

## Invocation

```
qemu-system-x86_64 [options] [disk_image]
```

disk\_image is a raw hard disk image for IDE hard disk 0. Some targets do not need a disk image.

When dealing with options parameters as arbitrary strings containing commas, such as in “file=my,file” and “string=a,b”, it’s necessary to double the commas. For instance, “-fw\_cfg name=z,string=a,,b” will be parsed as “-fw\_cfg name=z,string=a,b”.

## Standard options

## -h

Display help and exit

```
-version
```

Display version information and exit

```
-machine [type=]name[,prop=value[,...]]
```

Select the emulated machine by name. Use `-machine help` to list available machines.

For architectures which aim to support live migration compatibility across releases, each release will introduce a new versioned machine type. For example, the 2.8.0 release introduced machine types “pc-i440fx-2.8” and “pc-q35-2.8” for the x86\_64/i686 architectures.

To allow live migration of guests from QEMU version 2.8.0, to QEMU version 2.9.0, the 2.9.0 version must support the “pc-i440fx-2.8” and “pc-q35-2.8” machines too. To allow users live migrating VMs to skip multiple intermediate releases when upgrading, new releases of QEMU will support machine types from many previous versions.

Supported machine properties are:

```
accel=accels1[:accels2[...]]
```

This is used to enable an accelerator. Depending on the target architecture, `kvm`, `xen`, `hvf`, `nvmm`, `whpx` or `tcg` can be available. By default, `tcg` is used. If there is more than one accelerator specified, the next one is used if the previous one fails to initialize.

**vmport=on|off|auto**

Enables emulation of VMWare IO port, for vmmouse etc. auto says to select the value based on accel and i8042. For accel=xen or i8042=off the default is off otherwise the default is on.

```
dump-guest-core=on|off
```

Include guest memory in a core dump. The default is on.

mem-merge=on|off

Enables or disables memory merge support. This feature, when supported by the host, de-duplicates identical memory pages among VMs instances (enabled by default).

`aes-key-wrap=on | off`

Enables or disables AES key wrapping support on s390-ccw hosts. This feature controls whether AES wrapping keys will be created to allow execution of AES cryptographic functions. The default is on.

```
dea-key-wrap=on | off
```

Enables or disables DEA key wrapping support on s390-ccw hosts. This feature controls whether DEA wrapping keys will be created to allow execution of DEA cryptographic functions. The default is on.

**nvdimm=on | off**

Enables or disables NVDIMM support. The default is off.

```
memory-encryption=
```

Memory encryption object to use. The default is none.

## hmat=on|off

Enables or disables ACPI Heterogeneous Memory Attribute Table (HMAT) support. The default is off.

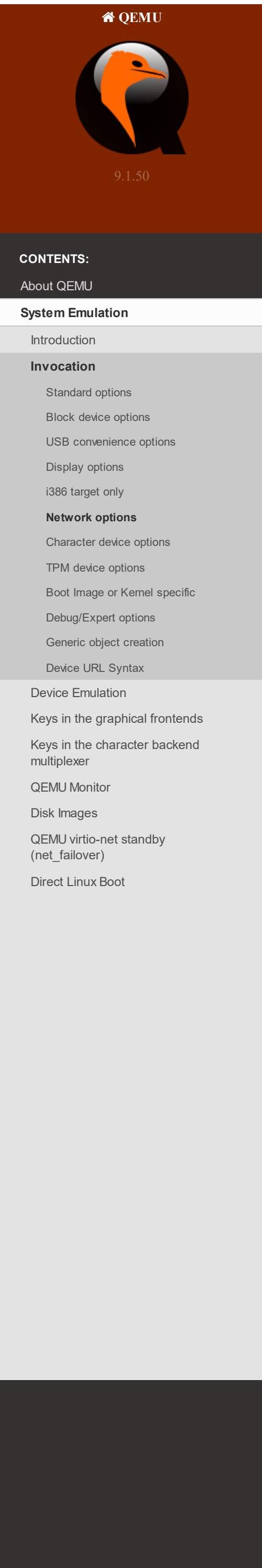
```
memory-backend='id'
```

An alternative to `legacy -mem-path` and `mem-prealloc` options. Allows to use a memory backend as main RAM.

For example:

```
-object memory-backend-file,id=pc.ram,size=512M,mem-path=/hugetlbfs,prealloc=on,share=on
-machine memory-backend=pc.ram
-m 512M
```





4096 could be a good initial value if you have no idea which is the best. Set this value to 0 to disable the feature. By default, this feature is disabled (dirty-ring-size=0). When enabled, KVM will instead record dirty pages in a bitmap.

eager-split-size=n

KVM implements dirty page logging at the PAGE\_SIZE granularity and enabling dirty-logging on a huge-page requires breaking it into PAGE\_SIZE pages in the first place. KVM on ARM does this splitting lazily by default. There are performance benefits in doing huge-page split eagerly, especially in situations where TLBI costs associated with break-before-make sequences are considerable and also if guest workloads are read intensive. The size here specifies how many pages to break at a time and needs to be a valid block size which is 1GB/2MB/4KB, 32MB/16KB and 512MB/64KB for 4KB/16KB/64KB PAGE\_SIZE respectively. Be wary of specifying a higher size as it will have an impact on the memory. By default, this feature is disabled (eager-split-size=0).

```
notify-vmexit=run|internal-error|disable,notify-window=n
```

Enables or disables notify VM exit support on x86 host and specify the corresponding notify window to trigger the VM exit if enabled. `run` option enables the feature. It does nothing and continue if the exit happens. `internal-error` option enables the feature. It raises a internal error. `disable` option doesn't enable the feature. This feature can mitigate the CPU stuck issue due to event windows don't open up for a specified of time (i.e. `notify-window`). Default: `notify-vmexit=run,notify-window=0`.

```
device=path
```

Sets the path to the KVM device node. Defaults to `/dev/kvm`. This option can be used to pass the KVM device to use via a file descriptor by setting the value to `/dev/fdset/NN`.

```
-smp [[cpus=n][,maxcpus=maxcpus][,drawers=drawers][,books=books][,sockets=sockets][,dies=dies][,clusters=clusters][,modules=modules][,cores=cores][,threads=threads]
```

Simulate a SMP system with ‘n’ CPUs initially present on the machine type board. On boards supporting CPU hotplug, the optional ‘maxcpus’ parameter can be set to enable further CPUs to be added at runtime. When both parameters are omitted, the maximum number of CPUs will be calculated from the provided topology members and the initial CPU count will match the maximum number. When only one of them is given then the omitted one will be set to its counterpart’s value. Both parameters may be specified, but the maximum number of CPUs must be equal to or greater than the initial CPU count. Product of the CPU topology hierarchy must be equal to the maximum number of CPUs. Both parameters are subject to an upper limit that is determined by the specific machine type chosen.

To control reporting of CPU topology information, values of the topology parameters can be specified. Machines may only support a subset of the parameters and different machines may have different subsets supported which vary depending on capacity of the corresponding CPU targets. So for a particular machine type board, an expected topology hierarchy can be defined through the supported sub-option. Unsupported parameters can also be provided in addition to the sub-option, but their values must be set as 1 in the purpose of correct parsing.

Either the initial CPU count, or at least one of the topology parameters must be specified. The specified parameters must be greater than zero, explicit configuration like “cpus=0” is not allowed. Values for any omitted parameters will be computed from those which are given.

For example, the following sub-option defines a CPU topology hierarchy (2 sockets totally on the machine, 2 cores per socket, 2 threads per core) for a machine that only supports sockets/cores/threads. Some members of the option can be omitted but their values will be automatically computed:

```
-smp 8,sockets=2,cores=2,threads=2,maxcpus=8
```

The following sub-option defines a CPU topology hierarchy (2 sockets totally on the machine, 2 dies per socket, 2 modules per die, 2 cores per module, 2 threads per core) for PC machines which support sockets/dies /modules/cores/threads. Some members of the option can be omitted but their values will be automatically computed:

```
-smp 32,sockets=2,dies=2,modules=2,cores=2,threads=2,maxcpus=32
```

The following sub-option defines a CPU topology hierarchy (2 sockets totally on the machine, 2 clusters per socket, 2 cores per cluster, 2 threads per core) for ARM virt machines which support sockets/clusters /cores/threads. Some members of the option can be omitted but their values will be automatically computed:

```
-smp 16,sockets=2,clusters=2,cores=2,threads=2,maxcpus=16
```

Historically preference was given to the coarsest topology parameters when computing missing values (ie sockets preferred over cores, which were preferred over threads), however, this behaviour is considered liable to change. Prior to 6.2 the preference was sockets over cores over threads. Since 6.2 the preference is cores over sockets over threads.

For example, the following option defines a machine board with 2 sockets of 1 core before 6.2 and 1 socket of 2 cores after 6.2:

- smp 2

Note: The cluster topology will only be generated in ACPI and exposed to guest if it's explicitly specified in -smp.

```
-numa node[,mem=size][,cpus=firstcpu[-lastcpu]][,nodeid=node][,initiator=initiator]
```

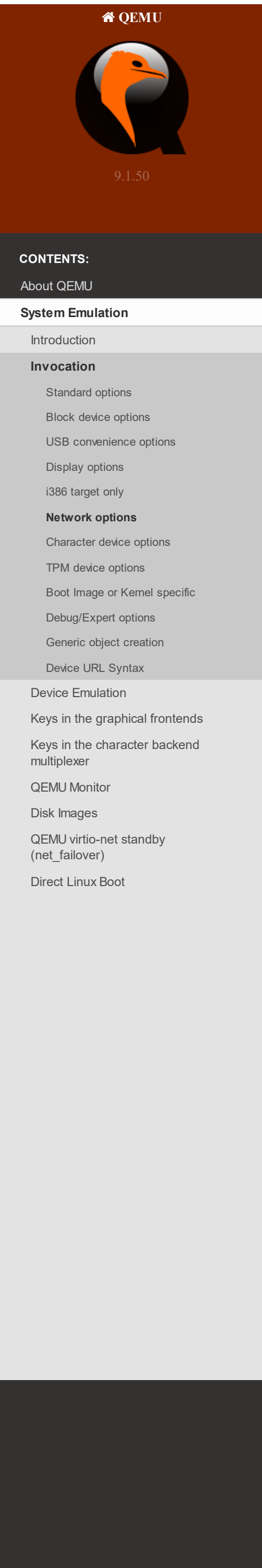
```
-numa node[,memdev=id][,cpus=firstcpu[-lastcpu]][,nodeid=node][,initiator=initiator]
```

```
- numa dist,src=source,dst=destination,val=distance
```

```
- numa cpu,node-id=node[,socket-id=x][,core-id=y][,thread-id=z]
```

```
-numa hmat-lb,initiator=node,target=node,hierarchy=hierarchy,data-type=type[,latency=lat][,bandwidth=bw]
```

```
-numa hmat-cache,node-id=node,size=size,level=level[,associativity=str][,policy=str][,line=size]
```



Define a NUMA node and assign RAM and VCPUs to it. Set the NUMA distance from a source node to a destination node. Set the ACPI Heterogeneous Memory Attributes for the given nodes.

Legacy VCPU assignment uses ‘cpus’ option where firstcpu and lastcpu are CPU indexes. Each ‘cpus’ option represent a contiguous range of CPU indexes (or a single VCPU if lastcpu is omitted). A non-contiguous set of VCPUs can be represented by providing multiple ‘cpus’ options. If ‘cpus’ is omitted on all nodes, VCPUs are automatically split between them.

For example, the following option assigns VCPUs 0, 1, 2 and 5 to a NUMA node:

```
- numa node, cpus=0-2, cpus=5
```

‘cpu’ option is a new alternative to ‘cpus’ option which uses ‘socket-id|core-id|thread-id’ properties to assign CPU objects to a node using topology layout properties of CPU. The set of properties is machine specific, and depends on used machine type/‘smp’ options. It could be queried with ‘hotpluggable-cpus’ monitor command. ‘node-id’ property specifies node to which CPU object will be assigned, it’s required for node to be declared with ‘node’ option before it’s used with ‘cpu’ option.

For example:

```
-M pc \
-smp 1,sockets=2,maxcpus=2 \
-numa node,nodeid=0 -numa node,nodeid=1 \
-numa cpu,node-id=0,socket-id=0 -numa cpu,node-id=1,socket-id=1
```

‘memdev’ option assigns RAM from a given memory backend device to a node. It is recommended to use ‘memdev’ option over legacy ‘mem’ option. This is because ‘memdev’ option provides better performance and more control over the backend’s RAM (e.g. ‘prealloc’ parameter of ‘-memory-backend-ram’ allows memory preallocation).

For compatibility reasons, legacy ‘mem’ option is supported in 5.0 and older machine types. Note that ‘mem’ and ‘memdev’ are mutually exclusive. If one node uses ‘memdev’, the rest nodes have to use ‘memdev’ option, and vice versa.

Users must specify memory for all NUMA nodes by 'memdev' (or legacy 'mem' if available). In QEMU 5.2, the support for '-numa node' without memory specified was removed.

‘initiator’ is an additional option that points to an initiator NUMA node that has best performance (the lowest latency or largest bandwidth) to this NUMA node. Note that this option can be set only when the machine property ‘hmat’ is set to ‘on’.

Following example creates a machine with 2 NUMA nodes, node 0 has CPU. node 1 has only memory, and its initiator is node 0. Note that because node 0 has CPU, by default the initiator of node 0 is itself and must be itself.

```
-machine hmat=on \
-m 2G,slots=2,maxmem=4G \
-object memory-backend-ram,size=1G,id=m0 \
-object memory-backend-ram,size=1G,id=m1 \
-numa node,nodeid=0,memdev=m0 \
-numa node,nodeid=1,memdev=m1,initiator=0 \
-smp 2,sockets=2,maxcpus=2 \
-numa cpu,node-id=0,socket-id=0 \
-numa cpu,node-id=0,socket-id=1
```

source and destination are NUMA node IDs. distance is the NUMA distance from source to destination. The distance from a node to itself is always 10. If any pair of nodes is given a distance, then all pairs must be given distances. Although, when distances are only given in one direction for each pair of nodes, then the distances in the opposite directions are assumed to be the same. If, however, an asymmetrical pair of distances is given for even one node pair, then all node pairs must be provided distance values for both directions, even when they are symmetrical. When a node is unreachable from another node, set the pair's distance to 255.

Note that the `-numa` option doesn't allocate any of the specified resources, it just assigns existing resources to NUMA nodes. This means that one still has to use the `-m`, `-smp` options to allocate RAM and VCPUs respectively.

Use 'hmat-1b' to set System Locality Latency and Bandwidth Information between initiator and target NUMA nodes in ACPI Heterogeneous Attribute Memory Table (HMAT). Initiator NUMA node can create memory requests, usually it has one or more processors. Target NUMA node contains addressable memory.

In ‘hmat-1b’ option, node are NUMA node IDs. hierarchy is the memory hierarchy of the target NUMA node: if hierarchy is ‘memory’, the structure represents the memory performance; if hierarchy is ‘first-level|second-level|third-level’, this structure represents aggregated performance of memory side caches for each domain. type of ‘data-type’ is type of data represented by this structure instance: if ‘hierarchy’ is ‘memory’, ‘data-type’ is ‘access|read|write’ latency or ‘access|read|write’ bandwidth of the target memory; if ‘hierarchy’ is ‘first-level|second-level|third-level’, ‘data-type’ is ‘access|read|write’ hit latency or ‘access|read|write’ hit bandwidth of the target memory side cache.

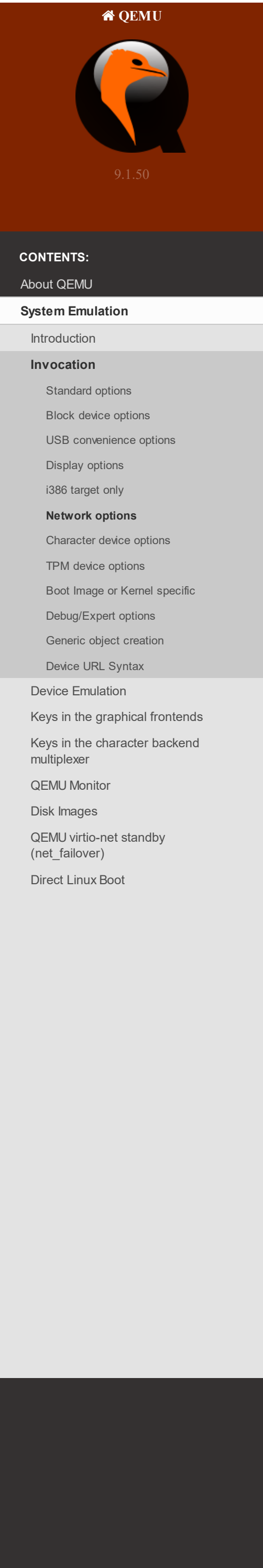
lat is latency value in nanoseconds. bw is bandwidth value, the possible value and units are NUM[M|G|T], mean that the bandwidth value are NUM byte per second (or MB/s, GB/s or TB/s depending on used suffix). Note that if latency or bandwidth value is 0, means the corresponding latency or bandwidth information is not provided.

In 'hmat-cache' option, node-id is the NUMA-id of the memory belongs. size is the size of memory side cache in bytes. level is the cache level described in this structure, note that the cache level 0 should not be used with 'hmat-cache' option. associativity is the cache associativity, the possible value is 'none/direct(direct-mapped)/complex(complex cache indexing)'. policy is the write policy. line is the cache Line size in bytes.

For example, the following options describe 2 NUMA nodes. Node 0 has 2 cpus and a ram, node 1 has only a ram. The processors in node 0 access memory in node 0 with access-latency 5 nanoseconds, access-bandwidth is 200 MB/s; The processors in NUMA node 0 access memory in NUMA node 1 with access-latency 10 nanoseconds, access-bandwidth is 100 MB/s. And for memory side cache information, NUMA node 0 and 1 both have 1 level memory cache, size is 10KB, policy is write-back, the cache Line size is 8 bytes:







Sets guest startup RAM size to megs megabytes. Default is 128 MiB. Optionally, a suffix of “M” or “G” can be used to signify a value in megabytes or gigabytes respectively. Optional pair slots, maxmem could be used to set amount of hotpluggable memory slots and maximum amount of memory. Note that maxmem must be aligned to the page size.

For example, the following command-line sets the guest startup RAM size to 1GB, creates 3 slots to hotplug additional memory and sets the maximum memory the guest can reach to 4GB:

```
qemu-system-x86_64 -m 1G,slots=3,maxmem=4G
```

If slots and maxmem are not specified, memory hotplug won't be enabled and the guest startup RAM will never increase.

**-mem-path path**

Allocate guest RAM from a temporarily created file in path.

```
-mem-prealloc
```

Preallocate memory when using -mem-path.

-k language

Use keyboard layout language (for example `fr` for French). This option is only needed where it is not easy to get raw PC keycodes (e.g. on Macs, with some X11 servers or with a VNC or curses display). You don't normally need to use it on PC/Linux or PC/Windows hosts.

The available layouts are:

ar	de-ch	es	fo	fr-ca	hu	ja	mk	no	pt-br	sv
da	en-gb	et	fr	fr-ch	is	lt	nl	pl	ru	th
de	en-us	fi	fr-be	hr	it	lv	nl-be	pt	sl	tr

The default is en-us.

```
-audio [driver=]driver[,model=value][,prop[=value][,...]]
```

If the `model` option is specified, `-audio` is a shortcut for configuring both the guest audio hardware and the host audio backend in one go. The guest hardware model can be set with `model=modelname`. Use `model=help` to list the available device types.

The following two example do exactly the same, to show how `-audio` can be used to shorten the command line length:

```
qemu-system-x86_64 -audiodev pa,id=pa -device sb16,audiodev=pa
qemu-system-x86_64 -audio pa,model=sb16
```

If the `model` option is not specified, `-audio` is used to configure a default audio backend that will be used whenever the `audiodev` property is not set on a device or machine. In particular, `-audio none` ensures that no audio is produced even for machines that have embedded sound hardware.

In both cases, the driver option is the same as with the corresponding `-audiodev` option below. Use `driver=help` to list the available drivers.

```
-audiodev [driver=]driver,id=id[,prop[=value][,...]]
```

Adds a new audio backend driver identified by `id`. There are global and driver specific properties. Some values can be set differently for input and output, they're marked with `in|out..` You can set the input's property with `in.prop` and the output's property with `out.prop`. For example:

```
-audiodev alsa,id=example,in.frequency=44100,out.frequency=8000
-audiodev alsa,id=example,out.channels=1 # leaves in.channels unspecified
```

NOTE: parameter validation is known to be incomplete, in many cases specifying an invalid option causes QEMU to print an error message and continue emulation without sound.

Valid global options are:

```
id=identifier
```

Identifies the audio backend.

```
timer-period=period
```

Sets the timer period used by the audio subsystem in microseconds. Default is 10000 (10 ms).

```
in|out.mixing-engine=on|off
```

Use QEMU's mixing engine to mix all streams inside QEMU and convert audio formats when not supported by the backend. When off, fixed-settings must be off too. Note that disabling this option means that the selected backend must support multiple streams and the audio formats used by the virtual cards, otherwise you'll get no sound. It's not recommended to disable this option unless you want to use 5.1 or 7.1 audio, as mixing engine only supports mono and stereo audio. Default is on.

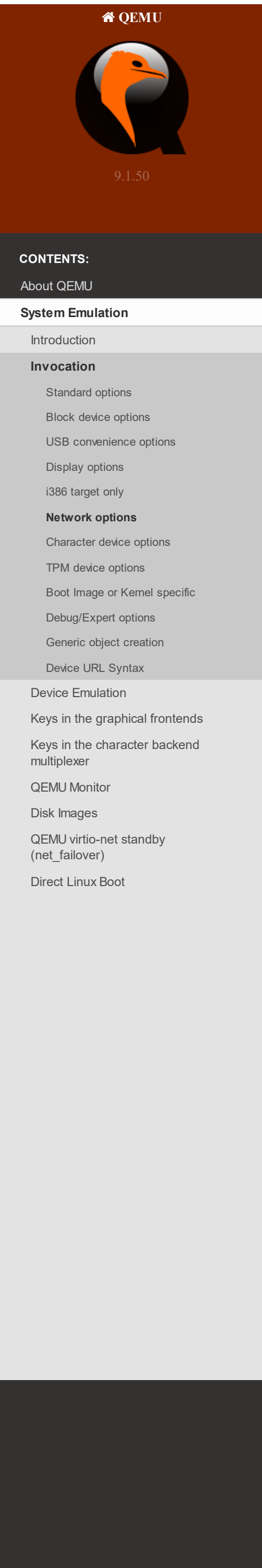
```
in|out.fixed-settings=on|off
```

Use fixed settings for host audio. When off, it will change based on how the guest opens the sound card. In this case you must not specify frequency, channels or format. Default is on.

```
in|out.frequency=frequency
```

Specify the frequency to use when using fixed-settings. Default is 44100Hz.





PulseAudio specific options are:

```
server=server
```

Sets the PulseAudio server to connect to.

```
in|out.name=sink
```

Use the specified source/sink for recording/playback.

```
in|out.latency=usecs
```

Desired latency in microseconds. The PulseAudio server will try to honor this value but actual latencies may be lower or higher.

```
-audiodev pipewire,id=id[,prop[=value][,...]]
```

Creates a backend using PipeWire. This backend is available on most systems.

PipeWire specific options are:

```
in|out.latency=usecs
```

Desired latency in microseconds.

```
in|out.name=sink
```

Use the specified source/sink for recording/playback.

```
in|out.stream-name
```

Specify the name of pipewire stream.

```
-audiodev sdl,id=id[,prop[=value][,...]]
```

Creates a backend using SDL. This backend is available on most systems, but you should use your platform's native backend if possible.

SDL specific options are:

```
in|out.buffer-count=count
```

Sets the count of the buffers.

```
-audiodev sndio,id=id[,prop[=value][,...]]
```

Creates a backend using SNDIO. This backend is available on OpenBSD and most other Unix-like systems.

Sndio specific options are:

```
in|out.dev=device
```

Specify the sndio device to use for input and/or output. Default is default.

```
in|out.latency=usecs
```

Sets the desired period length in microseconds.

```
-audiodev spice,id=id[,prop[=value][,...]]
```

Creates a backend that sends audio through SPICE. This backend requires `-spice` and automatically selected in that case, so usually you can ignore this option. This backend has no backend specific properties.

```
-audiodev wav,id=id[,prop[=value][,...]]
```

Creates a backend that writes audio to a WAV file.

Backend specific options are:

```
path=path
```

Write recorded audio into the specified file. Default is `qemu.wav`.

```
-device driver[,prop[=value][,...]]
```

Add device driver. `prop=value` sets driver properties. Valid properties depend on the driver. To get help on possible drivers and properties, use `-device help` and `-device driver,help`.

Some drivers are:

```
-device ipmi-bmc-sim,id=id[,prop[=value][,...]]
```

Add an IPMI BMC. This is a simulation of a hardware management interface processor that normally sits on a system. It provides a watchdog and the ability to reset and power control the system. You need to connect this to an IPMI interface to make it useful

The IPMI slave address to use for the BMC. The default is 0x20. This address is the BMC's address on the I2C network of management controllers. If you don't know what this means, it is safe to ignore it.

```
id=id
```

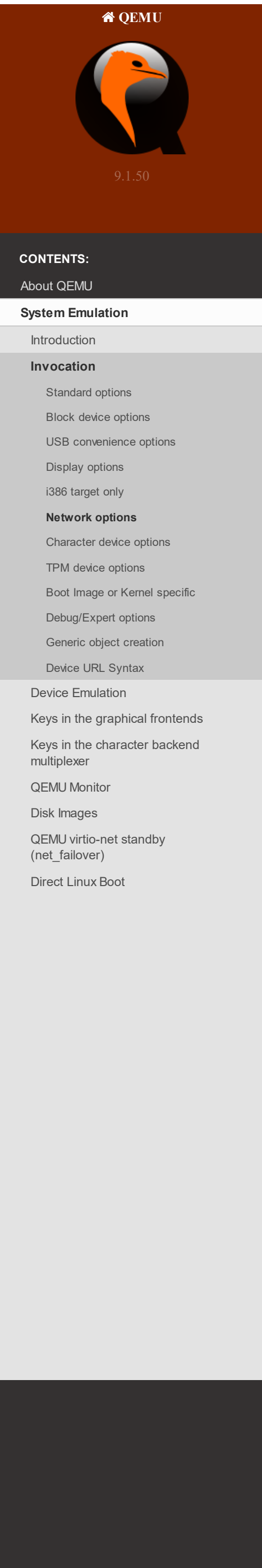
The BMC id for interfaces to use this device.

```
slave_addr=val
```

Define slave address to use for the BMC. The default is 0x20.

```
sdrfile=file
```





file containing raw Sensor Data Records (SDR) data. The default is none.

```
fruareasize=val
```

size of a Field Replaceable Unit (FRU) area. The default is 1024.

```
frudatafile=file
```

file containing raw Field Replaceable Unit (FRU) inventory data. The default is none.

```
guid=uuid
```

value for the GUID for the BMC, in standard UUID format. If this is set, get “Get GUID” command to the BMC will return it. Otherwise “Get GUID” will return an error.

```
-device ipmi-bmc-extern,id=id,chardev=id[,slave_addr=val]
```

Add a connection to an external IPMI BMC simulator. Instead of locally emulating the BMC like the above item, instead connect to an external entity that provides the IPMI services.

A connection is made to an external BMC simulator. If you do this, it is strongly recommended that you use the “reconnect-ms=” chardev option to reconnect to the simulator if the connection is lost. Note that if this is not used carefully, it can be a security issue, as the interface has the ability to send resets, NMIs, and power off the VM. It’s best if QEMU makes a connection to an external simulator running on a secure port on localhost, so neither the simulator nor QEMU is exposed to any outside network.

See the “lanserv/README.vm” file in the OpenIPMI library for more details on the external interface.

```
-device isa-ipmi-kcs,bmc=id[,ioport=val][,irq=val]
```

Add a KCS IPMI interface on the ISA bus. This also adds a corresponding ACPI and SMBIOS entries, if appropriate.

```
bmc=id
```

The BMC to connect to, one of ipmi-bmc-sim or ipmi-bmc-extern above.

```
ioport=val
```

Define the I/O address of the interface. The default is 0xca0 for KCS.

```
irq=val
```

Define the interrupt to use. The default is 5. To disable interrupts, set this to 0.

```
-device isa-ipmi-bt,bmc=id[,ioport=val][,irq=val]
```

Like the KCS interface, but defines a BT interface. The default port is 0xe4 and the default interrupt is 5.

```
-device pci-ipmi-kcs,bmc=id
```

Add a KCS IPMI interface on the PCI bus.

**bmc=id**

The BMC to connect to, one of ipmi-bmc-sim or ipmi-bmc-extern above.

```
-device pci-ipmi-bt,bmc=id
```

Like the KCS interface, but defines a BT interface on the PCI bus.

```
-device intel-iommu[,option=...]
```

This is only supported by `-machine q35`, which will enable Intel VT-d emulation within the guest. It supports below options:

**intremap=on|off (default: auto)**

This enables interrupt remapping feature. It's required to enable complete x2apic. Currently it only supports kvm kernel-irqchip modes `off` or `split`, while full kernel-irqchip is not yet supported. The default value is "auto", which will be decided by the mode of kernel-irqchip.

**caching-mode=on | off (default: off)**

This enables caching mode for the VT-d emulated device. When caching-mode is enabled, each guest DMA buffer mapping will generate an IOTLB invalidation from the guest IOMMU driver to the vIOMMU device in a synchronous way. It is required for -device vfio-pci to work with the VT-d device, because host assigned devices requires to setup the DMA mapping on the host before guest DMA starts.

**device-iotlb=on|off (default: off)**

This enables device-iotlb capability for the emulated VT-d device. So far virtio/vhost should be the only real user for this parameter, paired with ats=on configured for the device.

**aw-bits=39 | 48 (default: 39)**

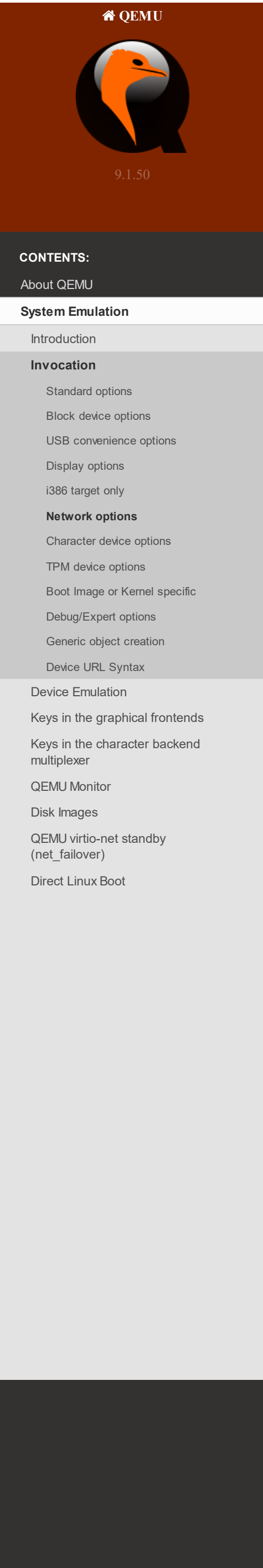
This decides the address width of IOVA address space. The address space has 39 bits width for 3-level IOMMU page tables, and 48 bits for 4-level IOMMU page tables.

Please also refer to the wiki page for general scenarios of VT-d emulation in QEMU: <https://wiki.qemu.org/Features/VT-d>.

```
-device virtio-iommu-pci[,option=...]
```

This is only supported by `-machine q35 (x86_64)` and `-machine virt (ARM)`. It supports below options:

**granule=val (possible values are 4k, 8k, 16k, 64k and host; default: host)**



This decides the default granule to be exposed by the virtio-iommu. If host, the granule matches the host page size.

**aw-bits=val (val between 32 and 64, default depends on machine)**

This decides the address width of the IOVA address space.

**-name name**

Sets the name of the guest. This name will be displayed in the SDL window caption. The name will also be used for the VNC server. Also optionally set the top visible process name in Linux. Naming of individual threads can also be enabled on Linux to aid debugging.

```
-uuid uuid
```

Set system UUID.

## Block device options

The QEMU block device handling options have a long history and have gone through several iterations as the feature set and complexity of the block layer have grown. Many online guides to QEMU often reference older and deprecated options, which can lead to confusion.

The most explicit way to describe disks is to use a combination of `-device` to specify the hardware device and `-blockdev` to describe the backend. The device defines what the guest sees and the backend describes how QEMU handles the data. It is the only guaranteed stable interface for describing block devices and as such is recommended for management tools and scripting.

The `-drive` option combines the device and backend into a single command line option which is a more human friendly. There is however no interface stability guarantee although some older board models still need updating to work with the modern blockdev forms.

Older options like `-hda` are essentially macros which expand into `-drive` options for various drive interfaces. The original forms bake in a lot of assumptions from the days when QEMU was emulating a legacy PC, they are not recommended for modern configurations.

```
-fda file
```

```
-fdb file
```

Use file as floppy disk 0/1 image (see the [Disk Images](#) chapter in the System Emulation Users Guide).

### -hda file

### -hdb file

## -hdc file

### -hdd file

Use file as hard disk 0, 1, 2 or 3 image on the default bus of the emulated machine (this is for example the IDE bus on most x86 machines, but it can also be SCSI, virtio or something else on other target architectures). See also the [Disk Images](#) chapter in the System Emulation Users Guide.

```
-cdrom file
```

Use file as CD-ROM image on the default bus of the emulated machine (which is IDE1 master on x86, so you cannot use `-hdc` and `-cdrom` at the same time there). On systems that support it, you can use the host CD-ROM by using `/dev/cdrom` as filename.

```
-blockdev option[,option[,option[,...]]]
```

Define a new block driver node. Some of the options apply to all block drivers, other options are only accepted for a specific block driver. See below for a list of generic options and options for the most common block drivers.

Options that expect a reference to another node (e.g. `file`) can be given in two ways. Either you specify the node name of an already existing node (`file=node-name`), or you define a new node inline, adding options for the referenced node after a dot (`file.filename=path,file.aio=native`).

A block driver node created with `-blockdev` can be used for a guest device by specifying its node name for the `drive` property in a `-device` argument that defines a block device.

Valid options for any block driver node:

```
driver
```

Specifies the block driver to use for the given node.

node-name

This defines the name of the block driver node by which it will be referenced later. The name must be unique, i.e. it must not match the name of a different block driver node, or (if you use `-drive` as well) the ID of a drive.

If no node name is specified, it is automatically generated. The generated node name is not intended to be predictable and changes between QEMU invocations. For the top level, an explicit node name must be specified.

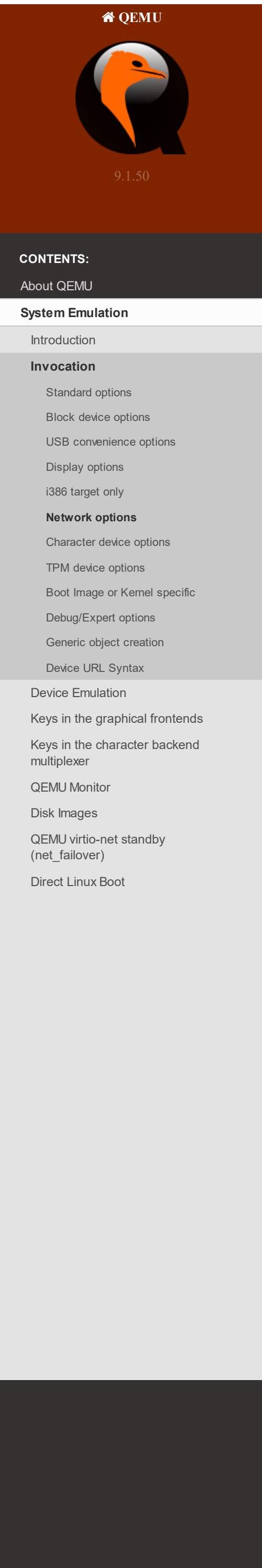
read-only

Open the node read-only. Guest write attempts will fail.

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

**auto-read-only**

If `auto-read-only=on` is set, QEMU may fall back to read-only usage even when `read-only=off` is requested, or even switch between modes as needed, e.g. depending on whether the image file is writable or whether a writing user is attached to the node.



**force-share**

Override the image locking system of QEMU by forcing the node to utilize weaker shared access for permissions where it would normally request exclusive access. When there is the potential for multiple instances to have the same file open (whether this invocation of QEMU is the first or the second instance), both instances must permit shared access for the second instance to succeed at opening the file.

Enabling force-share=on requires read-only=on.

cache.direct

The host page cache can be avoided with `cache.direct=on`. This will attempt to do disk IO directly to the guest's memory. QEMU may still perform an internal copy of the data.

cache.no-flush

In case you don't care about data integrity over host failures, you can use `cache.no-flush=on`. This option tells QEMU that it never needs to write any data to the disk but can instead keep things in cache. If anything goes wrong, like your host losing power, the disk storage getting disconnected accidentally, etc. your image will most probably be rendered unusable.

discard=discard

discard is one of “ignore” (or “off”) or “unmap” (or “on”) and controls whether discard (also known as trim or unmap) requests are ignored or passed to the filesystem. Some machine types may not support discard requests.

detect-zeroes=detect-zeroes

detect-zeroes is “off”, “on” or “unmap” and enables the automatic conversion of plain zero writes by the OS to driver specific optimized zero write commands. You may even choose “unmap” if discard is set to “unmap” to allow a zero write to be converted to an unmap operation.

## Driver-specific options for file

This is the protocol-level block driver for accessing regular files.

**filename**

The path to the image file in the local filesystem

## aio

Specifies the AIO backend (threads/native/io\_uring, default: threads)

locking

Specifies whether the image file is protected with Linux OFD / POSIX locks. The default is to use the Linux Open File Descriptor API if available, otherwise no lock is applied. (auto/on/off, default: auto)

Example:

```
-blockdev driver=file,node-name=disk,filename=disk.img
```

## Driver-specific options for raw

This is the image format block driver for raw images. It is usually stacked on top of a protocol level block driver such as `file`.

## file

Reference to or definition of the data source block driver node (e.g. a file driver node)

Example 1:

```
-blockdev driver=file,node-name=disk_file,filename=disk.img
-blockdev driver=raw,node-name=disk,file=disk_file
```

Example 2:

```
-blockdev driver=raw,node-name=disk,file.driver=file,file.filename=disk.img
```

## Driver-specific options for qcow2

This is the image format block driver for qcow2 images. It is usually stacked on top of a protocol level block driver such as `file`.

## file

Reference to or definition of the data source block driver node (e.g. a file driver node)

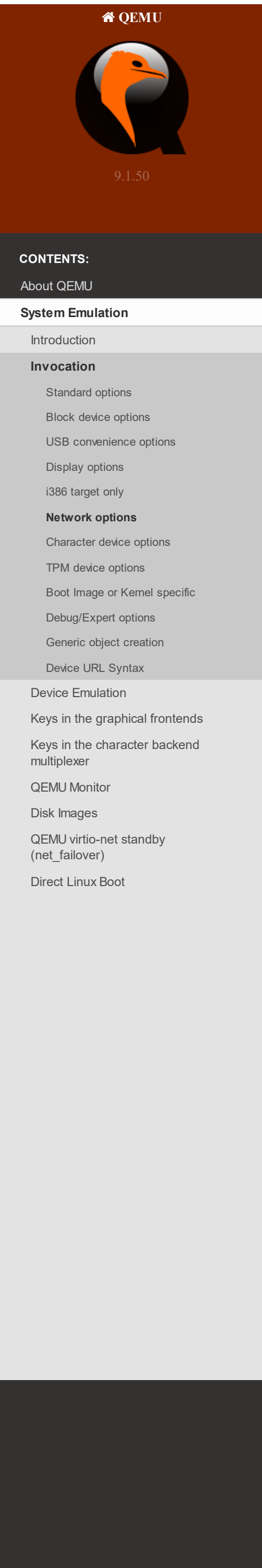
## backing

Reference to or definition of the backing file block device (default is taken from the image file). It is allowed to pass `null` here in order to disable the default backing file.

## lazy-refcounts

Whether to enable the lazy refcounts feature (on/off; default is taken from the image file)

cache-size



The maximum total size of the L2 table and recount block caches in bytes (default: the sum of l2-cache-size and recount-cache-size)

## 12-cache-size

The maximum size of the L2 table cache in bytes (default: if cache-size is not specified - 32M on Linux platforms, and 8M on non-Linux platforms; otherwise, as large as possible within the cache-size, while permitting the requested or the minimal refcount cache size)

```
refcount-cache-size
```

The maximum size of the refcount block cache in bytes (default: 4 times the cluster size; or if cache-size is specified, the part of it which is not used for the L2 cache)

**cache-clean-interval**

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. Setting it to 0 disables this feature.

**pass-discard-request**

Whether discard requests to the qcow2 device should be forwarded to the data source (on/off; default: on if discard=unmap is specified, off otherwise)

**pass-discard-snapshot**

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 (on/off; default: on)

pass-discard-other

Whether discard requests for the data source should be issued on other occasions where a cluster gets freed (on/off; default: off)

**discard-no-unref**

When enabled, data clusters will remain preallocated when they are no longer used, e.g. because they are discarded or converted to zero clusters. As usual, whether the old data is discarded or kept on the protocol level (i.e. in the image file) depends on the setting of the pass-discard-request option. Keeping the clusters preallocated prevents qcow2 fragmentation that would otherwise be caused by freeing and re-allocating them later. Besides potential performance degradation, such fragmentation can lead to increased allocation of clusters past the end of the image file, resulting in image files whose file length can grow much larger than their guest disk size would suggest. If image file length is of concern (e.g. when storing qcow2 images directly on block devices), you should consider enabling this option.

## overlap-check

Which overlap checks to perform for writes to the image (none/constant/cached/all; default: cached). For details or finer granularity control refer to the QAPI documentation of `blockdev-add`.

Example 1:

```
-blockdev driver=file,node-name=my_file,filename=/tmp/disk.qcow2
-blockdev driver=qcow2,node-name=hda,file=my_file,overlap-check=none,cache-size=16777216
```

Example 2:

```
-blockdev driver=qcow2,node-name=disk,file.driver=http,file.filename=http://example.com/image.qcow2
```

## Driver-specific options for other drivers

Please refer to the QAPI documentation of the `blockdev-add` QMP command.

```
-drive option[,option[,option[,...]]]
```

Define a new drive. This includes creating a block driver node (the backend) as well as a guest device, and is mostly a shortcut for defining the corresponding `-blockdev` and `-device` options.

-drive accepts all options that are accepted by -blockdev. In addition, it knows the following options:

```
file=file
```

This option defines which disk image (see the [Disk Images](#) chapter in the System Emulation Users Guide) to use with this drive. If the filename contains comma, you must double it (for instance, “file=my,,file” to use file “my,file”).

Special files such as iSCSI devices can be specified using protocol specific URLs. See the section for “Device URL Syntax” for more information.

```
if=interface
```

This option defines on which type on interface the drive is connected. Available types are: ide, scsi, sd, mtd, floppy, pflash, virtio, none.

```
bus=bus,unit=unit
```

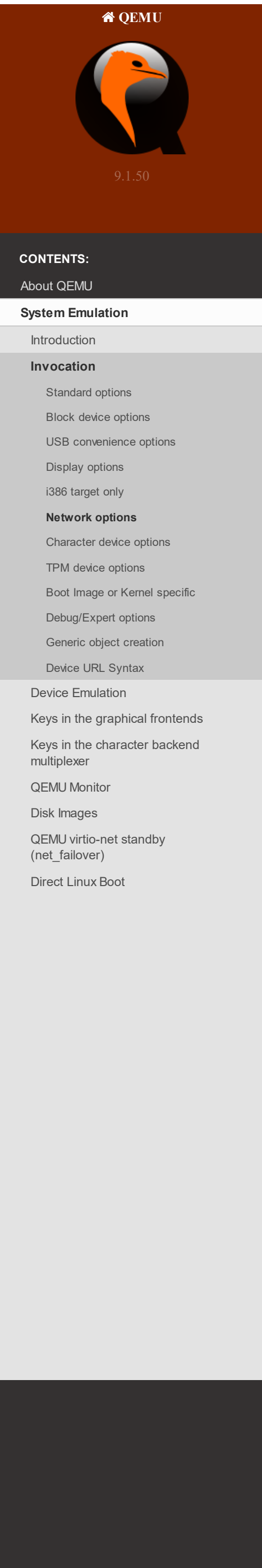
These options define where is connected the drive by defining the bus number and the unit id.

```
index=index
```

This option defines where the drive is connected by using an index in the list of available connectors of a given interface type.

```
media=media
```





This option defines the type of the media: disk or cdrom.

```
snapshot=snapshot
```

snapshot is “on” or “off” and controls snapshot mode for the given drive (see -snapshot).

**cache=cache**

cache is “none”, “writeback”, “unsafe”, “directsync” or “writethrough” and controls how the host cache is used to access block data. This is a shortcut that sets the `cache.direct` and `cache.no-flush` options (as in `-blockdev`), and additionally `cache.writeback`, which provides a default for the `write-cache` option of block guest devices (as in `-device`). The modes correspond to the following settings:

	cache.writeback	cache.direct	cache.no-flush
writeback	on	off	off
none	on	on	off
writethrough	off	off	off
directsync	off	on	off
unsafe	on	off	on

The default mode is `cache=writeback`.

```
aio=aio
```

aio is “threads”, “native”, or “io\_uring” and selects between pthread based disk I/O, native Linux AIO, or Linux io\_uring API.

```
format=format
```

Specify which disk format will be used rather than detecting the format. Can be used to specify format=raw to avoid interpreting an untrusted format header.

```
werror=action,rerror=action
```

Specify which action to take on write and read errors. Valid actions are: “ignore” (ignore the error and try to continue), “stop” (pause QEMU), “report” (report the error to the guest), “enospc” (pause QEMU only if the host disk is full; report the error to the guest otherwise). The default setting is `werror=enospc` and `rerror=report`.

copy-on-read=copy-on-read

copy-on-read is “on” or “off” and enables whether to copy read backing file sectors into the image file.

```
bps=b, bps_rd=r, bps_wr=w
```

Specify bandwidth throttling limits in bytes per second, either for all request types or for reads or writes only. Small values can lead to timeouts or hangs inside the guest. A safe minimum for disks is 2 MB/s.

```
bps_max=bm, bps_rd_max=rm, bps_wr_max=wm
```

Specify bursts in bytes per second, either for all request types or for reads or writes only. Bursts allow the guest I/O to spike above the limit temporarily.

```
iops=i,iops_rd=r,iops_wr=w
```

Specify request rate limits in requests per second, either for all request types or for reads or writes only.

```
iops_max=bm,iops_rd_max=rm,iops_wr_max=wm
```

Specify bursts in requests per second, either for all request types or for reads or writes only. Bursts allow the guest I/O to spike above the limit temporarily.

```
iops_size=is
```

Let every is bytes of a request count as a new request for iops throttling purposes. Use this option to prevent guests from circumventing iops limits by sending fewer but larger requests.

group=g

Join a throttling quota group with given name `g`. All drives that are members of the same group are accounted for together. Use this option to prevent guests from circumventing throttling limits by using many small disks instead of a single larger disk.

By default, the `cache.writeback=on` mode is used. It will report data writes as completed as soon as the data is present in the host page cache. This is safe as long as your guest OS makes sure to correctly flush disk caches where needed. If your guest OS does not handle volatile disk write caches correctly and your host crashes or loses power, then the guest may experience data corruption.

For such guests, you should consider using `cache.writeback=off`. This means that the host page cache will be used to read and write data, but write notification will be sent to the guest only after QEMU has made sure to flush each write to the disk. Be aware that this has a major impact on performance.

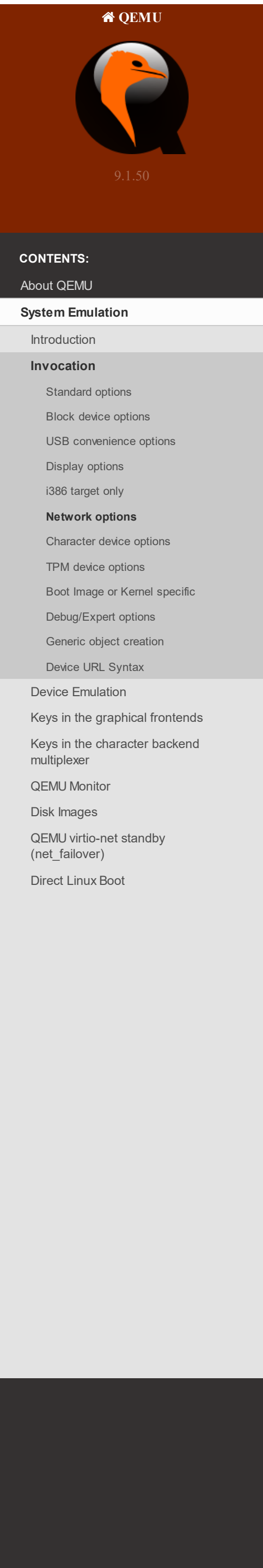
When using the `-snapshot` option, unsafe caching is always used.

Copy-on-read avoids accessing the same backing file sectors repeatedly and is useful when the backing file is over a slow network. By default copy-on-read is off.

Instead of `-cdrom` you can use:

```
qemu-system-x86_64 -drive file=file,index=2,media=cdrom
```





this security model cannot interact with other unix tools. “none” security model is same as passthrough except the sever won’t report failures if it fails to set file attributes like ownership. Security model is mandatory only for local fsdriver.

```
writeout=writeout
```

This is an optional argument. The only supported value is “immediate”. This means that host page cache will be used to read and write data but write notification will be sent to the guest only when the data has been reported as written by the storage subsystem.

**readonly=on**

Enables exporting 9p share as a readonly mount for guests. By default read-write access is given.

## fmode=fmode

Specifies the default mode for newly created files on the host. Works only with security models “mapped-xattr” and “mapped-file”.

```
dmode=dmode
```

Specifies the default mode for newly created directories on the host. Works only with security models “mapped-xattr” and “mapped-file”.

throttling.bps-total=b,throttling.bps-read=r,throttling.bps-write=w

Specify bandwidth throttling limits in bytes per second, either for all request types or for reads or writes only.

```
throttling.bps-total-max=bm, bps-read-max=rm, bps-write-max=wm
```

Specify bursts in bytes per second, either for all request types or for reads or writes only. Bursts allow the guest I/O to spike above the limit temporarily.

```
throttling.iops-total=i,throttling.iops-read=r, throttling.iops-write=w
```

Specify request rate limits in requests per second, either for all request types or for reads or writes only.

```
throttling.iops-total-max=im,throttling.iops-read-max=irm, throttling.iops-write-max=iwm
```

Specify bursts in requests per second, either for all request types or for reads or writes only. Bursts allow the guest I/O to spike above the limit temporarily.

throttling.iops-size=1s

Let every `is` bytes of a request count as a new request for `iops` throttling purposes.

-fsdev option is used along with -device driver “virtio-9p-...”.

```
-device virtio-9p-type,fsdev=id,mount_tag=mount_tag
```

Options for virtio-9p-... driver are:

## type

Specifies the variant to be used. Supported values are “pci”, “ccw” or “device”, depending on the machine type.

```
fsdev=id
```

Specifies the id value specified along with -fsdev option.

```
mount_tag=mount_tag
```

Specifies the tag name to be used by the guest to mount this export point.

```
-virtfs local,path=path,mount_tag=mount_tag ,security_model=security_model[,writeout=writeout]
[,readonly=on] [,fmode=fmode][,dmode=dmode][,multidevs=multidevs]
```

```
-virtfs synth,mount_tag=mount_tag
```

Define a new virtual filesystem device and expose it to the guest using a virtio-9p-device (a.k.a. 9pfs), which essentially means that a certain directory on host is made directly accessible by guest as a pass-through file system by using the 9P network protocol for communication between host and guests, if desired even accessible, shared by several guests simultaneously.

Note that `-virtfs` is actually just a convenience shortcut for its generalized form `-fsdev -device virtio-9p-pci`.

The general form of pass-through file system options are:

local

Accesses to the filesystem are done by QEMU.

synth

Synthetic filesystem, only used by QTests.

```
id=id
```

Specifies identifier for the filesystem device

```
path=path
```

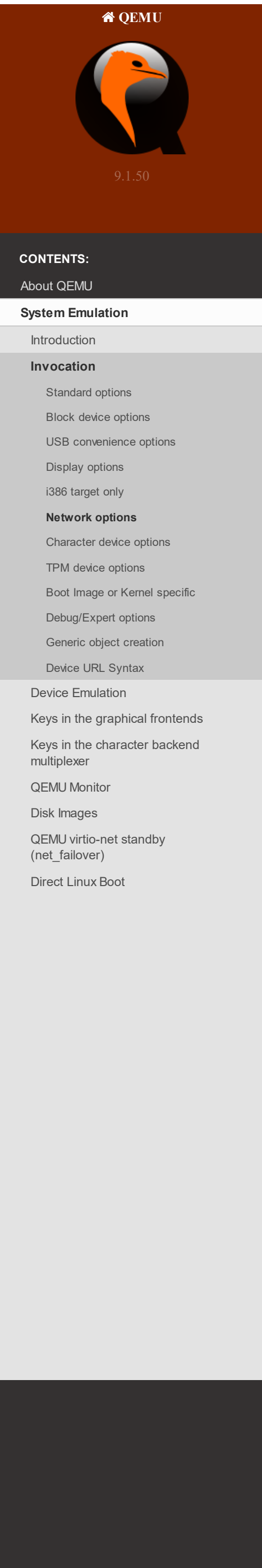
Specifies the export path for the file system device. Files under this path will be available to the 9p client on the guest.

```
security_model=security_model
```

Specifies the security model to be used for this export path. Supported security models are “passthrough”, “mapped-xattr”, “mapped-file” and “none”. In “passthrough” security model, files are stored using the same credentials as they are created on the guest. This







```
spice-app[,gl=on|off]
```

Start QEMU as a Spice server and launch the default Spice client application. The Spice server will redirect the serial consoles and QEMU monitors. (Since 4.0)

dbus

Export the display over D-Bus interfaces. (Since 7.0)

The connection is registered with the “org.gemu” name (and queued when already owned).

addr=<dbusaddr> : D-Bus bus address to connect to.

p2p=yes | no : Use peer-to-peer connection, accepted via QMP add\_client.

`gl=on|off|core|es` : Use OpenGL for rendering (the D-Bus interface will share framebuffers with DMABUF file descriptors).

## sdl

Display video output via SDL (usually in a separate graphics window; see the SDL documentation for other possibilities). Valid parameters are:

`grab-mod=<mods>` : Used to select the modifier keys for toggling the mouse grabbing in conjunction with the “g” key. <mods> can be either `lshift-lctrl-lalt` or `rctrl`.

gl=on|off|core|es : Use OpenGL for displaying

show-cursor=on|off : Force showing the mouse cursor

window-close=on|off : Allow to quit qemu with window close button

gtk

Display video output in a GTK window. This interface provides drop-down menus and other UI elements to configure and control the VM during runtime. Valid parameters are:

full-screen=on|off : Start in fullscreen mode

gl=on|off : Use OpenGL for displaying

grab-on-hover=on|off : Grab keyboard input on mouse hover

**show-tabs=on|off :** Display the tab bar for switching between the

various graphical interfaces (e.g. VGA and virtual console character devices) by default.

show-cursor=on|off : Force showing the mouse cursor

window-close=on|off : Allow to quit qemu with window close button

`show-menubar=on|off` : Display the main window menubar, defaults to “on”

**zoom-to-fit=on|off** : Expand video output to the window size,

defaults to “off”

```
curses[,charset=<encoding>]
```

Display video output via curses. For graphics device models which support a text mode, QEMU can display this output using a curses/ncurses interface. Nothing is displayed when the graphics device is in graphical mode or if the graphics device does not support a text mode. Generally only the VGA device models support text mode. The font charset used by the guest can be specified with the `charset` option, for example `charset=CP850` for IBM CP850 encoding. The default is CP437.

**cocoa**

Display video output in a Cocoa window. Mac only. This interface provides drop-down menus and other UI elements to configure and control the VM during runtime. Valid parameters are:

**full-grab=on|off** : Capture all key presses, including system combos.

This requires accessibility permissions, since it performs a global grab on key events. (default: off) See <https://support.apple.com/en-in/guide/mac-help/mh32356/mac>

**swap-opt-cmd=on|off** : Swap the Option and Command keys so that their

key codes match their position on non-Mac keyboards and you can use Meta/Super and Alt where you expect them. (default: off)

show-cursor=on|off : Force showing the mouse cursor

left-command-key=on|off : Disable forwarding left command key to host


full-screen=on|off : Start in fullscreen mode

**zoom-to-fit=on|off** : Expand video output to the window size,

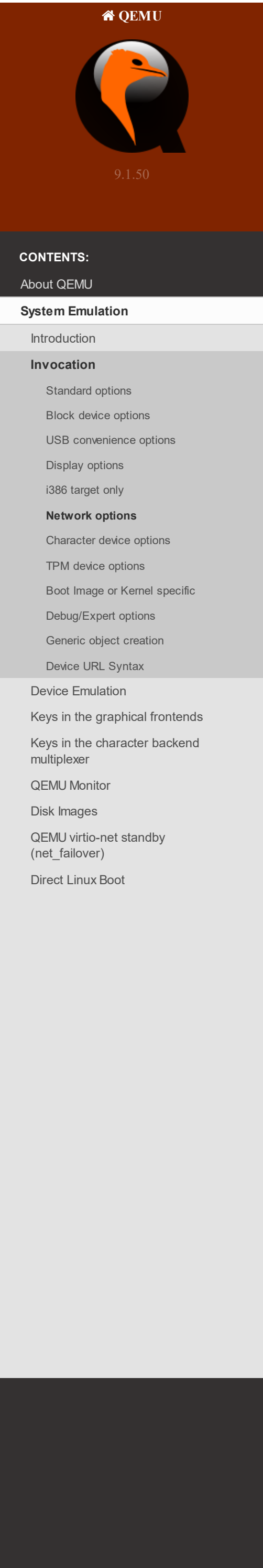
defaults to “off”

```
egl-headless[,rendernode=<file>]
```

Offload all OpenGL operations to a local DRI device. For any graphical display, this display needs to be paired with either VNC or SPICE displays.

<div><div>9.1.50</div></div>	
CONTENTS:	
About QEMU	
System Emulation	
Introduction	
Invocation	
Standard options	
Block device options	
USB convenience options	
Display options	
i386 target only	
Network options	
Character device options	
TPM device options	
Boot Image or Kernel specific	
Debug/Expert options	
Generic object creation	
Device URL Syntax	
Device Emulation	
Keys in the graphical frontends	
Keys in the character backend multiplexer	
QEMU Monitor	
Disk Images	
QEMU virtio-net standby (net_failover)	
Direct Linux Boot	

<b>vnc=&lt;display&gt;</b>
Start a VNC server on display <display>
<b>none</b>
Do not display video output. The guest will still see an emulated graphics card, but its output will not be displayed to the QEMU user. This option differs from the -nographic option in that it only affects what is done with video output; -nographic also changes the destination of the serial and parallel port data.
<b>-nographic</b>
Normally, if QEMU is compiled with graphical window support, it displays output such as guest graphics, guest console, and the QEMU monitor in a window. With this option, you can totally disable graphical output so that QEMU is a simple command line application. The emulated serial port is redirected on the console and muxed with the monitor (unless redirected elsewhere explicitly). Therefore, you can still use QEMU to debug a Linux kernel with a serial console. Use C-a h for help on switching between the console and monitor.
<b>-spice option[,option[,...]]</b>
Enable the spice remote desktop protocol. Valid options are
<b>port=&lt;nr&gt;</b>
Set the TCP port spice is listening on for plaintext channels.
<b>addr=&lt;addr&gt;</b>
Set the IP address spice is listening on. Default is any address.
<b>ipv4=on off; ipv6=on off; unix=on off</b>
Force using the specified IP version.
<b>password-secret=&lt;secret-id&gt;</b>
Set the ID of the secret object containing the password you need to authenticate.
<b>sasl=on off</b>
Require that the client use SASL to authenticate with the spice. The exact choice of authentication method used is controlled from the system/ user’s SASL configuration file for the ‘qemu’ service. This is typically found in /etc/sasl2/qemu.conf. If running QEMU as an unprivileged user, an environment variable SASL_CONF_PATH can be used to make it search alternate locations for the service config. While some SASL auth methods can also provide data encryption (eg GSSAPI), it is recommended that SASL always be combined with the ‘tls’ and ‘x509’ settings to enable use of SSL and server certificates. This ensures a data encryption preventing compromise of authentication credentials.
<b>disable-ticketing=on off</b>
Allow client connects without authentication.
<b>disable-copy-paste=on off</b>
Disable copy paste between the client and the guest.
<b>disable-agent-file-xfer=on off</b>
Disable spice-vdagent based file-xfer between the client and the guest.
<b>tls-port=&lt;nr&gt;</b>
Set the TCP port spice is listening on for encrypted channels.
<b>x509-dir=&lt;dir&gt;</b>
Set the x509 file directory. Expects same filenames as -vnc \$display,x509=\$dir
<b>x509-key-file=&lt;file&gt;; x509-key-password=&lt;file&gt;; x509-cert-file=&lt;file&gt;; x509-cacert-file=&lt;file&gt;; x509-dh-key-file=&lt;file&gt;</b>
The x509 file names can also be configured individually.
<b>tls-ciphers=&lt;list&gt;</b>
Specify which ciphers to use.
<b>tls-channel=[main display cursor inputs record playback]; plaintext-channel=[main display cursor inputs record playback]</b>
Force specific channel to be used with or without TLS encryption. The options can be specified multiple times to configure multiple channels. The special name “default” can be used to set the default mode. For channels which are not explicitly forced into one mode the spice client is allowed to pick tls/plaintext as he pleases.
<b>image-compression=[auto_glz auto_lz quic glz lz off]</b>
Configure image compression (lossless). Default is auto_glz.
<b>jpeg-wan-compression=[auto never always]; zlib-glz-wan-compression=[auto never always]</b>
Configure wan image compression (lossy for slow links). Default is auto.
<b>streaming-video=[off all filter]</b>
Configure video stream detection. Default is off.



```
agent-mouse=[on|off]
```

Enable/disable passing mouse events via vdagent. Default is on.

```
playback-compression=[on|off]
```

Enable/disable audio stream compression (using celt 0.5.1). Default is on.

```
seamless-migration=[on|off]
```

Enable/disable spice seamless migration. Default is off.

```
g1=[on | off]
```

Enable/disable OpenGL context. Default is off.

```
rendernode=<file>
```

DRM render node for OpenGL rendering. If not specified, it will pick the first available. (Since 2.9)

```
-vga type
```

Select type of VGA card to emulate. Valid values for type are

**cirrus**

Cirrus Logic GD5446 Video card. All Windows versions starting from Windows 95 should recognize and use this graphic card. For optimal performances, use 16 bit color depth in the guest and the host OS. (This card was the default before QEMU 2.2)

## std

Standard VGA card with Bochs VBE extensions. If your guest OS supports the VESA 2.0 VBE extensions (e.g. Windows XP) and if you want to use high resolution modes ( $\geq 1280 \times 1024 \times 16$ ) then you should use this option. (This card is the default since QEMU 2.2)

## vmware

VMWare SVGA-II compatible adapter. Use it if you have sufficiently recent XFree86/XOrg server or Windows guest with a driver for this card.

qx1

QXL paravirtual graphic card. It is VGA compatible (including VESA 2.0 VBE support). Works best with qxl guest drivers installed though. Recommended choice when using the spice protocol.

## tcx

(sun4m only) Sun TCX framebuffer. This is the default framebuffer for sun4m machines and offers both 8-bit and 24-bit colour depths at a fixed resolution of 1024x768.

## cg3

(sun4m only) Sun cgtthree framebuffer. This is a simple 8-bit framebuffer for sun4m machines available in both 1024x768 (OpenBIOS) and 1152x900 (OBP) resolutions aimed at people wishing to run older Solaris versions.

## virtio

Virtio VGA card.

none

Disable VGA card.

```
-full-screen
```

Start in full screen.

**-g *widthxheight[xdepth]***

Set the initial graphical resolution and depth (PPC, SPARC only).

For PPC the default is 800x600x32.

For SPARC with the TCX graphics device, the default is 1024x768x8 with the option of 1024x768x24. For cgthree, the default is 1024x768x8 with the option of 1152x900x8 for people who wish to use OBP.

```
-vnc display[,option[,option[,...]]]
```

Normally, if QEMU is compiled with graphical window support, it displays output such as guest graphics, guest console, and the QEMU monitor in a window. With this option, you can have QEMU listen on VNC display `display` and redirect the VGA display over the VNC session. It is very useful to enable the usb tablet device when using this option (option `-device usb-tablet`). When using the VNC display, you must use the `-k` parameter to set the keyboard layout if you are not using `en-us`. Valid syntax for the display is

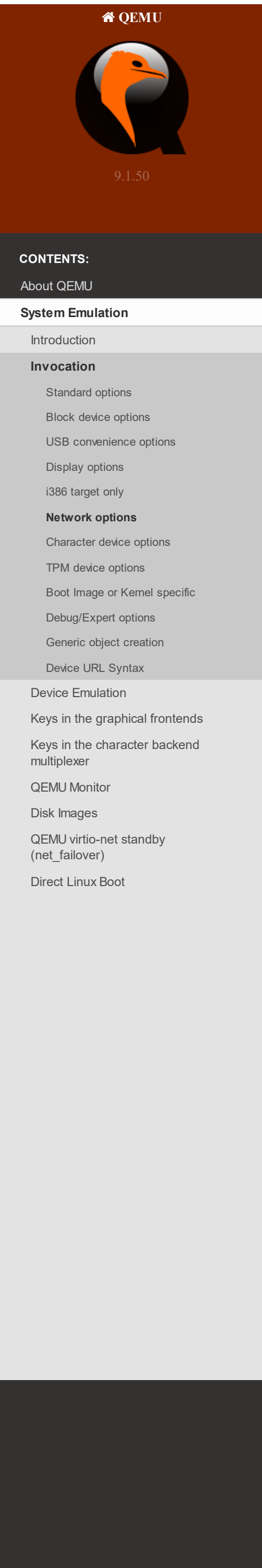
to=L

With this option, QEMU will try next available VNC displays, until the number L, if the originally defined “-vnc display” is not available, e.g. port 5900+display is already used by another application. By default, to=0.

host:d

TCP connections will only be allowed from host on display d. By convention the TCP port is 5900+d. Optionally, host can be omitted in which case the server will accept connections from any host.

```
unix:path
```



Connections will be allowed over UNIX domain sockets where path is the location of a unix socket to listen for connections on.

**none**

VNC is initialized but not started. The monitor change command can be used to later start the VNC server.

Following the display value there may be one or more option flags separated by commas. Valid options are

```
reverse=on|off
```

Connect to a listening VNC client via a “reverse” connection. The client is specified by the display. For reverse network connections (host:d, “reverse”), the d argument is a TCP port number, not a display number.

websocket=on | off

Opens an additional TCP listening port dedicated to VNC Websocket connections. If a bare websocket option is given, the Websocket port is 5700+display. An alternative port can be specified with the syntax `websocket=port`.

If host is specified connections will only be allowed from this host. It is possible to control the websocket listen address independently, using the syntax `websocket=host:port`.

Websocket could be allowed over UNIX domain socket, using the syntax `websocket=unix:path`, where `path` is the location of a unix socket to listen for connections on.

If no TLS credentials are provided, the websocket connection runs in unencrypted mode. If TLS credentials are provided, the websocket connection requires encrypted client connections.

```
password=on|off
```

Require that password based authentication is used for client connections.

The password must be set separately using the `set_password` command in the [QEMU Monitor](#). The syntax to change your password is: `set_password <protocol> <password>` where `<protocol>` could be either “vnc” or “spice”.

If you would like to change <protocol> password expiration, you should use `expire_password <protocol> <expiration-time>` where expiration time could be one of the following options: now, never, +seconds or UNIX time of expiration, e.g. +60 to make password expire in 60 seconds, or 1335196800 to make password expire on “Mon Apr 23 12:00:00 EDT 2012” (UNIX time for this date and time).

You can also use keywords “now” or “never” for the expiration time to allow <protocol> password to expire immediately or never expire.

```
password-secret=<secret-id>
```

Require that password based authentication is used for client connections, using the password provided by the secret object identified by `secret-id`.

```
tls-creds=ID
```

Provides the ID of a set of TLS credentials to use to secure the VNC server. They will apply to both the normal VNC server socket and the websocket socket (if enabled). Setting TLS credentials will cause the VNC server socket to enable the VeNCrypt auth mechanism. The credentials should have been previously created using the `-object tls-creds` argument.

```
tls-authz=ID
```

Provides the ID of the QAuthZ authorization object against which the client's x509 distinguished name will validated. This object is only resolved at time of use, so can be deleted and recreated on the fly while the VNC server is active. If missing, it will default to denying access.

```
sasl=on|off
```

Require that the client use SASL to authenticate with the VNC server. The exact choice of authentication method used is controlled from the system / user's SASL configuration file for the 'qemu' service. This is typically found in `/etc/sasl2/qemu.conf`. If running QEMU as an unprivileged user, an environment variable `SASL_CONF_PATH` can be used to make it search alternate locations for the service config. While some SASL auth methods can also provide data encryption (eg GSSAPI), it is recommended that SASL always be combined with the 'tls' and 'x509' settings to enable use of SSL and server certificates. This ensures a data encryption preventing compromise of authentication credentials. See the [VNC security](#) section in the System Emulation Users Guide for details on using SASL authentication.

```
sasl-authz=ID
```

Provides the ID of the QAuthZ authorization object against which the client's SASL username will be validated. This object is only resolved at time of use, so can be deleted and recreated on the fly while the VNC server is active. If missing, it will default to denying access.

```
acl=on|off
```

Legacy method for enabling authorization of clients against the x509 distinguished name and SASL username. It results in the creation of two `authz-list` objects with IDs of `vnc.username` and `vnc.x509dname`. The rules for these objects must be configured with the HMP ACL commands.

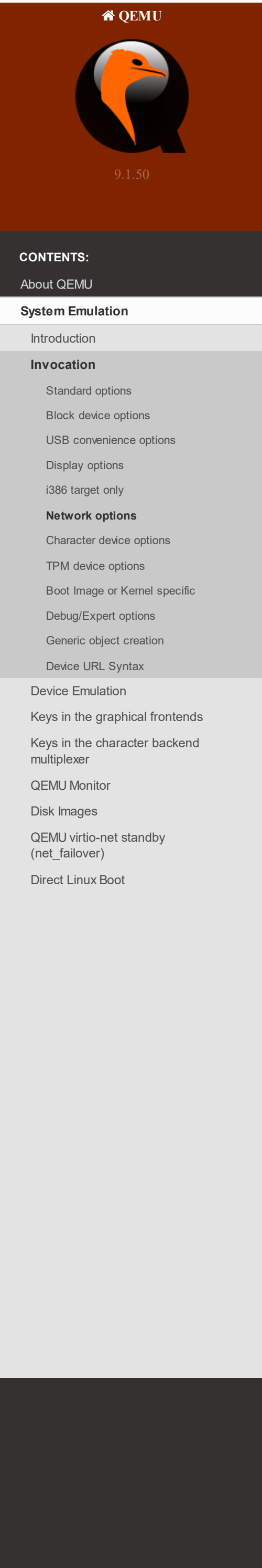
This option is deprecated and should no longer be used. The new `sasl-authz` and `tls-authz` options are a replacement.

```
lossy=on|off
```

Enable lossy compression methods (gradient, JPEG, ...). If this option is set, VNC client may receive lossy framebuffer updates depending on its encoding settings. Enabling this option can save a lot of bandwidth at the expense of quality.

```
non-adaptive=on|off
```





Disable adaptive encodings. Adaptive encodings are enabled by default. An adaptive encoding will try to detect frequently updated screen regions, and send updates in these regions using a lossy encoding (like JPEG). This can be really helpful to save bandwidth when playing videos. Disabling adaptive encodings restores the original static behavior of encodings like Tight.

```
share=[allow-exclusive|force-shared|ignore]
```

Set display sharing policy. 'allow-exclusive' allows clients to ask for exclusive access. As suggested by the rfb spec this is implemented by dropping other connections. Connecting multiple clients in parallel requires all clients asking for a shared session (vncviewer: -shared switch). This is the default. 'force-shared' disables exclusive client access. Useful for shared desktop sessions, where you don't want someone forgetting specify -shared disconnect everybody else. 'ignore' completely ignores the shared flag and allows everybody connect unconditionally. Doesn't conform to the rfb spec but is traditional QEMU behavior.

**key-delay-ms**

Set keyboard delay, for key down and key up events, in milliseconds. Default is 10. Keyboards are low-bandwidth devices, so this slowdown can help the device and guest to keep up and not lose events in case events are arriving in bulk. Possible causes for the latter are flaky network connections, or scripts for automated testing.

```
audiodev=audiodev
```

Use the specified audiodev when the VNC client requests audio transmission. When not using an -audiodev argument, this option must be omitted, otherwise it must be present and specify a valid audiodev.

```
power-control=on|off
```

Permit the remote client to issue shutdown, reboot or reset power control requests.

## i386 target only

```
-win2k-hack
```

Use it when installing Windows 2000 to avoid a disk full bug. After Windows 2000 is installed, you no longer need this option (this option slows down the IDE transfers). Synonym of `-global ide-device.win2k-install-hack=on`.

```
-no-fd-bootchk
```

Disable boot signature checking for floppy disks in BIOS. May be needed to boot from old floppy disks. Synonym of `-m fd-bootchk=off`.

```
-acpitable [sig=str][,rev=n][,oem_id=str][,oem_table_id=str][,oem_rev=n] [,asl_compiler_id=str]
[,asl_compiler_rev=n][,data=file1[:file2]...]
```

Add ACPI table with specified header fields and context from specified files. For file=, take whole ACPI table from the specified files, including all ACPI headers (possible overridden by other options). For data=, only data portion of the table is used, all header information is specified in the command line. If a SLIC table is supplied to QEMU, then the SLIC's oem\_id and oem\_table\_id fields will override the same in the RSDT and the FADT (a.k.a. FACP), in order to ensure the field matches required by the Microsoft SLIC spec and the ACPI spec.

```
-smbios file=binary
```

Load SMBIOS entry from binary file.

```
-smbios type=0[,vendor=str][,version=str][,date=str][,release=%d.%d][,uefi=on|off]
```

## Specify SMBIOS type 0 fields

```
-smbios type=1[,manufacturer=str][,product=str][,version=str][,serial=str][,uuid=uuid][,sku=str]
[,family=str]
```

## Specify SMBIOS type 1 fields

```
-smbios type=2[,manufacturer=str][,product=str][,version=str][,serial=str][,asset=str][,location=str]
```

## Specify SMBIOS type 2 fields

```
-smbios type=3[,manufacturer=str][,version=str][,serial=str][,asset=str][,sku=str]
```

## Specify SMBIOS type 3 fields

```
-smbios type=4[,sock_pfx=str][,manufacturer=str][,version=str][,serial=str][,asset=str][,part=str]
[,processor-family=%d][,processor-id=%d]
```

## Specify SMBIOS type 4 fields

```
-smbios type=9[,slot_designation=str][,slot_type=%d][,slot_data_bus_width=%d][,current_usage=%d][,slot_length=%d][,slot_id=%d][,slot_characteristics1=%d][,slot_characteristics12=%d][,pci_device=str]
```

## Specify SMBIOS type 9 fields

```
-smbios type=11[,value=str][,path=filename]
```

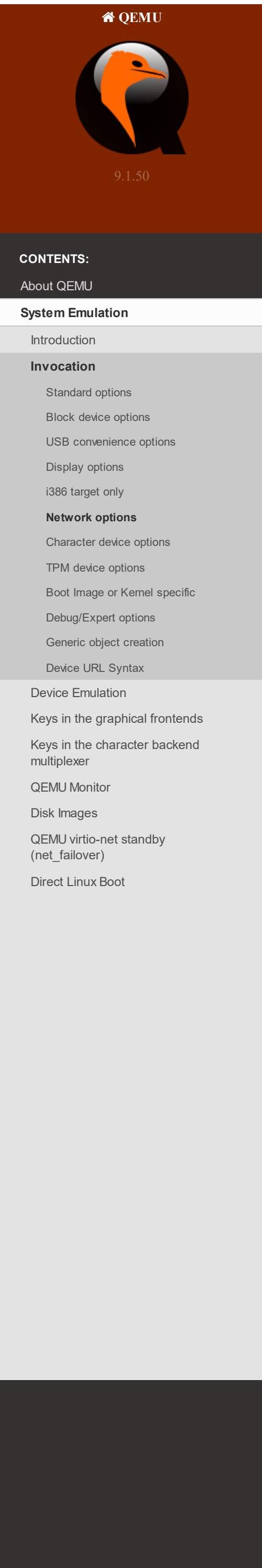
## Specify SMBIOS type 11 fields

This argument can be repeated multiple times, and values are added in the order they are parsed. Applications intending to use OEM strings data are encouraged to use their application name as a prefix for the value string. This facilitates passing information for multiple applications concurrently.

The `value=str` syntax provides the string data inline, while the `path=filename` syntax loads data from a file on disk. Note that the file is not permitted to contain any NUL bytes.

Both the `value` and `path` options can be repeated multiple times and will be added to the SMBIOS table in the order in which they appear.





```
ipv6-host=addr
```

Specify the guest-visible IPv6 address of the host. Default is the 2nd IPv6 in the guest network, i.e. xxxx::2.

```
restrict=on|off
```

If this option is enabled, the guest will be isolated, i.e. it will not be able to contact the host and no guest IP packets will be routed over the host to the outside. This option does not affect any explicitly set forwarding rules.

hostname=name

Specifies the client hostname reported by the built-in DHCP server.

```
dhcpstart=addr
```

Specify the first of the 16 IPs the built-in DHCP server can assign. Default is the 15th to 31st IP in the guest network, i.e. x.x.x.15 to x.x.x.31.

**dns=addr**

Specify the guest-visible address of the virtual nameserver. The address must be different from the host address. Default is the 3rd IP in the guest network, i.e. x.x.x.3.

**ipv6-dns=addr**

Specify the guest-visible address of the IPv6 virtual nameserver. The address must be different from the host address. Default is the 3rd IP in the guest network, i.e. xxxx:3.

**dnssearch=domain**

Provides an entry for the domain-search list sent by the built-in DHCP server. More than one domain suffix can be transmitted by specifying this option multiple times. If supported, this will cause the guest to automatically try to append the given domain suffix(es) in case a domain name can not be resolved.

Example:

```
qemu-system-x86_64 -nic user,dnssearch=mgmt.example.org,dnssearch=example.org
```

```
domainname=domain
```

Specifies the client domain name reported by the built-in DHCP server.

```
tftp=dir
```

When using the user mode network stack, activate a built-in TFTP server. The files in `dir` will be exposed as the root of a TFTP server. The TFTP client on the guest must be configured in binary mode (use the command `bin` of the Unix TFTP client). The built-in TFTP server is read-only; it does not implement any command for writing files. QEMU will not write to this directory.

```
tftp-server-name=name
```

In BOOTP reply, broadcast name as the “TFTP server name” (RFC2132 option 66). This can be used to advise the guest to load boot files or configurations from a different server than the host address.

```
bootfile=file
```

When using the user mode network stack, broadcast file as the BOOTP filename. In conjunction with `tftp`, this can be used to network boot a guest from a local directory.

Example (using pxelinux):

```
qemu-system-x86_64 -hda linux.img -boot n -device e1000,netdev=n1 \
-netdev user,id=n1,tftp=/path/to/tftp/files,bootfile=pxelinux.0
```

```
smb=dir[,smbserver=addr]
```

When using the user mode network stack, activate a built-in SMB server so that Windows OSES can access to the host files in dir transparently. The IP address of the SMB server can be set to addr. By default the 4th IP in the guest network is used, i.e. x.x.x.4.

In the guest Windows OS, the line:

#### 10.0.2.4 smbserver

must be added in the file C:\WINDOWS\LMHOSTS (for windows 9x/Me) or C:\WINNT\SYSTEM32\DRIVERS\ETC\LMHOSTS (Windows NT/2000).

Then `dir` can be accessed in `\\smbserver\gemu`.

Note that a SAMBA server must be installed on the host OS.

```
hostfwd=[tcp|udp]:[hostaddr]:hostport-[guestaddr]:guestport
```

Redirect incoming TCP or UDP connections to the host port `hostport` to the guest IP address `guestaddr` on guest port `guestport`. If `guestaddr` is not specified, its value is `x.x.x.15` (default first address given by the built-in DHCP server). By specifying `hostaddr`, the rule can be bound to a specific host interface. If no connection type is set, TCP is used. This option can be given multiple times.

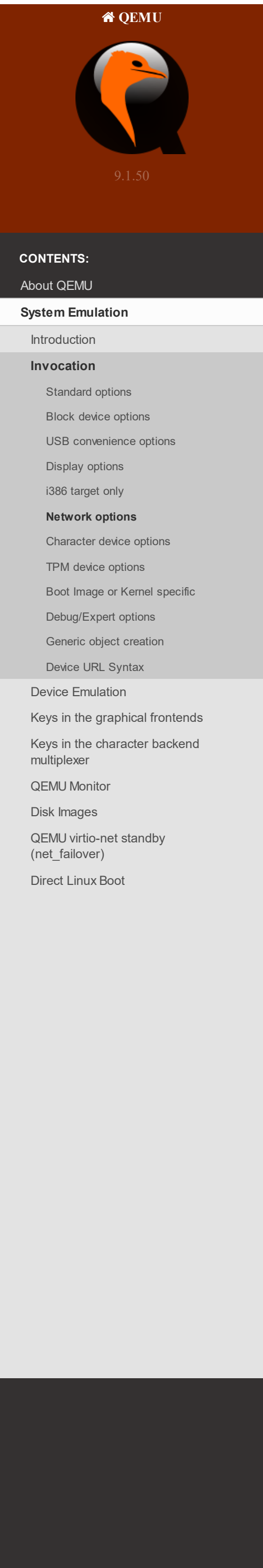
For example, to redirect host X11 connection from screen 1 to guest screen 0, use the following:











```
# launch another QEMU instance on same "bus"
qemu-system-x86_64 linux.img \
    -device virtio-net,netdev=net0,mac=52:54:00:12:34:57 \
    -netdev dgram,id=net0,remote.type=inet,remote.host=224.0.0.1,remote.port=1234
# launch yet another QEMU instance on same "bus"
qemu-system-x86_64 linux.img \
    -device virtio-net,netdev=net0,mac=52:54:00:12:34:58 \
    -netdev dgram,id=net0,remote.type=inet,remote.host=224.0.0.1,remote.port=1234
```

```
-netdev dgram,id=str,remote.type=inet,remote.host=maddr,remote.port=port[,local.type=fd,local.str=file-descriptor]
```

Configure a network backend to connect to a multicast address using a UDP socket file descriptor.

```
remote.host=maddr,remote.port=port
```

multicast address

```
local.str=file-descriptor
```

File descriptor to use to send packets

```
-netdev
dgram,id=str,local.type=inet,local.host=addr,local.port=port[,remote.type=inet,remote.host=addr,remote.port=port]
```

Configure a network backend to connect to another QEMU virtual machine or a proxy using a datagram oriented unix domain socket.

```
local.host=addr,local.port=port
```

IP address to use to send the packets from

```
remote.host=addr,remote.port=port
```

Destination IP address

Example (two guests connected using an UDP/IP socket):

```
# first VM
qemu-system-x86_64 linux.img \
    -device virtio-net,netdev=net0,mac=52:54:00:12:34:56 \
    -netdev dgram,id=net0,local.type=inet,local.host=localhost,local.port=1234,remote.type=inet,remote.host=192.168.1.100
# second VM
qemu-system-x86_64 linux.img \
    -device virtio-net,netdev=net0,mac=52:54:00:12:34:56 \
    -netdev dgram,id=net0,local.type=inet,local.host=localhost,local.port=1235,remote.type=inet,remote.host=192.168.1.100
```

```
-netdev dgram,id=str,local.type=unix,local.path=path[,remote.type=unix,remote.path=path]
```

Configure a network backend to connect to another QEMU virtual machine or a proxy using a datagram oriented unix socket.

```
local.path=path
```

filesystem path to use to bind the socket

```
remote.path=path
```

filesystem path to use as a destination (see `sendto(2)`)

Example (two guests connected using an UDP/UNIX socket):

```
# first VM
qemu-system-x86_64 linux.img \
    -device virtio-net,netdev=net0,mac=52:54:00:12:34:56 \
    -netdev dgram,id=net0,local.type=unix,local.path=/tmp/qemu0,remote.type=unix,remote.path=/tmp/qemu1

# second VM
qemu-system-x86_64 linux.img \
    -device virtio-net,netdev=net0,mac=52:54:00:12:34:57 \
    -netdev dgram,id=net0,local.type=unix,local.path=/tmp/qemu1,remote.type=unix,remote.path=/tmp/qemu0
```

```
-netdev dgram,id=str,local.type=fd,local.str=file-descriptor
```

Configure a network backend to connect to another QEMU virtual machine or a proxy using a datagram oriented socket file descriptor.

```
local.str=file-descriptor
```

File descriptor to use to send packets

```
-netdev l2tpv3,id=id,src=srcaddr,dst=dstaddr[,srcport=srcport]
[,dstport=dstport],txsession=txsession[,rxsession=rxsession][,ipv6=on|off][,udp=on|off][,cookie64=on|off]
[,counter=on|off][,pincounter=on|off][,txcookie=txcookie][,rxcookie=rxcookie][,offset=offset]
```

Configure a L2TPv3 pseudowire host network backend. L2TPv3 (RFC3931) is a popular protocol to transport Ethernet (and other Layer 2) data frames between two systems. It is present in routers, firewalls and the Linux kernel (from version 3.3 onwards).

This transport allows a VM to communicate to another VM, router or firewall directly.

```
src=srcaddr
```

source address (mandatory)

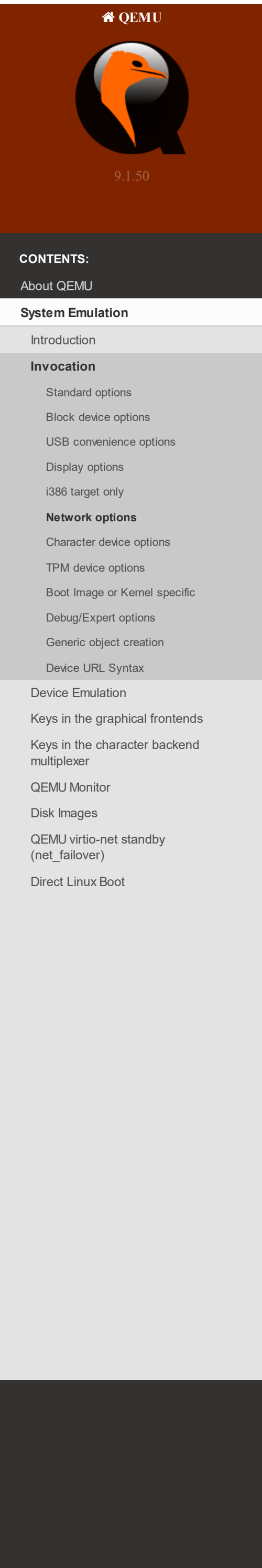
```
dst=dstaddr
```

destination address (mandatory)









```
-chardev stdio,mux=on,id=char0 \  
-mon chardev=char0,mode=readline \  
-parallel chardev:char0 \  
-chardev tcp,...,mux=on,id=char1 \  
-serial chardev:char1 \  
-serial chardev:char1
```

When you're using a multiplexed character device, some escape sequences are interpreted in the input. See the chapter about [Keys in the character backend multiplexer](#) in the System Emulation Users Guide for more details.

Note that some other command line options may implicitly create multiplexed character backends; for instance `-serial mon:stdio` creates a multiplexed stdio backend connected to the serial port and the QEMU monitor, and `-nographic` also multiplexes the console and the monitor to stdio.

There is currently no support for multiplexing in the other direction (where a single QEMU front end takes input and output from multiple chardevs).

Every backend supports the `logfile` option, which supplies the path to a file to record all data transmitted via the backend. The `logappend` option controls whether the log file will be truncated or appended to when opened.

available backends are:

```
harddev null,id=id
```

A void device. This device will not emit any data, and will drop any data it receives. The null backend does not take any options.

```
hardev socket,id=id[,TCP options or unix options][,server=on|off][,wait=on|off][,telnet=on|off]
websocket=on|off[,reconnect-ms=milliseconds][,tls-creds=id][,tls-authz=id]
```

Create a two-way stream socket, which can be either a TCP or a unix socket. A unix socket will be created if `path` is specified. Behaviour is undefined if TCP options are specified for a unix socket.

`server=on|off` specifies that the socket shall be a listening socket.

`wait=on|off` specifies that QEMU should not block waiting for a client to connect to a listening socket.

`telnet=on|off` specifies that traffic on the socket should interpret telnet escape sequences.

`websocket=on|off` specifies that the socket uses WebSocket protocol for communication.

`reconnect-ms` sets the timeout for reconnecting on non-server sockets when the remote end goes away. qemu will delay this many milliseconds and then attempt to reconnect. Zero disables reconnecting, and is the default.

`tls-creds` requests enablement of the TLS protocol for encryption, and specifies the id of the TLS credentials to use for the handshake. The credentials must be previously created with the `-object tls-creds` argument.

`tls-auth` provides 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.

TCP and unix socket options are given below:

TCP options: port=port[,host=host][,to=to][,ipv4=on|off][,ipv6=on|off][,nodelay=on|off]

host for a listening socket specifies the local address to be bound. For a connecting socket species the remote host to connect to. host is optional for listening sockets. If not specified it defaults to 0.0.0.0.

port for a listening socket specifies the local port to be bound. For a connecting socket specifies the port on the remote host to connect to. port can be given as either a port number or a service name. port is required.

to is only relevant to listening sockets. If it is specified, and port cannot be bound, QEMU will attempt to bind to subsequent ports up to and including to until it succeeds. to must be specified as a port number.

ip<sub>v</sub>4=on/off and ip<sub>v</sub>6=on/off specify that either IPv4 or IPv6 must be used. If neither is specified the socket may use either protocol.

`nodelay=on|off` disables the Nagle algorithm.

```
unix options: path=path[,abstract=on|off][,tight=on|off]
```

path specifies the local path of the unix socket. path is required. abstract=on|off specifies the use of the abstract socket namespace, rather than the filesystem. Optional, defaults to false. tight=on|off sets the socket length of abstract sockets to their minimum, rather than the full sun\_path length. Optional, defaults to true.

```
hardev udp,id=id[,host=host],port=port[,localaddr=localaddr][,localport=localport][,ipv4=on|off]
[,ipv6=on|off]
```

Sends all traffic from the guest to a remote host over UDP.

host specifies the remote host to connect to. If not specified it defaults to localhost.

port specifies the port on the remote host to connect to. port is required.

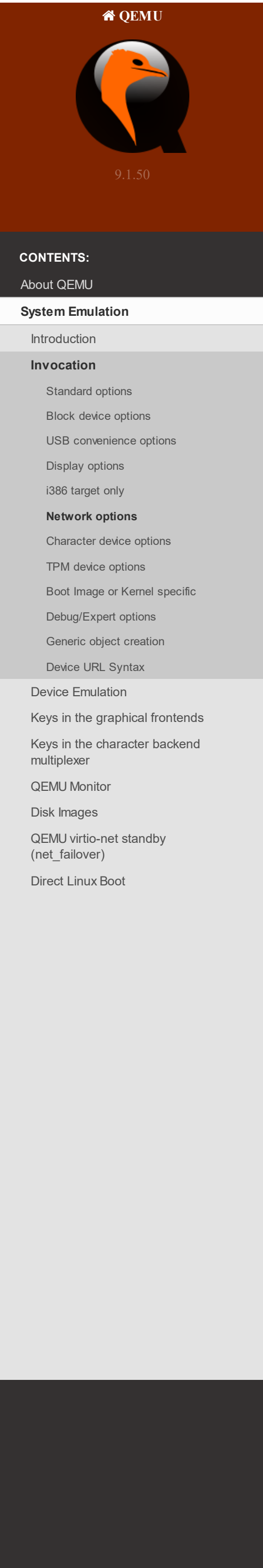
`localaddr` specifies the local address to bind to. If not specified it defaults to 0.0.0.0.

`localport` specifies the local port to bind to. If not specified any available local port will be used.

ipv4=on|off and ipv6=on|off specify that either IPv4 or IPv6 must be used. If neither is specified the device may use either protocol.

```
hardev msmouse,id=id
```





name name of spice port to connect to

Connect to a spice port, allowing a Spice client to handle the traffic identified by a name (preferably a fqdn).

## TPM device options

The general form of a TPM device option is:

```
-tpmdev backend,id=id[,options]
```

The specific backend type will determine the applicable options. The `-tpmdev` option creates the TPM backend and requires a `-device` option that specifies the TPM frontend interface model.

Use `-tpmdev help` to print all available TPM backend types.

The available backends are:

```
-tpmdev passthrough,id=id,path=path,cancel-path=cancel-path
```

(Linux-host only) Enable access to the host's TPM using the passthrough driver.

path specifies the path to the host's TPM device, i.e., on a Linux host this would be `/dev/tpm0`. path is optional and by default `/dev/tpm0` is used.

`cancel-path` specifies the path to the host TPM device's `sysfs` entry allowing for cancellation of an ongoing TPM command. `cancel-path` is optional and by default QEMU will search for the `sysfs` entry to use.

### Some notes about using the host's TPM with the passthrough driver:

The TPM device accessed by the passthrough driver must not be used by any other application on the host.

Since the host's firmware (BIOS/UEFI) has already initialized the TPM, the VM's firmware (BIOS/UEFI) will not be able to initialize the TPM again and may therefore not show a TPM-specific menu that would otherwise allow the user to configure the TPM, e.g., allow the user to enable/disable or activate/deactivate the TPM. Further, if TPM ownership is released from within a VM then the host's TPM will get disabled and deactivated. To enable and activate the TPM again afterwards, the host has to be rebooted and the user is required to enter the firmware's menu to enable and activate the TPM. If the TPM is left disabled and/or deactivated most TPM commands will fail.

To create a passthrough TPM use the following two options:

```
-tpmdev passthrough,id=tpm0 -device tpm-tis,tpmdev=tpm0
```

Note that the `-tpmdev id` is `tpm0` and is referenced by `tpmdev=tpm0` in the device option.

```
-tpmdev emulator,id=id,chardev=dev
```

(Linux-host only) Enable access to a TPM emulator using Unix domain socket based chardev backend.

`chardev` specifies the unique ID of a character device backend that provides connection to the software TPM server.

To create a TPM emulator backend device with chardev socket backend:

```
-chardev socket,id=chrtpm,path=/tmp/swtpm-sock -tpmdev emulator,id=tpm0,chardev=chrtpm -device tpm-tis,tpmdev=tpm0
```

## Boot Image or Kernel specific

There are broadly 4 ways you can boot a system with QEMU.

- specify a firmware and let it control finding a kernel
- specify a firmware and pass a hint to the kernel to boot
- direct kernel image boot
- manually load files into the guest's address space

The third method is useful for quickly testing kernels but as there is no firmware to pass configuration information to the kernel the hardware must either be probeable, the kernel built for the exact configuration or passed some configuration data (e.g. a DTB blob) which tells the kernel what drivers it needs. This exact details are often hardware specific.

The final method is the most generic way of loading images into the guest address space and used mostly for bare metal type development where the reset vectors of the processor are taken into account.

For x86 machines and some other architectures -bios will generally do the right thing with whatever it is given. For other machines the more strict -pflash option needs an image that is sized for the flash device for the given machine type.

Please see the [QEMU System Emulator Targets](#) section of the manual for more detailed documentation.

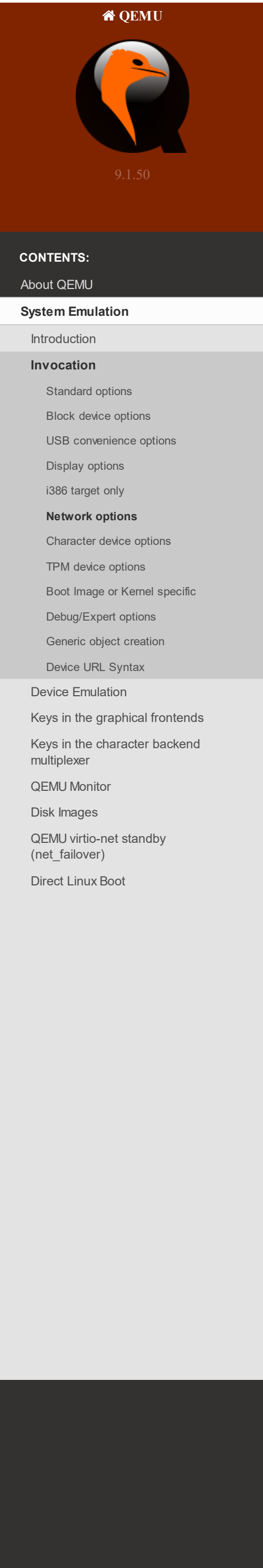
```
-bios file
```

Set the filename for the BIOS.

```
-pflash file
```

Use file as a parallel flash image.





The kernel options were designed to work with Linux kernels although other things (like hypervisors) can be packaged up as a kernel executable image. The exact format of a executable image is usually architecture specific.

The way in which the kernel is started (what address it is loaded at, what if any information is passed to it via CPU registers, the state of the hardware when it is started, and so on) is also architecture specific. Typically it follows the specification laid down by the Linux kernel for how kernels for that architecture must be started.

```
-kernel bzImage
```

Use bzImage as kernel image. The kernel can be either a Linux kernel or in multiboot format.

```
-append cmdline
```

Use cmdline as kernel command line

```
-initrd file
```

Use file as initial ram disk.

```
-initrd "file1 arg=foo,file2"
```

This syntax is only available with multiboot.

Use `file1` and `file2` as modules and pass `arg=foo` as parameter to the first module. Commas can be provided in module parameters by doubling them on the command line to escape them:

```
-initrd "bzImage earlyprintk=xen,,keep root=/dev/xvda1,initrd.img"
```

Multiboot only. Use bzImage as the first module with “earlyprintk=xen,keep root=/dev/xvda1” as its command line, and initrd.img as the second module.

```
-dtb file
```

Use file as a device tree binary (dtb) image and pass it to the kernel on boot.

Finally you can also manually load images directly into the address space of the guest. This is most useful for developers who already know the layout of their guest and take care to ensure something sane will happen when the reset vector executes.

The generic loader can be invoked by using the loader device:

```
-device loader,addr=<addr>,data=<data>,data-len=<data-len>[,data-be=<data-be>][,cpu-num=<cpu-num>]
```

there is also the guest loader which operates in a similar way but tweaks the DTB so a hypervisor loaded via `-kernel` can find where the guest image is:

```
-device guest-loader,addr=<addr>[,kernel=<path>,[bootargs=<arguments>]][,initrd=<path>]
```

## Debug/Expert options

```
-compat [deprecated-input=@var{input-policy}][,deprecated-output=@var{output-policy}]
```

Set policy for handling deprecated management interfaces (experimental):

**deprecated-input=accept (default)**

### Accept deprecated commands and arguments

```
deprecated-input=reject
```

## Reject deprecated commands and arguments

```
deprecated-input=crash
```

## Crash on deprecated commands and arguments

```
deprecated-output=accept (default)
```

### Emit deprecated command results and events

```
deprecated-output=hide
```

## Suppress deprecated command results and events

Limitation: covers only syntactic aspects of QMP.

```
-compat [unstable-input=@var{input-policy}][,unstable-output=@var{output-policy}]
```

Set policy for handling unstable management interfaces (experimental):

**unstable-input=accept (default)**

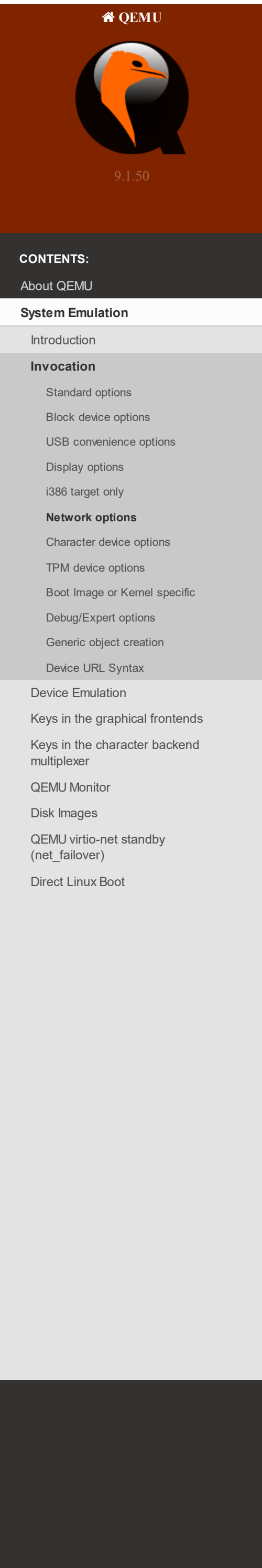
## Accept unstable commands and arguments

```
unstable-input=reject
```

## Reject unstable commands and arguments

```
unstable-input=crash
```

## Crash on unstable commands and arguments



**unstable-output=accept (default)**

### Emit unstable command results and events

```
unstable-output=hide
```

## Suppress unstable command results and events

Limitation: covers only syntactic aspects of QMP.

```
-fw_cfg [name=]name,file=file
```

Add named `fw_cfg` entry with contents from file `file`. If the filename contains comma, you must double it (for instance, “`file=my,,file`” to use file “`my,file`”).

```
-fw_cfg [name=]name,string=str
```

Add named `fw_cfg` entry with contents from string `str`. If the string contains comma, you must double it (for instance, “string=my,,string” to use file “my,string”).

The terminating NUL character of the contents of `str` will not be included as part of the `fw_cfg` item data. To insert contents with embedded NUL characters, you have to use the `file` parameter.

The fw\_cfg entries are passed by QEMU through to the guest.

Example:

```
-fw_cfg name=opt/com.mycompany/blob,file=./my_blob.bin
```

creates an `fw_cfg` entry named `opt/com.mycompany/blob` with contents from `./my_blob.bin`.

```
-serial dev
```

Redirect the virtual serial port to host character device dev. The default device is `vc` in graphical mode and `stdio` in non graphical mode.

This option can be used several times to simulate multiple serial ports.

You can use `-serial none` to suppress the creation of default serial devices.

Available character devices are:

## vc[:WxH]

Virtual console. Optionally, a width and height can be given in pixel with

vc:800x600

It is also possible to specify width or height in characters:

vc:80Cx24C

```
pty[:path]
```

[Linux only] Pseudo TTY (a new PTY is automatically allocated).

If path is specified, QEMU will create a symbolic link at that location which points to the new PTY device.

This avoids having to make QMP or HMP monitor queries to find out what the new PTY device path is.

Note that while QEMU will remove the symlink when it exits gracefully, it will not do so in case of crashes or on certain startup errors. It is recommended that the user checks and removes the symlink after QEMU terminates to account for this.

none

No device is allocated. Note that for machine types which emulate systems where a serial device is always present in real hardware, this may be equivalent to the `null` option, in that the serial device is still present but all output is discarded. For boards where the number of serial ports is truly variable, this suppresses the creation of the device.

```

null

```

A guest will see the UART or serial device as present in the machine, but all output is discarded, and there is no input. Conceptually equivalent to redirecting the output to `/dev/null`.

```
chardev:id
```

Use a named character device defined with the `-chardev` option.

```
/dev/XXX
```

[Linux only] Use host tty, e.g. /dev/ttyS0. The host serial port parameters are set according to the emulated ones.

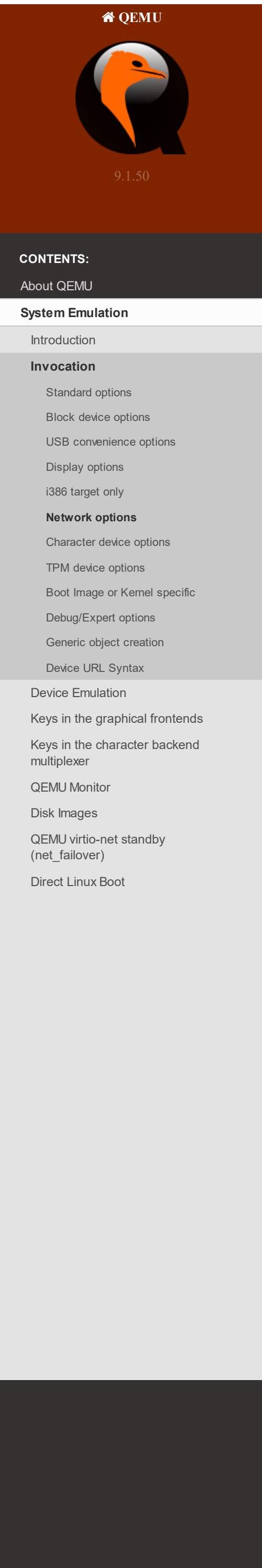
```
/dev/parportN
```

[Linux only, parallel port only] Use host parallel port N. Currently SPP and EPP parallel port features can be used.

```
file:filename
```

Write output to filename. No character can be read.

studio



[Unix only] standard input/output

```
pipe:filename
```

name pipe filename

## COMn

[Windows only] Use host serial port n

```
udp:[remote_host]:remote_port[@[src_ip]:src_port]
```

This implements UDP Net Console. When `remote_host` or `src_ip` are not specified they default to `0.0.0.0`. When not using a specified `src_port` a random port is automatically chosen.

If you just want a simple readonly console you can use netcat or nc, by starting QEMU with: `-serial udp::4555` and nc as: `nc -u -l -p 4555`. Any time QEMU writes something to that port it will appear in the netconsole session.

If you plan to send characters back via netconsole or you want to stop and start QEMU a lot of times, you should have QEMU use the same source port each time by using something like `-serial udp::4555@:4556` to QEMU. Another approach is to use a patched version of netcat which can listen to a TCP port and send and receive characters via udp. If you have a patched version of netcat which activates telnet remote echo and single char transfer, then you can use the following options to set up a netcat redirector to allow telnet on port 5555 to access the QEMU port.

### QEMU Options:

```
-serial udp::4555@:4556
```

## netcat options:

-u -P 4555 -L 0.0.0.0:4556 -t -p 5555 -I -T

```
telnet options:
```

localhost 5555

```
tcp:[host]:port[,server=on|off][,wait=on|off][,nodelay=on|off][,reconnect-ms=milliseconds]
```

The TCP Net Console has two modes of operation. It can send the serial I/O to a location or wait for a connection from a location. By default the TCP Net Console is sent to host at the port. If you use the `server=on` option QEMU will wait for a client socket application to connect to the port before continuing, unless the `wait=on|off` option was specified. The `nodelay=on|off` option disables the Nagle buffering algorithm. The `reconnect-ms` option only applies if `server=no` is set, if the connection goes down it will attempt to reconnect at the given interval. If host is omitted, 0.0.0.0 is assumed. Only one TCP connection at a time is accepted. You can use `telnet=on` to connect to the corresponding character device.

Example to send tcp console to 192.168.0.2 port 4444

```
-serial tcp:192.168.0.2:4444
```

### Example to listen and wait on port 4444 for connection

```
-serial tcp::4444,server=on
```

Example to not wait and listen on ip 192.168.0.100 port 4444

```
-serial tcp:192.168.0.100:4444,server=on,wait=off
```

```
telnet:host:port[,server=on|off][,wait=on|off][,nodelay=on|off]
```

The telnet protocol is used instead of raw tcp sockets. The options work the same as if you had specified `-serial tcp`. The difference is that the port acts like a telnet server or client using telnet option negotiation. This will also allow you to send the `MAGIC_SYSRQ` sequence if you use a telnet that supports sending the break sequence. Typically in unix telnet you do it with `Control-]` and then type “send break” followed by pressing the enter key.

```
websocket:host:port,server=on[,wait=on|off][,nodelay=on|off]
```

The WebSocket protocol is used instead of raw tcp socket. The port acts as a WebSocket server. Client mode is not supported.

```
unix:path[,server=on|off][,wait=on|off][,reconnect-ms=milliseconds]
```

A unix domain socket is used instead of a tcp socket. The option works the same as if you had specified `-serial tcp` except the unix domain socket path is used for connections.

```
mon:dev_string
```

This is a special option to allow the monitor to be multiplexed onto another serial port. The monitor is accessed with key sequence of Control-a and then pressing c. dev\_string should be any one of the serial devices specified above. An example to multiplex the monitor onto a telnet server listening on port 4444 would be:

```
-serial mon:telnet::4444,server=on,wait=off
```

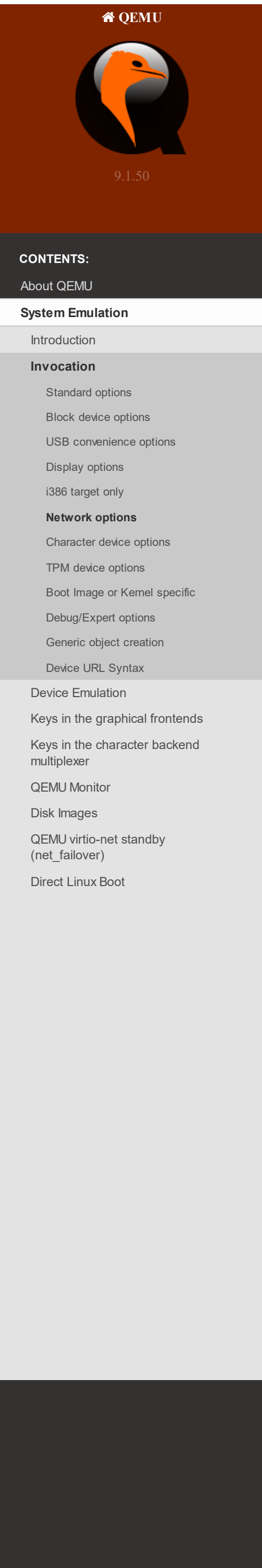
When the monitor is multiplexed to stdio in this way, Ctrl+C will not terminate QEMU any more but will be passed to the guest instead.

braille

Braille device. This will use BrlAPI to display the braille output on a real or fake device.

## msmouse

Three button serial mouse. Configure the guest to use Microsoft protocol.



```
-parallel dev
```

Redirect the virtual parallel port to host device dev (same devices as the serial port). On Linux hosts, /dev/parportN can be used to use hardware devices connected on the corresponding host parallel port.

This option can be used several times to simulate up to 3 parallel ports.

Use `-parallel none` to disable all parallel ports.

```
-monitor dev
```

Redirect the monitor to host device `dev` (same devices as the serial port). The default device is `vc` in graphical mode and `stdio` in non graphical mode. Use `-monitor none` to disable the default monitor.

```
-qmp dev
```

Like `-monitor` but opens in ‘control’ mode. For example, to make QMP available on localhost port 4444:

```
-qmp tcp:localhost:4444,server=on,wait=off
```

Not all options are configurable via this syntax; for maximum flexibility use the `-mon` option and an accompanying `-chardev`.

```
-qmp-pretty dev
```

Like -qmp but uses pretty JSON formatting.

```
-mon [chardev=]name[,mode=readline|control][,pretty[=on|off]]
```

Set up a monitor connected to the chardev name. QEMU supports two monitors: the Human Monitor Protocol (HMP; for human interaction), and the QEMU Monitor Protocol (QMP; a JSON RPC-style protocol). The default is HMP; `mode=control` selects QMP instead. `pretty` is only valid when `mode=control`, turning on JSON pretty printing to ease human reading and debugging.

For example:

```
-chardev socket,id=mon1,host=localhost,port=4444,server=on,wait=off \
-mon chardev=mon1,mode=control,pretty=on
```

enables the QMP monitor on localhost port 4444 with pretty-printing.

```
-debugcon dev
```

Redirect the debug console to host device `dev` (same devices as the serial port). The debug console is an I/O port which is typically port `0xe9`; writing to that I/O port sends output to this device. The default device is `vc` in graphical mode and `stdio` in non graphical mode.

**-pidfile file**

Store the QEMU process PID in file. It is useful if you launch QEMU from a script.

```
--preconfig
```

Pause QEMU for interactive configuration before the machine is created, which allows querying and configuring properties that will affect machine initialization. Use QMP command 'x-exit-preconfig' to exit the preconfig state and move to the next state (i.e. run guest if -S isn't used or pause the second time if -S is used). This option is experimental.

**-S**

Do not start CPU at startup (you must type 'c' in the monitor).

```
-overcommit mem-lock=on|off
```

```
-overcommit cpu-pm=on|off
```

Run qemu with hints about host resource overcommit. The default is to assume that host overcommits all resources.

Locking qemu and guest memory can be enabled via `mem-lock=on` (disabled by default). This works when host memory is not overcommitted and reduces the worst-case latency for guest.

Guest ability to manage power state of host cpus (increasing latency for other processes on the same host cpu, but decreasing latency for guest) can be enabled via `cpu-pm=on` (disabled by default). This works best when host CPU is not overcommitted. When used, host estimates of CPU cycle and power utilization will be incorrect, not taking into account guest idle time.

```
-gdb dev
```

Accept a gdb connection on device dev (see the [GDB usage](#) chapter in the System Emulation Users Guide). Note that this option does not pause QEMU execution – if you want QEMU to not start the guest until you connect with gdb and issue a `continue` command, you will need to also pass the `-S` option to QEMU.

The most usual configuration is to listen on a local TCP socket:

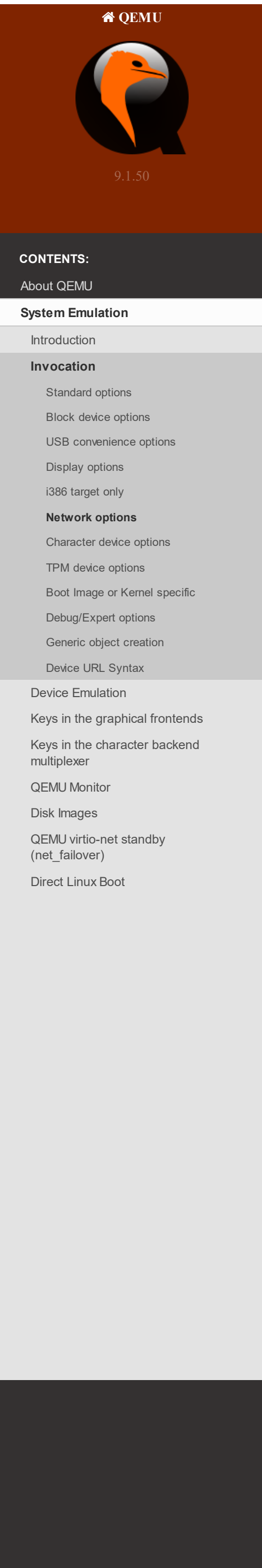
```
-gdb tcp::3117
```

but you can specify other backends; UDP, pseudo TTY, or even stdio are all reasonable use cases. For example, a stdio connection allows you to start QEMU from within gdb and establish the connection via a pipe:

```
(gdb) target remote | exec qemu-system-x86_64 -gdb stdio ...
```

-S





Shorthand for `-gdb tcp::1234`, i.e. open a gdbserver on TCP port 1234 (see the [GDB usage](#) chapter in the System Emulation Users Guide).

```
-d item1[,...]
```

Enable logging of specified items. Use '-d help' for a list of log items.

```
-D logfile
```

### Output log in logfile instead of to stderr

```
-dfilter range1[,...]
```

Filter debug output to that relevant to a range of target addresses. The filter spec can be either start+size, start-size or start..end where start end and size are the addresses and sizes required. For example:

```
-dfilter 0x8000..0x8fff,0xffffffffc000080000+0x200,0xffffffffc000060000-0x1000
```

Will dump output for any code in the 0x1000 sized block starting at 0x8000 and the 0x200 sized block starting at 0xfffffc000080000 and another 0x1000 sized block starting at 0xfffffc00005f000.

## -seed number

Force the guest to use a deterministic pseudo-random number generator, seeded with number. This does not affect crypto routines within the host.

**-L path**

Set the directory for the BIOS, VGA BIOS and keymaps.

To list all the data directories, use `-L help`.

```
-enable-kvm
```

Enable KVM full virtualization support. This option is only available if KVM support is enabled when compiling.

```
-xen-domid id
```

Specify xen guest domain id (XEN only).

**-xen-attach**

Attach to existing xen domain. libxl will use this when starting QEMU (XEN only). Restrict set of available xen operations to specified domain id (XEN only).

**-no-reboot**

Exit instead of rebooting.

```
-no-shutdown
```

Don't exit QEMU on guest shutdown, but instead only stop the emulation. This allows for instance switching to monitor to commit changes to the disk image.

```
-action event=action
```

The `action` parameter serves to modify QEMU's default behavior when certain guest events occur. It provides a generic method for specifying the same behaviors that are modified by the `-no-reboot` and `-no-shutdown` parameters.

Examples:

```
-action panic=none -action reboot=shutdown,shutdown=pause -device i6300esb -action watchdog=pause
```

```
-loadvm file
```

Start right away with a saved state (loadvm in monitor)

```
-daemonize
```

Daemonize the QEMU process after initialization. QEMU will not detach from standard IO until it is ready to receive connections on any of its devices. This option is a useful way for external programs to launch QEMU without having to cope with initialization race conditions.

```
-option-rom file
```

Load the contents of file as an option ROM. This option is useful to load things like EtherBoot.

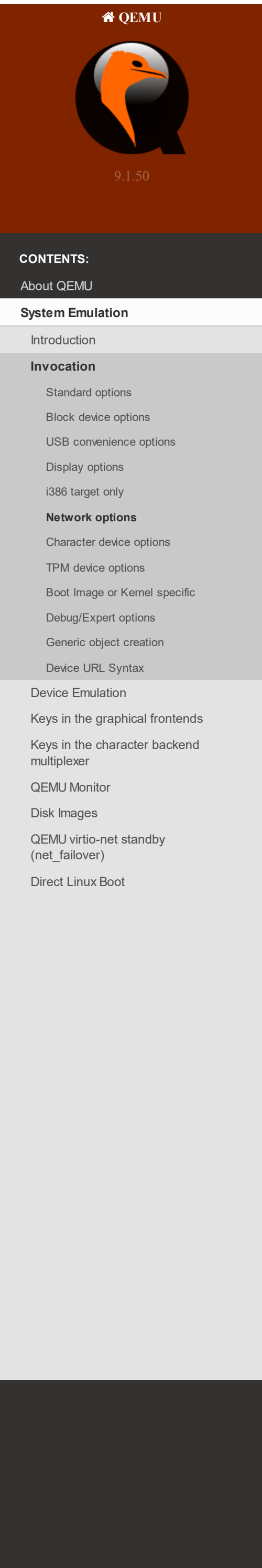
```
-rtc [base=utc|localtime|datetime][,clock=host|rt|vm][,driftfix=none|slew]
```

Specify `base` as `utc` or `localtime` to let the RTC start at the current UTC or local time, respectively. `localtime` is required for correct date in MS-DOS or Windows. To start at a specific point in time, provide `datetime` in the format `2006-06-17T16:01:21` or `2006-06-17`. The default `base` is UTC.

By default the RTC is driven by the host system time. This allows using of the RTC as accurate reference clock inside the guest, specifically if the host time is smoothly following an accurate external reference clock, e.g. via NTP. If you want to isolate the guest time from the host, you can set `clock` to `rt` instead, which provides a host monotonic clock if host support it. To even prevent the RTC from progressing during suspension, you can set `clock` to `vm` (virtual clock). ‘`clock=vm`’ is recommended especially in `icount` mode in order to preserve determinism; however, note that in `icount` mode the speed of the virtual clock is variable and can in general differ from the host clock.

Enable `driftfix` (i386 targets only) if you experience time drift problems, specifically with Windows' ACPI HAL. This option will try to figure out how many timer interrupts were not processed by the Windows guest and will re-inject them.

```
-icount [shift=N|auto][,align=on|off][,sleep=on|off]
[,rr=record|replay,rrfile=filename[,rrsnapshot=snapshot]]
```



Enable virtual instruction counter. The virtual cpu will execute one instruction every  $2^N$  ns of virtual time. If `auto` is specified then the virtual cpu speed will be automatically adjusted to keep virtual time within a few seconds of real time.

Note that while this option can give deterministic behavior, it does not provide cycle accurate emulation. Modern CPUs contain superscalar out of order cores with complex cache hierarchies. The number of instructions executed often has little or no correlation with actual performance.

When the virtual cpu is sleeping, the virtual time will advance at default speed unless `sleep=on` is specified. With `sleep=on`, the virtual time will jump to the next timer deadline instantly whenever the virtual cpu goes to sleep mode and will not advance if no timer is enabled. This behavior gives deterministic execution times from the guest point of view. The default ifcount is enabled is `sleep=off`. `sleep=on` cannot be used together with either `shift=auto` or `align=on`.

`align=on` will activate the delay algorithm which will try to synchronise the host clock and the virtual clock. The goal is to have a guest running at the real frequency imposed by the shift option. Whenever the guest clock is behind the host clock and if `align=on` is specified then we print a message to the user to inform about the delay. Currently this option does not work when `shift` is `auto`. Note: The sync algorithm will work for those shift values for which the guest clock runs ahead of the host clock. Typically this happens when the shift value is high (how high depends on the host machine). The default if `icount` is enabled is `align=off`.

When the `rr` option is specified deterministic record/replay is enabled. The `rrfile=` option must also be provided to specify the path to the replay log. In record mode data is written to this file, and in replay mode it is read back. If the `rrsnapshot` option is given then it specifies a VM snapshot name. In record mode, a new VM snapshot with the given name is created at the start of execution recording. In replay mode this option specifies the snapshot name used to load the initial VM state.

```
-watchdog-action action
```

The action controls what QEMU will do when the watchdog timer expires. The default is `reset` (forcefully reset the guest). Other possible actions are: `shutdown` (attempt to gracefully shutdown the guest), `poweroff` (forcefully poweroff the guest), `inject-nmi` (inject a NMI into the guest), `pause` (pause the guest), `debug` (print a debug message and continue), or `none` (do nothing).

Note that the shutdown action requires that the guest responds to ACPI signals, which it may not be able to do in the sort of situations where the watchdog would have expired, and thus `-watchdog-action shutdown` is not recommended for production use.

Examples:

```
-device i6300esb -watchdog-action pause
```

```
-echr numeric_ascii_value
```

Change the escape character used for switching to the monitor when using monitor and serial sharing. The default is 0x01 when using the -nographic option. 0x01 is equal to pressing Control-a. You can select a different character from the ascii control keys where 1 through 26 map to Control-a through Control-z. For instance you could use the either of the following to change the escape character to Control-t.

```
-echr 0x14; -echr 20
```

```
-incoming tcp:[host]:port[,to=maxport][,ipv4=on|off][,ipv6=on|off]
```

```
-incoming rdma:host:port[,ipv4=on|off][,ipv6=on|off]
```

Prepare for incoming migration, listen on a given tcp port.

```
-incoming unix:socketpath
```

Prepare for incoming migration, listen on a given unix socket.

```
-incoming fd:fd
```

Accept incoming migration from a given file descriptor.

```
-incoming file:filename[,offset=offset]
```

Accept incoming migration from a given file starting at offset. offset allows the common size suffixes, or a 0x prefix, but not both.

```
-incoming exec:cmdline
```

Accept incoming migration as an output from specified external command..

```
-incoming defer
```

Wait for the URI to be specified via `migrate_incoming`. The monitor can be used to change settings (such as migration parameters) prior to issuing the `migrate_incoming` to allow the migration to begin.

```
-only-migratable
```

Only allow migratable devices. Devices will not be allowed to enter an unmigratable state.

**-nodefaults**

Don't create default devices. Normally, QEMU sets the default devices like serial port, parallel port, virtual console, monitor device, VGA adapter, floppy and CD-ROM drive and others. The `-nodefaults` option will disable all those default devices.

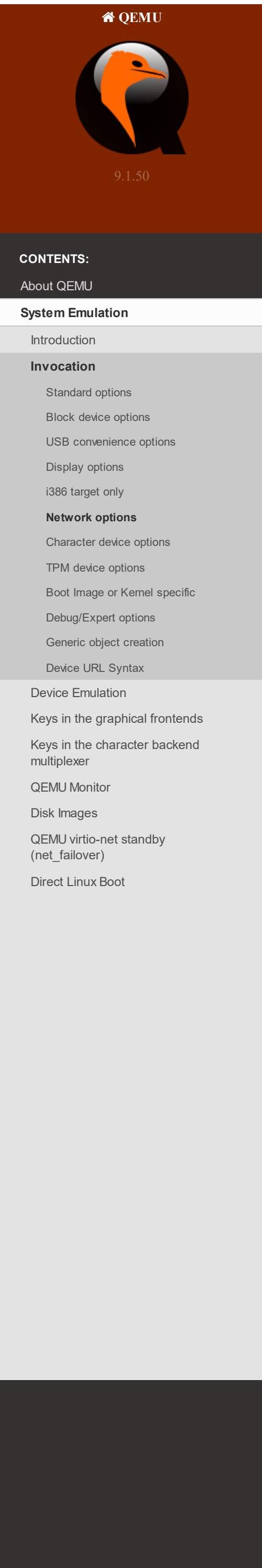
```
-runas user
```

Immediately before starting guest execution, drop root privileges, switching to the specified user. This option is deprecated, use `-run-with user=...` instead.

```
-prom-env variable=value
```

Set OpenBIOS nvram variable to given value (PPC, SPARC only).

```
qemu-system-sparc -prom-env 'auto-boot?=false' \
  -prom-env 'boot-device=sd(0,2,0):d' -prom-env 'boot-args=linux single'
```



```
qemu-system-ppc -prom-env 'auto-boot?=false' \
-prom-env 'boot-device=hd:2,\yaboot' \
-prom-env 'boot-args=conf=hd:2,\yaboot.conf'
```

```
-semihosting
```

Enable **Semihosting** mode (ARM, M68K, Xtensa, MIPS, RISC-V only).

### Warning

Note that this allows guest direct access to the host filesystem, so should only be used with a trusted guest OS.

See the `-semihosting-config` option documentation for further information about the facilities this enables.

```
-semihosting-config [enable=on|off][,target=native|gdb|auto][,chardev=id][,userspace=on|off]
[,arg=str[,...]]
```

Enable and configure [Semihosting](#) (ARM, M68K, Xtensa, MIPS, RISC-V only).

### ⚠ Warning

Note that this allows guest direct access to the host filesystem, so should only be used with a trusted guest OS.

```
target=native|gdb|auto
```

Defines where the semihosting calls will be addressed, to QEMU (native) or to GDB (gdb). The default is auto, which means gdb during debug sessions and native otherwise.

```
chardev=str1
```

Send the output to a chardev backend output for native or auto output when not in gdb

**userspace=on|off**

Allows code running in guest userspace to access the semihosting interface. The default is that only privileged guest code can make semihosting calls. Note that setting `userspace=on` should only be used if all guest code is trusted (for example, in bare-metal test case code).

arg=str1,arg=str2,...

Allows the user to pass input arguments, and can be used multiple times to build up a list. The old-style `-kernel/-append` method of passing a command line is still supported for backward compatibility. If both the `--semihosting-config arg` and the `-kernel/-append` are specified, the former is passed to semihosting as it always takes precedence.

**-old-param**

Old param mode (ARM only).

```
-sandbox arg[,obsolete=string][,elevateprivileges=string][,spawn=string][,resourcecontrol=string]
```

Enable Seccomp mode 2 system call filter. 'on' will enable syscall filtering and 'off' will disable it. The default is 'off'.

```
obsolete=string
```

### Enable Obsolete system calls

```
elevateprivileges=string
```

## Disable set\*uid|gid system calls

spawn=string

Disable \*fork and execve

```
resourcecontrol=string
```

## Disable process affinity and scheduler priority

```
-readconfig file
```

Read device configuration from file. This approach is useful when you want to spawn QEMU process with many command line options but you don't want to exceed the command line character limit.

```
-no-user-config
```

The `-no-user-config` option makes QEMU not load any of the user-provided config files on `sysconfdir`.

```
-trace [[enable=]pattern][,events=file][,file=file]
```

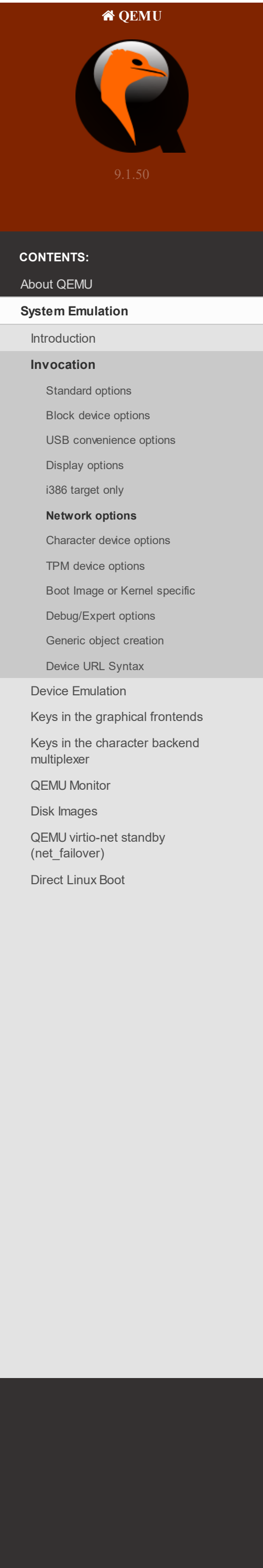
Specify tracing options.

```
[enable=]PATTERN
```

Immediately enable events matching *PATTERN* (either event name or a globbing pattern). This option is only available if QEMU has been compiled with the `simple`, `log` or `ftrace` tracing backend. To specify multiple events or patterns, specify the `-trace` option multiple times.

Use **-trace help** to print a list of names of trace points.

```
events=FILE
```



Immediately enable events listed in *FILE*. The file must contain one event name (as listed in the `trace-events-all` file) per line; globbing patterns are accepted too. This option is only available if QEMU has been compiled with the `simple`, `log` or `ftrace` tracing backend.

```
file=FILE
```

Log output traces to *FILE*. This option is only available if QEMU has been compiled with the simple tracing backend.

```
-plugin file=file[,argname=argvalue]
```

Load a plugin.

```
file=file
```

Load the given plugin from a shared library file.

```
argname=argvalue
```

Argument passed to the plugin. (Can be given multiple times.)

```
-run-with [async-teardown=on|off][,chroot=dir][user=username|uid:gid]
```

Set QEMU process lifecycle options.

async-teardown=on enables asynchronous teardown. A new process called “cleanup/<QEMU\_PID>” will be created at startup sharing the address space with the main QEMU process, using clone. It will wait for the main QEMU process to terminate completely, and then exit. This allows QEMU to terminate very quickly even if the guest was huge, leaving the teardown of the address space to the cleanup process. Since the cleanup process shares the same cgroups as the main QEMU process, accounting is performed correctly. This only works if the cleanup process is not forcefully killed with SIGKILL before the main QEMU process has terminated completely.

`chroot=dir` can be used for doing a chroot to the specified directory immediately before starting the guest execution. This is especially useful in combination with `-runas`.

`user=username` or `user=uid:gid` can be used to drop root privileges before starting guest execution. QEMU will use the `setuid` and `setgid` system calls to switch to the specified identity. Note that the `user=username` syntax will also apply the full set of supplementary groups for the user, whereas the `user=uid:gid` will use only the `gid` group.

```
-msg [timestamp[=on|off]][,guest-name[=on|off]]
```

### Control error message format.

```
timestamp=on|off
```

Prefix messages with a timestamp. Default is off.

guest-name=on | off

Prefix messages with guest name but only if -name guest option is set otherwise the option is ignored. Default is off.

```
-dump-vmstate file
```

Dump json-encoded vmstate information for current machine type to file in file

```
-enable-sync-profile
```

Enable synchronization profiling.

**-perfmap**

Generate a map file for Linux perf tools that will allow basic profiling information to be broken down into basic blocks.

```
-jitdump
```

Generate a dump file for Linux perf tools that maps basic blocks to symbol names, line numbers and JITted code.

## Generic object creation

```
-object typename[,prop1=value1,...]
```

Create a new object of type `typename` setting properties in the order they are specified. Note that the 'id' property must be set. These objects are placed in the '/objects' path.

```
-object memory-backend=file,id=id,size=size,mem-path=dir,share=on|off,discard-  
data=on|off,merge=on|off,dump=on|off,prealloc=on|off,host-nodes=host-  
nodes,policy=default|preferred|bind|interleave,align=align,offset=offset,readonly=on|off,rom=on|off|auto
```

Creates a memory file backend object, which can be used to back the guest RAM with huge pages.

The `id` parameter is a unique ID that will be used to reference this memory region in other parameters, e.g. `-numa`, `-device` `nvdimm`, etc.

The size option provides the size of the memory region, and accepts common suffixes, e.g. 500M.

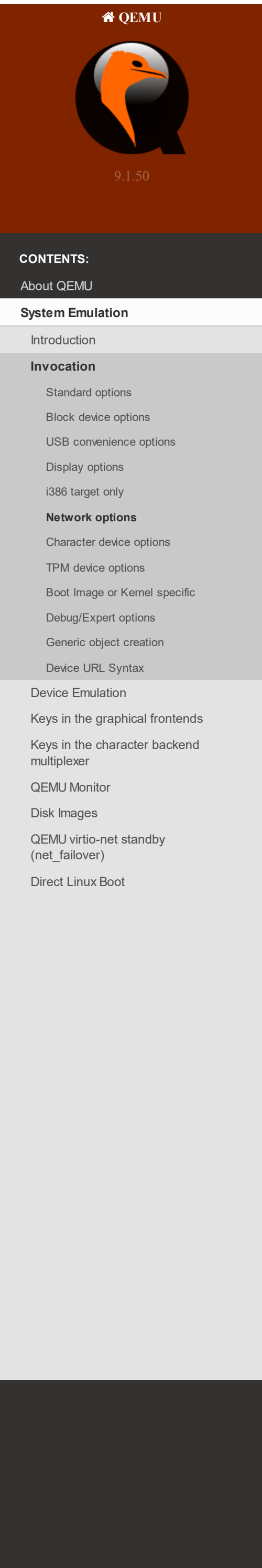
The mem-path provides the path to either a shared memory or huge page filesystem mount.

The `share` boolean option determines whether the memory region is marked as private to QEMU, or shared. The latter allows a co-operating external process to access the QEMU memory region.

Setting `share=on` might affect the ability to configure NUMA bindings for the memory backend under some circumstances, see [Documentation/vm/numa\\_memory\\_policy.txt](#) on the Linux kernel source tree for additional details.







Please refer to `memory-backend-file` for a description of the options.

The `share` boolean option is on by default with `shm`. Setting it to `off` will cause a failure during allocation because it is not supported by this backend.

```
-object iommufd,id=id[,fd=fd]
```

Creates an iommu backend which allows control of DMA mapping through the `/dev/iommu` device.

The `id` parameter is a unique ID which frontends (such as `vfi-pci` or `vdpa`) will use to connect with the `iommufd` backend.

The `fd` parameter is an optional pre-opened file descriptor resulting from `/dev/iommu` opening. Usually the `iommufd` is shared across all subsystems, bringing the benefit of centralized reference counting.

```
-object rng-builtin,id=id
```

Creates a random number generator backend which obtains entropy from QEMU builtin functions. The `id` parameter is a unique ID that will be used to reference this entropy backend from the `virtio-rng` device. By default, the `virtio-rng` device uses this RNG backend.

```
-object rng-random,id=id,filename=/dev/random
```

Creates a random number generator backend which obtains entropy from a device on the host. The `id` parameter is a unique ID that will be used to reference this entropy backend from the `virtio-rng` device. The `filename` parameter specifies which file to obtain entropy from and if omitted defaults to `/dev/urandom`.

```
-object rng-egd,id=id,chardev=chardev
```

Creates a random number generator backend which obtains entropy from an external daemon running on the host. The `id` parameter is a unique ID that will be used to reference this entropy backend from the `virtio-rng` device. The `chardev` parameter is the unique ID of a character device backend that provides the connection to the RNG daemon.

```
-object tls-creds-anon,id=id,endpoint=endpoint,dir=/path/to/cred/dir,verify-peer=on|off
```

Creates a TLS anonymous credentials object, which can be used to provide TLS support on network backends. The `id` parameter is a unique ID which network backends will use to access the credentials. The `endpoint` is either `server` or `client` depending on whether the QEMU network backend that uses the credentials will be acting as a client or as a server. If `verify-peer` is enabled (the default) then once the handshake is completed, the peer credentials will be verified, though this is a no-op for anonymous credentials.

The `dir` parameter tells QEMU where to find the credential files. For server endpoints, this directory may contain a file `dh-params.pem` providing diffie-hellman parameters to use for the TLS server. If the file is missing, QEMU will generate a set of DH parameters at startup. This is a computationally expensive operation that consumes random pool entropy, so it is recommended that a persistent set of parameters be generated upfront and saved.

```
-object tls-creds-psk,id=id,endpoint=endpoint,dir=/path/to/keys/dir[,username=username]
```

Creates a TLS Pre-Shared Keys (PSK) credentials object, which can be used to provide TLS support on network backends. The `id` parameter is a unique ID which network backends will use to access the credentials. The `endpoint` is either `server` or `client` depending on whether the QEMU network backend that uses the credentials will be acting as a client or as a server. For clients only, `username` is the username which will be sent to the server. If omitted it defaults to “qemu”.

The `dir` parameter tells QEMU where to find the keys file. It is called “`dir/keys.psk`” and contains “`username:key`” pairs. This file can most easily be created using the GnuTLS `psktool` program.

For server endpoints, `dir` may also contain a file `dh-params.pem` providing diffie-hellman parameters to use for the TLS server. If the file is missing, QEMU will generate a set of DH parameters at startup. This is a computationally expensive operation that consumes random pool entropy, so it is recommended that a persistent set of parameters be generated up front and saved.

```
-object tls-creds-x509,id=id,endpoint=endpoint,dir=/path/to/cred/dir,priority=priority,verify-peer=on|off,passwordid=id
```

Creates a TLS anonymous credentials object, which can be used to provide TLS support on network backends. The `id` parameter is a unique ID which network backends will use to access the credentials. The `endpoint` is either `server` or `client` depending on whether the QEMU network backend that uses the credentials will be acting as a client or as a server. If `verify-peer` is enabled (the default) then once the handshake is completed, the peer credentials will be verified. With x509 certificates, this implies that the clients must be provided with valid client certificates too.

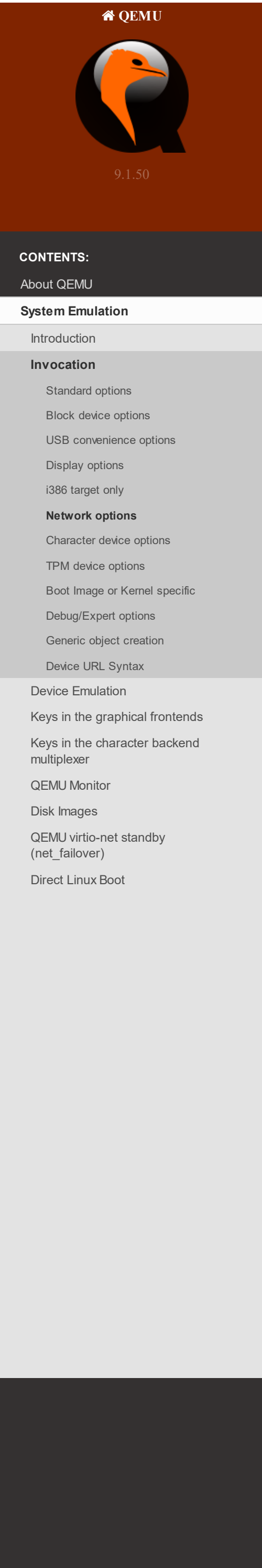
The `dir` parameter tells QEMU where to find the credential files. For server endpoints, this directory may contain a file `dh-params.pem` providing diffie-hellman parameters to use for the TLS server. If the file is missing, QEMU will generate a set of DH parameters at startup. This is a computationally expensive operation that consumes random pool entropy, so it is recommended that a persistent set of parameters be generated upfront and saved.

For x509 certificate credentials the directory will contain further files providing the x509 certificates. The certificates must be stored in PEM format, in filenames `ca-cert.pem`, `ca-crl.pem` (optional), `server-cert.pem` (only servers), `server-key.pem` (only servers), `client-cert.pem` (only clients), and `client-key.pem` (only clients).

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 priority parameter allows to override the global default priority used by `gnutls`. This can be useful if the system administrator needs to use a weaker set of crypto priorities for QEMU without potentially forcing the weakness onto all applications. Or conversely if one wants a stronger default for QEMU than for all other applications, they can do this through this parameter. Its format is a `gnutls` priority string as described at [https://gnutls.org/manual/html\\_node/Priority-Strings.html](https://gnutls.org/manual/html_node/Priority-Strings.html).

```
-object tls-cipher-suites,id=id,priority=priority
```



Creates a TLS cipher suites object, which can be used to control the TLS cipher/protocol algorithms that applications are permitted to use.

The `id` parameter is a unique ID which frontends will use to access the ordered list of permitted TLS cipher suites from the host.

The `priority` parameter allows to override the global default priority used by `gnutls`. This can be useful if the system administrator needs to use a weaker set of crypto priorities for QEMU without potentially forcing the weakness onto all applications. Or conversely if one wants a stronger default for QEMU than for all other applications, they can do this through this parameter. Its format is a `gnutls` priority string as described at [https://gnutls.org/manual/html\\_node/Priority-Strings.html](https://gnutls.org/manual/html_node/Priority-Strings.html).

An example of use of this object is to control UEFI HTTPS Boot. The `tls-cipher-suites` object exposes the ordered list of permitted TLS cipher suites from the host side to the guest firmware, via `fw_cfg`. The list is represented as an array of `IANA_TLS_CIPHER` objects. The firmware uses the `IANA_TLS_CIPHER` array for configuring guest-side TLS.

In the following example, the priority at which the host-side policy is retrieved is given by the `priority` property. Given that QEMU uses GNUTLS, `priority=@SYSTEM` may be used to refer to `/etc/crypto-policies/back-ends/gnutls.config`.

```
# qemu-system-x86_64 \
  -object tls-cipher-suites,id=mysuite0,priority=@SYSTEM \
  -fw_cfg name=etc/edk2/https/ciphers,gen_id=mysuite0
```

```
-object filter-buffer,id=id,netdev=netdevid,interval=t[,queue=all|rx|tx][,status=on|off]
[,position=head|tail|id=<id>][,insert=behind|before]
```

Interval `t` can't be 0, this filter batches the packet delivery: all packets arriving in a given interval on netdev `netdevid` are delayed until the end of the interval. Interval is in microseconds. `status` is optional that indicate whether the netfilter is on (enabled) or off (disabled), the default status for netfilter will be 'on'.

queue all|rx|tx is an option that can be applied to any netfilter.

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.

position `head|tail|id=<id>` is an option to specify where the filter should be inserted in the filter list. It can be applied to any netfilter.

head: the filter is inserted at the head of the filter list, before any existing filters.

**tail:** the filter is inserted at the tail of the filter list, behind any existing filters (default).

**id=<id>**: the filter is inserted before or behind the filter specified by <id>, see the insert option below.

insert behind|before is an option to specify where to insert the new filter relative to the one specified with position=id=<id>. It can be applied to any netfilter.

before: insert before the specified filter.

behind: insert behind the specified filter (default).

```
-object filter-mirror,id=id,netdev=netdevid,outdev=chardevid,queue=all|rx|tx[,vnet_hdr_support]
[,position=head|tail|id=<id>][,insert=behind|before]
```

filter-mirror on netdev netdevid,mirror net packet to chardevchardevid, if it has the vnet\_hdr\_support flag, filter-mirror will mirror packet with vnet\_hdr len.

```
-object filter-
redirector,id=id,netdev=netdev,indv=chardev,outdev=chardev,queue=all|rx|tx[,vnet_hdr_support]
[,position=head|tail|id=<id>][,insert=behind|before]
```

filter-redirector on netdev netdevid, redirect filter's net packet to chardev chardevid, and redirect indev's packet to filter. if it has the vnet\_hdr\_support flag, filter-redirector will redirect packet with vnet\_hdr\_len. Create a filter-redirector we need to differ outdev id from indev id, id can not be the same. we can just use indev or outdev, but at least one of indev or outdev need to be specified.

```
-object filter-rewriter,id=id,netdev=netdev,queue=all|rx|tx,[vnet_hdr_support]
[,position=head|tail|id=<id>][,insert=behind|before]
```

Filter-rewriter is a part of COLO project. It will rewrite tcp packet to secondary from primary to keep secondary tcp connection, and rewrite tcp packet to primary from secondary make tcp packet can be handled by client. if it has the `vnet_hdr_support` flag, we can parse packet with vnet header.

```
usage: colo secondary: -object filter-redirector,id=f1,netdev=hn0,queue=tx,indev=red0 -object filter-
redirector,id=f2,netdev=hn0,queue=rx,outdev=red1 -object filter-rewriter,id=rew0,netdev=hn0,queue=all
```

```
-object filter-dump,id=id,netdev=dev[,file=filename][,maxlen=len][,position=head|tail|id=<id>]
[,insert=behind|before]
```

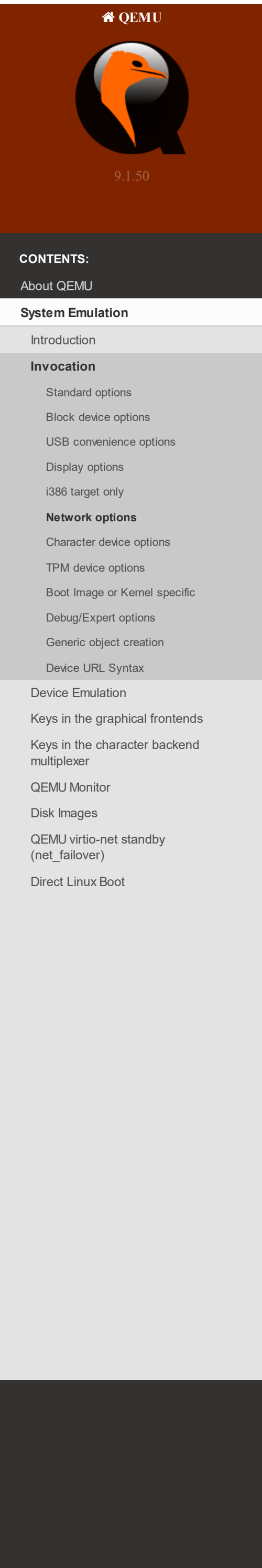
Dump the network traffic on netdev dev to the file specified by filename. At most len bytes (64k by default) per packet are stored. The file format is libpcap, so it can be analyzed with tools such as tcpdump or Wireshark.

```
-object colo-
compare,id=id,primary_in=chardev,secondary_in=chardev,outdev=chardev,iotthread=id[,vnet_hdr_support]
[,notify_dev=id][,compare_timeout=@var{ms}][,expired_scan_cycle=@var{ms}][,max_queue_size=@var{size}]
```

Colo-compare gets packet from `primary_in` chardev and `secondary_in`, then compare whether the payload of primary packet and secondary packet are the same. If same, it will output primary packet to `out_dev`, else it will notify COLO-framework to do







Defines a secret to store a password, encryption key, or some other sensitive data. The sensitive data can either be passed directly via the data parameter, or indirectly via the file parameter. Using the data parameter is insecure unless the sensitive data is encrypted.

The sensitive data can be provided in raw format (the default), or base64. When encoded as JSON, the raw format only supports valid UTF-8 characters, so base64 is recommended for sending binary data. QEMU will convert from which ever format is provided to the format it needs internally. eg, an RBD password can be provided in raw format, even though it will be base64 encoded when passed onto the RBD sever.

For added protection, it is possible to encrypt the data associated with a secret using the AES-256-CBC cipher. Use of encryption is indicated by providing the `keyid` and `iv` parameters. The `keyid` parameter provides the ID of a previously defined secret that contains the AES-256 decryption key. This key should be 32-bytes long and be base64 encoded. The `iv` parameter provides the random initialization vector used for encryption of this particular secret and should be a base64 encrypted string of the 16-byte IV.

The simplest (insecure) usage is to provide the secret inline

```
# qemu-system-x86_64 -object secret,id=sec0,data=letmein,format=raw
```

The simplest secure usage is to provide the secret via a file

```
# printf "letmein"> mypasswd.txt # QEMU_SYSTEM_MACRO -object secret,id=sec0,file=mypasswd.txt,format=raw
```

For greater security, AES-256-CBC should be used. To illustrate usage, consider the openssl command line tool which can encrypt the data. Note that when encrypting, the plaintext must be padded to the cipher block size (32 bytes) using the standard PKCS#5/6 compatible padding algorithm.

First a master key needs to be created in base64 encoding:

```
# openssl rand -base64 32 > key.b64
# KEY=$(base64 -d key.b64 | hexdump -v -e '/1 "%02X"')
```

Each secret to be encrypted needs to have a random initialization vector generated. These do not need to be kept secret

```
# openssl rand -base64 16 > iv.b64
# IV=$(base64 -d iv.b64 | hexdump -v -e '/1 "%02X"')
```

The secret to be defined can now be encrypted, in this case we're telling openssl to base64 encode the result, but it could be left as raw bytes if desired.

```
# SECRET=$(printf "letmein" |
    openssl enc -aes-256-cbc -a -K $KEY -iv $IV)
```

When launching QEMU, create a master secret pointing to `key.b64` and specify that to be used to decrypt the user password. Pass the contents of `iv.b64` to the second secret

```
# qemu-system-x86_64 \
  -object secret,id=secmaster0,format=base64,file=key.b64 \
  -object secret,id=sec0,keyid=secmaster0,format=base64,\
    data=$SECRET,iv=$(iv.b64)
```

```
-object sev-guest,id=id,cbitpos=cbitpos,reduced-phys-bits=val,[sev-  
device=string,policy=policy,handle=handle,dh-cert-file=file,session-file=file,kernel-hashes=on|off]
```

Create a Secure Encrypted Virtualization (SEV) guest object, which can be used to provide the guest memory encryption support on AMD processors.

When memory encryption is enabled, one of the physical address bit (aka the C-bit) is utilized to mark if a memory page is protected. The `cbtpos` is used to provide the C-bit position. The C-bit position is Host family dependent hence user must provide this value. On EPYC, the value should be 47.

When memory encryption is enabled, we loose certain bits in physical address space. The `reduced-phys-bits` is used to provide the number of bits we loose in physical address space. Similar to C-bit, the value is Host family dependent. On EPYC, a guest will lose a maximum of 1 bit, so the value should be 1.

The sev-device provides the device file to use for communicating with the SEV firmware running inside AMD Secure Processor. The default device is `/dev/sev`. If hardware supports memory encryption then `/dev/sev` devices are created by CCP driver.

The policy provides the guest policy to be enforced by the SEV firmware and restrict what configuration and operational commands can be performed on this guest by the hypervisor. The policy should be provided by the guest owner and is bound to the guest and cannot be changed throughout the lifetime of the guest. The default is 0.

If guest policy allows sharing the key with another SEV guest then handle can be use to provide handle of the guest from which to share the key.

The `dh-cert-file` and `session-file` provides the guest owner's Public Diffie-Hillman key defined in SEV spec. The PDH and session parameters are used for establishing a cryptographic session with the guest owner to negotiate keys used for attestation. The file must be encoded in base64.

The `kernel-hashes` adds the hashes of given `kernel/initrd/ cmdline` to a designated guest firmware page for measured Linux boot with `-kernel`. The default is off. (Since 6.2)

e.g to launch a SEV guest











## CONTENTS:

## About QEMU

## System Emulation

## Introduction

## Invocation

## Standard options

## Block device options

## USB convenience options

## Display options

i386 target only

## Network options

## Character device options

## TPM device options

### Boot Image or Kernel specific

### Debug/Expert options

## Generic object creation

## Device URL Syntax

## Device Emulation

## Keys in the graphical frontends

## Keys in the character backend multiplexer

QEMU Monitor

## Disk Images

QEMU virtio-net standby  
(net\_failover)

## Direct Linux Boot

cookie

Send this cookie (it can also be a list of cookies separated by ';') with each outgoing request. Only supported when using protocols such as HTTP which support cookies, otherwise ignored.

timeout

Set the timeout in seconds of the CURL connection. This timeout is the time that CURL waits for a response from the remote server to get the size of the image to be downloaded. If not set, the default timeout of 5 seconds is used.

Note that when passing options to qemu explicitly, driver is the value of <protocol>.

### Example: boot from a remote Fedora 20 live ISO image

```
qemu-system-x86_64 --drive media=cdrom,file=https://archives.fedoraproject.org/pub/archive/fedora/linux/releases/20/LiveOS/x86_64/iso/Fedora-Linux-20-20200101-1.x86_64-1.iso
qemu-system-x86_64 --drive media=cdrom,file.driver=http,file.url=http://archives.fedoraproject.org/pub/fedora/linux/releases/20/LiveOS/x86_64/iso/Fedora-Linux-20-20200101-1.x86_64-1.iso
```

Example: boot from a remote Fedora 20 cloud image using a local overlay for writes, copy-on-read, and a readahead of 64k

```
qemu-img create -f qcow2 -o backing_file='json:{"file.driver":"http",, "file.url":"http://archives.fedoraproject.org
qemu-system-x86_64 -drive file=/tmp/Fedora-x86_64-20-20131211.1-sda.qcow2,copy-on-read=on
```

Example: boot from an image stored on a VMware vSphere server with a self-signed certificate using a local overlay for writes, a readahead of 64k and a timeout of 10 seconds.

```
qemu-img create -f qcow2 -o backing_file='json:{"file.driver":"https",, "file.url":"https://user:password@vsphere.example.com/vmfs/volume1/centos.qcow2"}' centos.qcow2
qemu-system-x86_64 -drive file=/tmp/test.qcow2
```

[← Previous](#)

Next ➔

© Copyright 2024, The QEMU Project Developers.

Built with [Sphinx](#) using a [theme](#) provided by [Read the Docs](#).

This documentation is for QEMU version 9.1.50.

QEMU and this manual are released under the GNU General Public License, version 2.