

## NAME

xm - xVM management user interface

## SYNOPSIS

xm subcommand [options] domain

## DESCRIPTION

The main interface for command and control of both xVM and guest domains is `virsh(1M)`. Users should use `virsh` wherever possible, as it provides a generic and stable interface to controlling virtualized operating systems. Some xVM operations are not yet implemented by `virsh`. In those cases, the legacy utility `xm` can be used for detailed control.

With minor variations, the basic structure of an `xm` command is:

```
xm subcommand [options] domain
```

...where `subcommand` is one of the subcommands listed below, `domain` is the domain name (which is internally translated to a numeric domain id), and `options` are subcommand-specific options. The exceptions to this structure occur where a subcommand acts on all domains, on the entire machine, or directly on the xVM hypervisor. These exceptions are obvious in the descriptions of the subcommands.

All `xm` operations rely upon the xVM control daemon, `xend(1M)`. `xend` must be running before any `xm` commands can run. As described in the man page, `xend` runs under the service management facility (`smf(5)`), which enables the daemon to start when a system is booted.

Most `xm` subcommands require either root privileges or that you assume the Primary Administrator role.

Most `xm` commands act asynchronously, so the fact that an `xm` command returns immediately does not mean that the requested action is complete. Many operations on domains, such as create and shutdown, can take considerable time (30 seconds or more) to complete.

## SUBCOMMANDS

The xm program supports the subcommands listed below. The parameters and options for a given subcommand are described in the description for that subcommand.

`block-attach domain be-dev fe-dev mode [bedomain]`

Create a new virtual block device. This will notify the guest domain of the new virtual block device..

The block-attach subcommand has the following arguments and options:

`domain`

The guest domain name to which the device will be attached.

`be-dev`

The device in the backend domain (domain 0) to be exported. This can be specified as a physical partition (for example, `phy:/dev/md/dsk/mydisk`, a ZFS volume or a normal file ('`file:/export/disk-image`').

`fe-dev`

The form, either a symbolic name or a numeric id, by which a device should be identified to the guest domain. In Linux, an example of a symbolic name is `/dev/hdc`. For Solaris guest domains, a single number should be used. The specified number will correspond to a Solaris disk ID. For example, disk ID 3 will have a slice 0 name of `/dev/dsk/c0d3s0`.

`mode`

The access mode for the device from the guest domain. Supported modes are `w` (read/write) and `r` (read-only).

`bedomain`

The backend domain hosting the device. This defaults to domain 0. Currently, no other ID is supported.

See EXAMPLES for an example of the use of this subcommand.

`block-configure domain back_dev front_dev mode [back_domain]`

Change block device configuration. Used for changing CDs in an HVM (hardware-based virtual machine) domain; in particular, changing the backend device to refer to a different ISO file. See block-attach for parameter descriptions.

block-detach domain dev-id

Destroy a domain's virtual block device. devid must be the device id given to the device by domain 0. You must run `xm block-list` to determine that number.

block-list [-l|--long] domain

List virtual block devices for a domain. The `block-list` subcommand has a single option:

-l, --long

Display output in long format.

console domain

Attach to domain domain's console. If you have set up your domains to have a text-based login console, you receive a normal login screen.

The console supports only paravirtualized domains. The attached console performs similarly to a serial console.

`control-]` exits the virtual console.

create [option] -f=config-file [name=value]...

The `create` subcommand creates a domain, according to the specifications in the mandatory `config-file` argument. `create` optionally accepts a set of name-value pairs that can override or add to the variables defined in `config-file`.

`config-file` can be an absolute pathname.

The `create` subcommand returns immediately upon domain startup. However, the starting of a domain is independent of the booting of the guest operating system in that domain and independent of that OS's availability for input.

The `create` and `new` subcommands are legacy features. These subcommands are used for existing domains that use the old configuration file format. New domains should be created with `virt-install(1M)`.

The `create` subcommand has the following options:

-c  
--console\_autoconnect

Attach to the console of the domain as soon as it

has started.

`-f=file, --defconfig=file`

Use the given Python configuration script, `file`. The configuration script is loaded after arguments have been processed. Each command-line option sets a configuration variable named after its long option name, and these variables are placed in the environment of the script before it is loaded. Variables for options that can be repeated have list values. Other variables can be set using `var=value` on the command line. After the script is loaded, option values that were not set on the command line are replaced by the values set in the script.

`-F=file, --config=file`

Use the given SXP-format configuration file. This is an internal format; this option is useful only for debugging purposes.

`-h, --help`

Display list of options for create subcommand.

`--help_config`

Display the available configuration variables (vars) from the configuration script.

`-n, --dryrun`

Dry run. Displays the resulting configuration in SXP but does not create the domain.

`-p, --paused`

Leave the domain paused after it is created.

`-q, --quiet`

Display no messages over the course of domain creation.

`delete domain`

Removes the domain `domain` from xVM domain management. This is the same as the `virsh(1M) undefine`, which should be used in place of this subcommand.

## destroy domain

Immediately terminate the domain domain. For the domain OS, this is the equivalent of abruptly removing the power from a physical machine. In most cases, you will want to use the shutdown command instead.

## dmesg [-c]

Displays recent messages in the xVM message buffer; analogous to dmesg(1M). The message buffer contains informational, warning, and error messages created during xVM's operation.

The dmesg subcommand supports the following option:

-c, --clear

Clears xVM's message buffer.

## domid domain

Converts a domain name to a domain ID.

Domain IDs change on each boot, whereas names are permanent. See xVM(5) for an explanation of the differences among a domain ID, UUID, and name.

## domname domain

Converts a domain ID to a domain name.

## dump-core domain [output-file]

Dumps core for the domain domain. By default, the domain continues to run after a dump is collected. If output-file is not specified, the domain core dump is generated in /var/xen/dump/. Core dump files can be large. Solaris guest domain cores can be debugged using mdb(1).

The dump-core domain has the following options:

-C, --crash

Crash domain after dumping core.

-L, --live

Dump core without pausing the domain.

help [-l, --long]

Displays a list of common xm subcommands. xm help supports the following option:

-l, --long      Display a complete list of xm subcommands, grouped by function.

info

Display information about the xVM host in name : value format. The information reported by info is useful for inclusion in a bug report.

list [-l, --long] [domain, ...]

Displays information about one or more domains. If no domains are specified, displays information about all domains.

An example of list output:

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	3456	2	r-----	244.7
solaris	1	511	30	-b----	353.8

The fields in this output are as follows:

Name

Domain name

ID

Numeric domain ID.

Mem

Amount of memory, in MB, currently allocated to a domain.

VCPUS

Number of virtual CPUs assigned to a domain.

State

Run state (described below).

## Time

Total run time of the domain as accounted for by xVM.

The State field in `xm list` output can, for a given domain, display one of the following letters.

r

Running. The domain is currently running on a CPU.

b

Blocked. The domain is not currently running. It is either idle or waiting on I/O.

p

Paused. The domain has been paused, occurring usually as a result of an administrator running `xm pause`. When in a paused state the domain still consumes allocated resources, such as memory, but will not be eligible for scheduling by the xVM hypervisor. See also the `virsh suspend` subcommand.

s

Shutdown. The domain is in the state it was in prior to startup. This state will, most likely, never be visible.

c

Crashed. The domain has crashed, which means that it terminated in an abrupt, unexpected manner. Usually this state can occur only if the domain has been configured not to restart on crash.

d

Dying. The domain is in process of moving to a shutdown or crashed state.

The `list` subcommand supports the following option:

`-l, --long`

Displays more detailed information about each domain than is shown in the standard list output table.

Display the `xend(1M)` log. The log file is `/var/log/xen/xend.log`.

#### `mem-max` domain mem

Specify the maximum amount of memory a domain is able to use. mem is specified in megabytes.

The `mem-max` value might not correspond to the actual memory used in a domain, because a domain might scale down its memory usage to return memory to the OS.

#### `mem-set` domain mem

Set the amount of memory used by the running domain. Because this operation requires cooperation from the domain operating system, there is no guarantee that it will succeed.

Warning: There is no good way to know in advance how small of a `mem-set` value will make a domain unstable and cause it to crash. Be very careful when using this command on running domains. Solaris guest domains attempt to refuse potentially dangerous settings.

#### `migrate` [options] domain host

Migrate a domain to another host machine. On the target host machine, the following conditions must obtain for this subcommand to be successful:

- o The other host must be running the same version of xVM.
- o The migration TCP port must be open and accepting connections from the source host.
- o There must be sufficient resources-memory, disk, and so forth-for the domain to run.

See `xend(1M)` for an explanation of how to set up a machine to receive a remote migration.

The domain's accessible disks must reside on some form of shared storage, such as NFS files or iSCSI volumes, and this storage must be accessible to both hosts

The `migrate` subcommand supports the following option:

`-l, --live`

Use live migration. This option migrates the domain between hosts without shutting down the domain.

#### `network-attach` domain [script=scriptname] [ip=ipaddr]

[mac=macaddr] [bridge=link] [backend=bedomain]

Creates a new network device in the domain specified by domain. The subcommand has the following arguments:

domain

Domain in which the network device is to be created.

script=scriptname

Use the specified script name to bring up the network.

ip=ipaddr

Passes the specified IP address to the adapter upon creation. This address might be ignored by the specified domain.

mac=macaddr

The MAC address that the domain will see on its Ethernet device. If the MAC address is not specified, it will be randomly generated with the 00:16:3e vendor id prefix.

bridge=link

The name of the network link to which to attach a virtual interface, in case you have more than one.

backend=bedomain

The backend domain id. By default, this is domain 0. Note that backend != 0 is not currently operational.

network-detach domain dev-id

Removes the network device from the domain specified bydomain. dev-id is the virtual interface device number within the domain.

network-list [-l|--long] domain

List virtual network interfaces for a domain.

-l, --long

Display output in long format.

## new domain

The new subcommand creates (but does not start) the domain defined by the given configuration file.

The new and create subcommands are legacy features. These subcommands are used for existing domains that use the old configuration file format. New domains should use virt-install(1M).

`-f=file, --defconfig=file`

Use the given Python configuration script, file. The configuration script is loaded after arguments have been processed. Each command-line option sets a configuration variable named after its long option name, and these variables are placed in the environment of the script before it is loaded. Variables for options that can be repeated have list values. Other variables can be set using `var=value` on the command line. After the script is loaded, option values that were not set on the command line are replaced by the values set in the script.

`-F=file, --config=file`

Use the given SXP-format configuration file. This is an internal format; this option is useful only for debugging purposes.

`--help_config`

Display the available configuration variables (vars) from the configuration script.

`-n, --dryrun`

Dry run. Displays the resulting configuration in SXP but does not create the domain.

## pause domain

Pause a domain. When in a paused state the domain still consumes allocated resources, such as memory, but will not be eligible for scheduling by the xVM hypervisor.

## reboot [options] domain

Reboot a domain. The effect of this subcommand is the same as if the domain had the `init 6` command (see `init(1M)`) run from the console. Unless `-w` is specified,

reboot returns as soon as it has initiated the reboot process, which can be significantly before the domain actually reboots.

The reboot subcommand supports the following options:

-a, --all

Reboot all domains.

-w, --wait

Wait for reboot to complete before returning. This might take an extended period, as all services in the domain will have to be shutdown cleanly.

rename oldname newname

Renames the domain oldname to newname.

restore state-file

Build a domain from an xm save state file. See the save subcommand.

resume domain

Resume the activities of the domain domain, which is in a suspended state as a result of the suspend subcommand.

save domain state-file

Saves a running domain to a file state-file, so that it can later be restored, using the restore subcommand. Once saved, the domain will no longer be running on the system, thus the memory allocated for the domain will be free for the use of other domains.

Note that network connections present before the save operation might be severed, as TCP timeouts might have expired.

sched-credit -d domain [-w weight|-ccap]

Get and set credit scheduler parameters for the specified domain. See xVM(5) for a description of the credit scheduler. Without the -w or -c options, the current settings for the given domain are shown. Otherwise, the relevant parameter is set.

The parameters to the sched-credit subcommand are as follows:

-c cap, --cap=cap

Set the maximum amount of CPU a domain can consume. A value of zero (the default) means no maximum is set. This value is expressed in percentage points of a physical CPU. For example, a value of 50 specifies a cap of half a physical CPU.

-d domain, --domain=domain

Domain for which to set scheduling parameters.

-w weight, --weight=weight

Set the relative weight of the domain. A domain with twice the weight will receive twice the CPU time as another domain, if CPU use is in contention. Valid weights are in the range 1-65536 and the default is 256.

sched-sedf domain period slice latency-hint extratime weight

Set Simple EDF scheduler parameters. This scheduler provides weighted CPU sharing in an intuitive way and uses realtime algorithms to ensure time guarantees. The Simple EDF scheduler is not the default scheduler used in xVM.

The parameters to the sched-sedf subcommand are as follows:

domain

The domain for which scheduling parameters applies.

period

The normal EDF scheduling usage, in nanoseconds.

slice

The normal EDF scheduling usage, in nanoseconds.

latency-hint

Scaled period if domain is performing heavy I/O.

extratime

Flag for allowing domain to run in extra time.

weight

Another way of setting CPU slice.

shell

Launches an interactive shell.

shutdown [options] domain

Gracefully shuts down a domain. The effect of this subcommand is the same as if the domain had the `init 5` command (see `init(1M)`) run from the console. This subcommand coordinates with the domain OS to perform graceful shutdown. The duration of the entire shutdown will vary, depending on what services must be shutdown in the domain. The shutdown subcommand returns immediately after signalling the domain, unless the `-w` option is used.

The shutdown subcommand supports the following options:

`-a`

Shutdown all domains.

`-w`

Wait for the domain to complete shutdown before returning.

start domain

Start the domain specified by domain.

suspend domain

Suspend the activities of all services in the domain specified by domain.

sysrq domain letter

For the accepted signals in a Linux domain, refer to the Linux documentation. For Solaris signalling the letter `b` causes the domain to enter `kldb(1)`, the Solaris kernel debugger, if that debugger is loaded. Any other letter has no effect.

top domain...

Invokes the `xentop(1M)` command. Monitor a host and one or more domains in real time.

unpause domain

Moves the domain `domain` out of the paused state. This will allow a previously paused domain to now be eligible for scheduling by the xVM hypervisor. See the `pause` subcommand.

uptime domain

Provides information on resource usage for domain `domain`. Analogous to the `uptime(1)` command.

vcpu-list domain

Lists VCPU information for the domain `domain`. If no domain is specified, the subcommand provides VCPU information for all domains.

vcpu-pin domain vcpu cpus

Pins the VCPU to only run on the specified CPUs. The keyword `all` can be used to apply the `cpus` list to all VCPUs in the domain.

Normally VCPUs can float between available CPUs whenever xVM deems a different run state is appropriate. Pinning can be used to restrict this, by ensuring certain VCPUs can run only on certain physical CPUs.

vcpu-set domain vcpu-count

Enables the number `vcpu-count` of virtual CPUs for the domain in question. Like the `mem-sets` subcommand, `vcpu-set` can allocate only up to the maximum virtual CPU count configured at boot time for a domain.

If `vcpu-count` is smaller than the current number of active VCPUs, the highest numbered VCPUs will be hotplug removed. This might have consequences for pinned VCPUs.

Attempting to set the VCPUs to a number larger than the initially configured VCPU count is an error. Trying to set VCPUs to less than one will be silently ignored.

## EXAMPLES

Example 1 Attach a File as a Read-only Block Device

The following example attaches a file as a read-only block device to a Solaris guest domain, as /dev/dsk/c0d3.

```
xm block-attach solaris1 file:/data/disk.img 3 r
```

Example 2 Live Migration of a domU to a Different Host

```
xm migrate --live solaris1 solaris-host2
```

Example 3 Pin a Domain's vcpus to Corresponding CPUs

```
xm vcpu-pin solaris1 0 5  
xm vcpu-pin solaris1 1 6
```

Example 4 Balloon Down a Domain to Use Less Memory

```
xm mem-set solaris1 512
```

#### AUTHORS

- o Sean Dague, sean at dague dot net
- o Daniel Stekloff, dsteklof at us dot ibm dot com
- o Reiner Sailer, sailer at us dot ibm dot com

#### ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxvmu
Interface Stability	Volatile

#### SEE ALSO

kmdb(1), uptime(1), dmesg(1M), init(1M), virsh(1M), virt-install(1M), xend(1M), xentop(1M), xenstored(1M), attributes(5), smf(5), xVM(5)