mapped access, falling back to non−memory−mapped access on failure (default true)

exclusive: boolean (optional)
open device in exclusive mode (vmix won't work) (default false)

dsp−policy: int (optional)
set the timing policy of the device (between 0 and 10, where smaller number means smaller latency but higher CPU usage) or −1 to use fragment mode (option ignored on some platforms) (default 5)

Since
4.0

AudiodevPaPerDirectionOptions (Object)
Options of the Pulseaudio backend that are used for both playback and recording.

Members

name: string (optional)
name of the sink/source to use

stream−name: string (optional)
name of the PulseAudio stream created by qemu. Can be used to identify the stream in PulseAudio when you create multiple PulseAudio devices or run multiple qemu instances (default: audiodev's id, since 4.2)

latency: int (optional)
latency you want PulseAudio to achieve in microseconds (default 15000)

The members of AudiodevPerDirectionOptions

Since
4.0

AudiodevPaOptions (Object)
Options of the PulseAudio audio backend.

Members

in: AudiodevPaPerDirectionOptions (optional)
options of the capture stream

out: AudiodevPaPerDirectionOptions (optional)
options of the playback stream

server: string (optional)
PulseAudio server address (default: let PulseAudio choose)

Since
4.0

AudiodevSdlPerDirectionOptions (Object)
Options of the SDL audio backend that are used for both playback and recording.

Members

buffer−count: int (optional)
number of buffers (default 4)

The members of AudiodevPerDirectionOptions

Since
6.0

AudiodevSdlOptions (Object)
Options of the SDL audio backend.

Members

in: AudiodevSdlPerDirectionOptions (optional)
options of the recording stream

out: AudiodevSdlPerDirectionOptions (optional)
options of the playback stream

Since
6.0

AudiodevWavOptions (Object)
Options of the wav audio backend.

Members

in: AudiodevPerDirectionOptions (optional)
options of the capture stream

out: AudiodevPerDirectionOptions (optional)
options of the playback stream

path: string (optional)
name of the wav file to record (default 'qemu.wav')

Since
4.0

AudioFormat (Enum)
An enumeration of possible audio formats.

Values

u8	unsigned 8 bit integer
s8	signed 8 bit integer
u16	unsigned 16 bit integer
s16	signed 16 bit integer
u32	unsigned 32 bit integer
s32	signed 32 bit integer
f32	single precision floating−point (since 5.0)

Since
4.0

AudiodevDriver (Enum)
An enumeration of possible audio backend drivers.

Values

jack	JACK audio backend (since 5.1)
none	Not documented
alsa	Not documented
coreaudio	Not documented
dsound	Not documented
oss	Not documented
pa	Not documented
sdl	Not documented
spice	Not documented
wav	Not documented

Since
4.0

Audiodev (Object)
Options of an audio backend.

Members

id: string
identifier of the backend

driver: AudiodevDriver
the backend driver to use

timer−period: int (optional)
timer period (in microseconds, 0: use lowest possible)

The members of AudiodevGenericOptions when driver is "none"

The members of AudiodevAlsaOptions when driver is "alsa"

The members of AudiodevCoreaudioOptions when driver is "coreaudio"

The members of AudiodevDsoundOptions when driver is "dsound"

The members of AudiodevJackOptions when driver is "jack"

The members of AudiodevOssOptions when driver is "oss"

The members of AudiodevPaOptions when driver is "pa"

The members of AudiodevSdlOptions when driver is "sdl"

The members of AudiodevGenericOptions when driver is "spice"

The members of AudiodevWavOptions when driver is "wav"

Since
4.0

ACPI

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

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":
                 { "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
OSPM Status Indication

Since
2.1

Example

```
<- { "event": "ACPI_DEVICE_OST",
     "data": { "device": "d1", "slot": "0",
               "slot-type": "DIMM", "source": 1, "status": 0 } }
```

PCI

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

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

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

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

subsystem: int (optional)
the PCI subsystem id (since 3.1)

subsystem−vendor: int (optional)
the PCI subsystem vendor id (since 3.1)

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

irq_pin: int
: the IRQ pin, zero means no IRQ (since 5.1)

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

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

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

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.

COPYRIGHT

2022, The QEMU Project Developers