NAME

qemu-storage-daemon-qmp-ref − QEMU Storage Daemon QMP Reference Manual

Contents

- *QCryptoBlockAmendOptions (Object)*
- *SecretCommonProperties (Object)*
- *SecretProperties (Object)*
- *SecretKeyringProperties (Object)*
- *TlsCredsProperties (Object)*
- *TlsCredsAnonProperties (Object)*
- *TlsCredsPskProperties (Object)*
- *TlsCredsX509Properties (Object)*
- *Background jobs*

- *Socket data types*
  - *NetworkAddressFamily (Enum)*
  - *InetSocketAddressBase (Object)*
  - *InetSocketAddress (Object)*
  - *UnixSocketAddress (Object)*
  - *VsockSocketAddress (Object)*
  - *InetSocketAddressWrapper (Object)*
  - *UnixSocketAddressWrapper (Object)*
  - *VsockSocketAddressWrapper (Object)*
  - *StringWrapper (Object)*
  - *SocketAddressLegacy (Object)*
  - *SocketAddressType (Enum)*
  - *SocketAddress (Object)*
  - *SnapshotInfo (Object)*
  - *ImageInfoSpecificQCow2EncryptionBase (Object)*
  - *ImageInfoSpecificQCow2Encryption (Object)*
  - *ImageInfoSpecificQCow2 (Object)*
  - *ImageInfoSpecificVmdk (Object)*
  - *ImageInfoSpecificRbd (Object)*
  - *ImageInfoSpecificKind (Enum)*
  - *ImageInfoSpecificQCow2Wrapper (Object)*
  - *ImageInfoSpecificVmdkWrapper (Object)*
  - *ImageInfoSpecificLUKSWrapper (Object)*
  - *ImageInfoSpecificRbdWrapper (Object)*
  - *ImageInfoSpecific (Object)*
  - *ImageInfo (Object)*
  - *ImageCheck (Object)*
  - *MapEntry (Object)*
  - *BlockdevCacheInfo (Object)*

- *BlockDeviceInfo (Object)*
- *BlockDeviceIoStatus (Enum)*
- *BlockDirtyInfo (Object)*
- *Qcow2BitmapInfoFlags (Enum)*
- *Qcow2BitmapInfo (Object)*
- *BlockLatencyHistogramInfo (Object)*
- *BlockInfo (Object)*
- *BlockMeasureInfo (Object)*
- *query−block (Command)*
- *BlockDeviceTimedStats (Object)*
- *BlockDeviceStats (Object)*
- *BlockStatsSpecificFile (Object)*
- *BlockStatsSpecificNvme (Object)*
- *BlockStatsSpecific (Object)*
- *BlockStats (Object)*
- *query−blockstats (Command)*
- *BlockdevOnError (Enum)*
- *MirrorSyncMode (Enum)*
- *BitmapSyncMode (Enum)*
- *MirrorCopyMode (Enum)*
- *BlockJobInfo (Object)*
- *query−block−jobs (Command)*
- *block_resize (Command)*
- *NewImageMode (Enum)*
- *BlockdevSnapshotSync (Object)*
- *BlockdevSnapshot (Object)*
- *BackupPerf (Object)*
- *BackupCommon (Object)*
- *DriveBackup (Object)*
- *BlockdevBackup (Object)*
- *blockdev−snapshot−sync (Command)*
- *blockdev−snapshot (Command)*
- *change−backing−file (Command)*
- *block−commit (Command)*
- *drive−backup (Command)*
- *blockdev−backup (Command)*
- *query−named−block−nodes (Command)*
- *XDbgBlockGraphNodeType (Enum)*

- *XDbgBlockGraphNode (Object)*
- *BlockPermission (Enum)*
- *XDbgBlockGraphEdge (Object)*
- *XDbgBlockGraph (Object)*
- *x−debug−query−block−graph (Command)*
- *drive−mirror (Command)*
- *DriveMirror (Object)*
- *BlockDirtyBitmap (Object)*
- *BlockDirtyBitmapAdd (Object)*
- *BlockDirtyBitmapMergeSource (Alternate)*
- *BlockDirtyBitmapMerge (Object)*
- *block−dirty−bitmap−add (Command)*
- *block−dirty−bitmap−remove (Command)*
- *block−dirty−bitmap−clear (Command)*
- *block−dirty−bitmap−enable (Command)*
- *block−dirty−bitmap−disable (Command)*
- *block−dirty−bitmap−merge (Command)*
- *BlockDirtyBitmapSha256 (Object)*
- *x−debug−block−dirty−bitmap−sha256 (Command)*
- *blockdev−mirror (Command)*
- *BlockIOThrottle (Object)*
- *ThrottleLimits (Object)*
- *ThrottleGroupProperties (Object)*
- *block−stream (Command)*
- *block−job−set−speed (Command)*
- *block−job−cancel (Command)*
- *block−job−pause (Command)*
- *block−job−resume (Command)*
- *block−job−complete (Command)*
- *block−job−dismiss (Command)*
- *block−job−finalize (Command)*
- *BlockdevDiscardOptions (Enum)*
- *BlockdevDetectZeroesOptions (Enum)*
- *BlockdevAioOptions (Enum)*
- *BlockdevCacheOptions (Object)*
- *BlockdevDriver (Enum)*
- *BlockdevOptionsFile (Object)*
- *BlockdevOptionsNull (Object)*

- *BlockdevOptionsNVMe (Object)*
- *BlockdevOptionsVVFAT (Object)*
- *BlockdevOptionsGenericFormat (Object)*
- *BlockdevOptionsLUKS (Object)*
- *BlockdevOptionsGenericCOWFormat (Object)*
- *Qcow2OverlapCheckMode (Enum)*
- *Qcow2OverlapCheckFlags (Object)*
- *Qcow2OverlapChecks (Alternate)*
- *BlockdevQcowEncryptionFormat (Enum)*
- *BlockdevQcowEncryption (Object)*
- *BlockdevOptionsQcow (Object)*
- *BlockdevQcow2EncryptionFormat (Enum)*
- *BlockdevQcow2Encryption (Object)*
- *BlockdevOptionsPreallocate (Object)*
- *BlockdevOptionsQcow2 (Object)*
- *SshHostKeyCheckMode (Enum)*
- *SshHostKeyCheckHashType (Enum)*
- *SshHostKeyHash (Object)*
- *SshHostKeyCheck (Object)*
- *BlockdevOptionsSsh (Object)*
- *BlkdebugEvent (Enum)*
- *BlkdebugIOType (Enum)*
- *BlkdebugInjectErrorOptions (Object)*
- *BlkdebugSetStateOptions (Object)*
- *BlockdevOptionsBlkdebug (Object)*
- *BlockdevOptionsBlklogwrites (Object)*
- *BlockdevOptionsBlkverify (Object)*
- *BlockdevOptionsBlkreplay (Object)*
- *QuorumReadPattern (Enum)*
- *BlockdevOptionsQuorum (Object)*
- *BlockdevOptionsGluster (Object)*
- *IscsiTransport (Enum)*
- *IscsiHeaderDigest (Enum)*
- *BlockdevOptionsIscsi (Object)*
- *RbdAuthMode (Enum)*
- *RbdImageEncryptionFormat (Enum)*
- *RbdEncryptionOptionsLUKSBase (Object)*
- *RbdEncryptionCreateOptionsLUKSBase (Object)*

- *RbdEncryptionOptionsLUKS (Object)*
- *RbdEncryptionOptionsLUKS2 (Object)*
- *RbdEncryptionCreateOptionsLUKS (Object)*
- *RbdEncryptionCreateOptionsLUKS2 (Object)*
- *RbdEncryptionOptions (Object)*
- *RbdEncryptionCreateOptions (Object)*
- *BlockdevOptionsRbd (Object)*
- *ReplicationMode (Enum)*
- *BlockdevOptionsReplication (Object)*
- *NFSTransport (Enum)*
- *NFSServer (Object)*
- *BlockdevOptionsNfs (Object)*
- *BlockdevOptionsCurlBase (Object)*
- *BlockdevOptionsCurlHttp (Object)*
- *BlockdevOptionsCurlHttps (Object)*
- *BlockdevOptionsCurlFtp (Object)*
- *BlockdevOptionsCurlFtps (Object)*
- *BlockdevOptionsNbd (Object)*
- *BlockdevOptionsRaw (Object)*
- *BlockdevOptionsThrottle (Object)*
- *BlockdevOptionsCor (Object)*
- *BlockdevOptionsCbw (Object)*
- *BlockdevOptions (Object)*
- *BlockdevRef (Alternate)*
- *BlockdevRefOrNull (Alternate)*
- *blockdev−add (Command)*
- *blockdev−reopen (Command)*
- *blockdev−del (Command)*
- *BlockdevCreateOptionsFile (Object)*
- *BlockdevCreateOptionsGluster (Object)*
- *BlockdevCreateOptionsLUKS (Object)*
- *BlockdevCreateOptionsNfs (Object)*
- *BlockdevCreateOptionsParallels (Object)*
- *BlockdevCreateOptionsQcow (Object)*
- *BlockdevQcow2Version (Enum)*
- *Qcow2CompressionType (Enum)*
- *BlockdevCreateOptionsQcow2 (Object)*
- *BlockdevCreateOptionsQed (Object)*

- *BlockdevCreateOptionsRbd (Object)*
- *BlockdevVmdkSubformat (Enum)*
- *BlockdevVmdkAdapterType (Enum)*
- *BlockdevCreateOptionsVmdk (Object)*
- *BlockdevCreateOptionsSsh (Object)*
- *BlockdevCreateOptionsVdi (Object)*
- *BlockdevVhdxSubformat (Enum)*
- *BlockdevCreateOptionsVhdx (Object)*
- *BlockdevVpcSubformat (Enum)*
- *BlockdevCreateOptionsVpc (Object)*
- *BlockdevCreateOptions (Object)*
- *blockdev−create (Command)*
- *BlockdevAmendOptionsLUKS (Object)*
- *BlockdevAmendOptionsQcow2 (Object)*
- *BlockdevAmendOptions (Object)*
- *x−blockdev−amend (Command)*
- *BlockErrorAction (Enum)*
- *BLOCK_IMAGE_CORRUPTED (Event)*
- *BLOCK_IO_ERROR (Event)*
- *BLOCK_JOB_COMPLETED (Event)*
- *BLOCK_JOB_CANCELLED (Event)*
- *BLOCK_JOB_ERROR (Event)*
- *BLOCK_JOB_READY (Event)*
- *BLOCK_JOB_PENDING (Event)*
- *PreallocMode (Enum)*
- *BLOCK_WRITE_THRESHOLD (Event)*
- *block−set−write−threshold (Command)*
- *x−blockdev−change (Command)*
- *x−blockdev−set−iothread (Command)*
- *QuorumOpType (Enum)*
- *QUORUM_FAILURE (Event)*
- *QUORUM_REPORT_BAD (Event)*
- *BlockdevSnapshotInternal (Object)*
- *blockdev−snapshot−internal−sync (Command)*
- *blockdev−snapshot−delete−internal−sync (Command)*
- *Block device exports*

- *Character devices*
  - *ChardevInfo (Object)*

- *query−chardev (Command)*
- *ChardevBackendInfo (Object)*
- *query−chardev−backends (Command)*
- *DataFormat (Enum)*
- *ringbuf−write (Command)*
- *ringbuf−read (Command)*
- *ChardevCommon (Object)*
- *ChardevFile (Object)*
- *ChardevHostdev (Object)*
- *ChardevSocket (Object)*
- *ChardevUdp (Object)*
- *ChardevMux (Object)*
- *ChardevStdio (Object)*
- *ChardevSpiceChannel (Object)*
- *ChardevSpicePort (Object)*
- *ChardevVC (Object)*
- *ChardevRingbuf (Object)*
- *ChardevQemuVDAgent (Object)*
- *ChardevBackendKind (Enum)*
- *ChardevFileWrapper (Object)*
- *ChardevHostdevWrapper (Object)*
- *ChardevSocketWrapper (Object)*
- *ChardevUdpWrapper (Object)*
- *ChardevCommonWrapper (Object)*
- *ChardevMuxWrapper (Object)*
- *ChardevStdioWrapper (Object)*
- *ChardevSpiceChannelWrapper (Object)*
- *ChardevSpicePortWrapper (Object)*
- *ChardevQemuVDAgentWrapper (Object)*
- *ChardevVCWrapper (Object)*
- *ChardevRingbufWrapper (Object)*
- *ChardevBackend (Object)*
- *ChardevReturn (Object)*
- *chardev−add (Command)*
- *chardev−change (Command)*
- *chardev−remove (Command)*
- *chardev−send−break (Command)*
- *VSERPORT_CHANGE (Event)*

- *qom−get (Command)*
- *qom−set (Command)*
- *ObjectTypeInfo (Object)*
- *qom−list−types (Command)*
- *qom−list−properties (Command)*
- *CanHostSocketcanProperties (Object)*
- *ColoCompareProperties (Object)*
- *CryptodevBackendProperties (Object)*
- *CryptodevVhostUserProperties (Object)*
- *DBusVMStateProperties (Object)*
- *NetfilterInsert (Enum)*
- *NetfilterProperties (Object)*
- *FilterBufferProperties (Object)*
- *FilterDumpProperties (Object)*
- *FilterMirrorProperties (Object)*
- *FilterRedirectorProperties (Object)*
- *FilterRewriterProperties (Object)*
- *InputBarrierProperties (Object)*
- *InputLinuxProperties (Object)*
- *IothreadProperties (Object)*
- *MemoryBackendProperties (Object)*
- *MemoryBackendFileProperties (Object)*
- *MemoryBackendMemfdProperties (Object)*
- *MemoryBackendEpcProperties (Object)*
- *PrManagerHelperProperties (Object)*
- *QtestProperties (Object)*
- *RemoteObjectProperties (Object)*
- *RngProperties (Object)*
- *RngEgdProperties (Object)*
- *RngRandomProperties (Object)*
- *SevGuestProperties (Object)*
- *ObjectType (Enum)*
- *ObjectOptions (Object)*
- *object−add (Command)*
- *object−del (Command)*

- *Transactions*
  - *Abort (Object)*
  - *ActionCompletionMode (Enum)*

- *TransactionActionKind (Enum)*
- *AbortWrapper (Object)*
- *BlockDirtyBitmapAddWrapper (Object)*
- *BlockDirtyBitmapWrapper (Object)*
- *BlockDirtyBitmapMergeWrapper (Object)*
- *BlockdevBackupWrapper (Object)*
- *BlockdevSnapshotWrapper (Object)*
- *BlockdevSnapshotInternalWrapper (Object)*
- *BlockdevSnapshotSyncWrapper (Object)*
- *DriveBackupWrapper (Object)*
- *TransactionAction (Object)*
- *TransactionProperties (Object)*
- *transaction (Command)*

BLOCK DEVICES

Block core (VM unrelated)

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

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 2^32, except where compatibility with pre−existing Linux dm−crypt volumes is required.

Values
plain 64−bit sector number truncated to 32−bits

plain64
64−bit sector number

essiv 64−bit sector number encrypted with a hash of the encryption key

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

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

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

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
     }
   }
```

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,serve
            "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 ^C) 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 } }
```

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

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

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": {} }
```

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": {} }
```

COPYRIGHT