NAME

Sys::Virt::StorageVol − Represent & manage a libvirt storage volume

DESCRIPTION

The `Sys::Virt::StorageVol` module represents a storage volume managed by libvirt. A storage volume is always associated with a containing storage pool (`Sys::Virt::StoragePool`).

METHODS

my `$name` = `$vol`->**get_name()**

Returns a string with a locally unique name of the storage vol

my `$name` = `$vol`->**get_key()**

Returns a string with a globally unique key for the storage vol

my `$name` = `$vol`->**get_path()**

Returns a string with a locally unique file path of the storage vol

my `$xml` = `$vol`->**get_xml_description()**

Returns an XML document containing a complete description of the storage vol's configuration

`$vol`->delete($flags)

Immediately delete the storage volume freeing its storage resources. The `flags` parameter indicates any special action to be taken when deleting the volume.

`$vol`->resize($newcapacity, `$flags`=0)

Adjust the size of the storage volume. The `$newcapacity` value semantics depend on the `$flags` parameter. If `$flags` specifies `RESIZE_DELTA` then the `$newcapacity` is relative to the current size. If `$flags` specifies `RESIZE_SHRINK` then the `$newcapacity` value is the amount of space to remove

`$vol`->wipe($flags = 0)

Clear the data in the storage volume to avoid future information leak. The `flags` parameter is currently unused and defaults to zero.

`$vol`->wipe_pattern($algorithm, `$flags` = 0)

Clear the data in the storage volume to avoid future information leak. The `$algorithm` parameter specifies the data pattern used to erase data, and should be one of the WIPE ALGORITHM CONSTANTS listed later. The `flags` parameter is currently unused and defaults to zero.

my `$info` = `$vol`->get_info($flags = 0)

Retrieve live information about the storage volume. The returned `$info` hash reference contains three keys. `type` indicates whether the volume is a file or block device. `capacity` provides the maximum logical size of the volume. `allocation` provides the current physical usage of the volume. The allocation may be less than the capacity for sparse, or grow-on-demand volumes. The allocation may also be larger than the capacity, if there is a metadata overhead for the volume format. `$flags` may take one of the values

Sys::Virt::StorageVol::USE_ALLOCATION

Return the current allocation in allocation

Sys::Virt::StorageVol::GET_PHYSICAL

Return the physical size in allocation

`$vol`->download($st, `$offset`, `$length`, `$flags`=0);

Download data from `$vol` using the stream `$st`. If `$offset` and `$length` are non-zero, then restrict data to the specified volume byte range. The `$flags` accept the following values:

Sys::Virt::StorageVol::VOL_DOWNLOAD_SPARSE_STREAM

If this flag is is set in `@flags` effective transmission of holes is enabled. This assumes using the stream `$st` with combination of `sparse_recv_all` or `recv($flags = VIR_STREAM_RECV_STOP_AT_HOLE)` for honouring holes sent by server.

$vol->upload($st, $offset, $length, $flags=0);

Upload data to $vol using the stream $st. If $offset and $length are non-zero, then restrict data to the specified volume byte range. The $flags accept the following values:

Sys::Virt::StorageVol::VOL_UPLOAD_SPARSE_STREAM

If this is set in $flags effective transmission of holes is enabled. This assumes using the stream $st with combination of sparse_send_all or send_hole to preserve source file sparseness.

CONSTANTS

The following sets of constants are useful when dealing with storage volumes

VOLUME TYPES

The following constants are useful for interpreting the type field in the hash returned by the get_info method

Sys::Virt::StorageVol::TYPE_FILE

The volume is a plain file

Sys::Virt::StorageVol::TYPE_BLOCK

The volume is a block device

Sys::Virt::StorageVol::TYPE_DIR

The volume is a directory

Sys::Virt::StorageVol::TYPE_NETWORK

The volume is a network source

Sys::Virt::StorageVol::TYPE_NETDIR

The volume is a network directory

Sys::Virt::StorageVol::TYPE_PLOOP

The volume is a ploop directory

CREATE MODES

The following constants are useful for the flags parameter of the create method

Sys::Virt::StorageVol::CREATE_PREALLOC_METADATA

Preallocate header metadata when creating the volume.

Sys::Virt::StorageVol::CREATE_REFLINK

Perform lightweight reference copy

DELETE MODES

The following constants are useful for the flags parameter of the delete method

Sys::Virt::StorageVol::DELETE_NORMAL

Do a plain delete without any attempt to scrub data.

Sys::Virt::StorageVol::DELETE_ZEROED

Zero out current allocated blocks when deleting the volume

Sys::Virt::StorageVol::DELETE_WITH_SNAPSHOTS

Delete snapshots associated with the volume

WIPE ALGORITHM CONSTANTS

The following constants specify the algorithm for erasing data

Sys::Virt::StorageVol::WIPE_ALG_BSI

9-pass method recommended by the German Center of Security in Information Technologies

Sys::Virt::StorageVol::WIPE_ALG_DOD

4-pass Dod 5220.22-M section, 8-306 procedure

Sys::Virt::StorageVol::WIPE_ALG_GUTMANN
The canonical 35−pass sequence

Sys::Virt::StorageVol::WIPE_ALG_NNSA
4−pass NNSA Policy Letter NAP−14.1−C (XVI−8)

Sys::Virt::StorageVol::WIPE_ALG_PFITZNER7
7−pass random

Sys::Virt::StorageVol::WIPE_ALG_PFITZNER33
33−pass random

Sys::Virt::StorageVol::WIPE_ALG_RANDOM
1−pass random

Sys::Virt::StorageVol::WIPE_ALG_SCHNEIER
7−pass method described by Bruce Schneier in "Applied Cryptography" (1996)

Sys::Virt::StorageVol::WIPE_ALG_ZERO
1−pass, all zeroes

Sys::Virt::StorageVol::WIPE_ALG_TRIM
1−pass, trim all data on the volume by using TRIM or DISCARD

VOLUME RESIZE CONSTANTS

The following constants control how storage volumes can be resized

Sys::Virt::StorageVol::RESIZE_ALLOCATE
Fully allocate the extra space required during resize

Sys::Virt::StorageVol::RESIZE_DELTA
Treat the new capacity as a delta to the current capacity

Sys::Virt::StorageVol::RESIZE_SHRINK
Treat the new capacity as an amount to remove from the capacity

AUTHORS

Daniel P. Berrange <berrange@redhat.com>

COPYRIGHT

LICENSE

SEE ALSO

Sys::Virt, Sys::Virt::Error, `http://libvirt.org`