

Libvirt

Generated by Doxygen 1.8.3.1

Mon Jul 29 2013 16:35:26



# Contents

<b>1</b>	<b>Data Structure Index</b>	<b>1</b>
1.1	Data Structures . . . . .	1
<b>2</b>	<b>File Index</b>	<b>5</b>
2.1	File List . . . . .	5
<b>3</b>	<b>Data Structure Documentation</b>	<b>7</b>
3.1	<a href="#">_gemuDomainPCIAddressSet Struct Reference</a> . . . . .	7
3.1.1	Field Documentation . . . . .	7
3.1.1.1	nextslot . . . . .	7
3.1.1.2	used . . . . .	7
3.2	<a href="#">_virBlkioDeviceWeight Struct Reference</a> . . . . .	7
3.2.1	Field Documentation . . . . .	7
3.2.1.1	path . . . . .	7
3.2.1.2	weight . . . . .	7
3.3	<a href="#">_virConnectAuth Struct Reference</a> . . . . .	8
3.3.1	Field Documentation . . . . .	8
3.3.1.1	cb . . . . .	8
3.3.1.2	cbdata . . . . .	8
3.3.1.3	credtype . . . . .	8
3.3.1.4	ncredtype . . . . .	8
3.4	<a href="#">_virConnectCredential Struct Reference</a> . . . . .	8
3.4.1	Field Documentation . . . . .	8
3.4.1.1	challenge . . . . .	8
3.4.1.2	defresult . . . . .	8
3.4.1.3	prompt . . . . .	8
3.4.1.4	result . . . . .	8
3.4.1.5	resultlen . . . . .	8
3.4.1.6	type . . . . .	9
3.5	<a href="#">_virDeviceMonitor Struct Reference</a> . . . . .	9
3.5.1	Detailed Description . . . . .	9
3.5.2	Field Documentation . . . . .	9

3.5.2.1	close	9
3.5.2.2	deviceCreateXML	9
3.5.2.3	deviceDestroy	9
3.5.2.4	deviceGetParent	9
3.5.2.5	deviceGetXMLDesc	9
3.5.2.6	deviceListCaps	9
3.5.2.7	deviceLookupByName	9
3.5.2.8	deviceNumOfCaps	9
3.5.2.9	listAllNodeDevices	9
3.5.2.10	listDevices	10
3.5.2.11	name	10
3.5.2.12	numOfDevices	10
3.5.2.13	open	10
3.6	<a href="#">_virDomainActualNetDef Struct Reference</a>	10
3.6.1	Field Documentation	10
3.6.1.1	bandwidth	10
3.6.1.2	bridge	10
3.6.1.3	brname	10
3.6.1.4	<b>POL New</b> brtype	10
3.6.1.5	data	10
3.6.1.6	def	10
3.6.1.7	direct	10
3.6.1.8	hostdev	11
3.6.1.9	linkdev	11
3.6.1.10	mode	11
3.6.1.11	type	11
3.6.1.12	virtPortProfile	11
3.6.1.13	vlan	11
3.7	<a href="#">_virDomainBIOSDef Struct Reference</a>	11
3.7.1	Field Documentation	11
3.7.1.1	rt_delay	11
3.7.1.2	rt_set	11
3.7.1.3	useserial	11
3.8	<a href="#">_virDomainBlockInfo Struct Reference</a>	11
3.8.1	Field Documentation	12
3.8.1.1	allocation	12
3.8.1.2	capacity	12
3.8.1.3	physical	12
3.9	<a href="#">_virDomainBlockIoTuneInfo Struct Reference</a>	12
3.9.1	Field Documentation	12

3.9.1.1	read_bytes_sec	12
3.9.1.2	read_iops_sec	12
3.9.1.3	total_bytes_sec	12
3.9.1.4	total_iops_sec	12
3.9.1.5	write_bytes_sec	12
3.9.1.6	write_iops_sec	12
3.10	_virDomainBlockJobInfo Struct Reference	12
3.10.1	Field Documentation	13
3.10.1.1	bandwidth	13
3.10.1.2	cur	13
3.10.1.3	end	13
3.10.1.4	type	13
3.11	_virDomainBlockStats Struct Reference	13
3.11.1	Field Documentation	13
3.11.1.1	errs	13
3.11.1.2	rd_bytes	13
3.11.1.3	rd_req	13
3.11.1.4	wr_bytes	13
3.11.1.5	wr_req	13
3.12	_virDomainChrDef Struct Reference	13
3.12.1	Field Documentation	14
3.12.1.1	addr	14
3.12.1.2	deviceType	14
3.12.1.3	info	14
3.12.1.4	name	14
3.12.1.5	nseclabels	14
3.12.1.6	port	14
3.12.1.7	seclabels	14
3.12.1.8	source	14
3.12.1.9	target	14
3.12.1.10	targetType	14
3.13	_virDomainChrSourceDef Struct Reference	14
3.13.1	Field Documentation	15
3.13.1.1	bindHost	15
3.13.1.2	bindService	15
3.13.1.3	connectHost	15
3.13.1.4	connectService	15
3.13.1.5	data	15
3.13.1.6	file	15
3.13.1.7	host	15

3.13.1.8	listen	15
3.13.1.9	nix	15
3.13.1.10	path	15
3.13.1.11	protocol	15
3.13.1.12	service	15
3.13.1.13	spicevmc	15
3.13.1.14	tcp	15
3.13.1.15	type	15
3.13.1.16	udp	15
3.14	_virDomainClockDef Struct Reference	16
3.14.1	Field Documentation	16
3.14.1.1	adjustment	16
3.14.1.2	basis	16
3.14.1.3	data	16
3.14.1.4	ntimers	16
3.14.1.5	offset	16
3.14.1.6	timers	16
3.14.1.7	timezone	16
3.14.1.8	utc_reset	16
3.14.1.9	variable	16
3.15	_virDomainControllInfo Struct Reference	16
3.15.1	Field Documentation	17
3.15.1.1	details	17
3.15.1.2	state	17
3.15.1.3	stateTime	17
3.16	_virDomainControllerDef Struct Reference	17
3.16.1	Field Documentation	17
3.16.1.1	idx	17
3.16.1.2	info	17
3.16.1.3	model	17
3.16.1.4	opts	17
3.16.1.5	type	17
3.16.1.6	vioserial	17
3.17	_virDomainDef Struct Reference	17
3.17.1	Field Documentation	19
3.17.1.1	apic_eoi	19
3.17.1.2	blkio	19
3.17.1.3	channels	19
3.17.1.4	clock	19
3.17.1.5	consoles	19

3.17.1.6 controllers	20
3.17.1.7 cpu	20
3.17.1.8 cpumask	20
3.17.1.9 cputune	20
3.17.1.10 cur_balloon	20
3.17.1.11 description	20
3.17.1.12 devices	20
3.17.1.13 disks	20
3.17.1.14 dump_core	20
3.17.1.15 emulator	20
3.17.1.16 emulator_period	20
3.17.1.17 emulator_quota	20
3.17.1.18 emulatorpin	20
3.17.1.19 features	20
3.17.1.20 fss	20
3.17.1.21 graphics	20
3.17.1.22 hard_limit	20
3.17.1.23 hostdevs	20
3.17.1.24 hubs	20
3.17.1.25 hugepage_backed	20
3.17.1.26 id	20
3.17.1.27 inputs	20
3.17.1.28 leases	20
3.17.1.29 max_balloon	20
3.17.1.30 maxvcpus	20
3.17.1.31 mem	20
3.17.1.32 memballoon	20
3.17.1.33 metadata	20
3.17.1.34 min_guarantee	21
3.17.1.35 name	21
3.17.1.36 namespaceData	21
3.17.1.37 nchannels	21
3.17.1.38 nconsoles	21
3.17.1.39 ncontrollers	21
3.17.1.40 ndevices	21
3.17.1.41 ndisks	21
3.17.1.42 nets	21
3.17.1.43 nfss	21
3.17.1.44 ngraphics	21
3.17.1.45 nhostdevs	21

3.17.1.46 nhubs	21
3.17.1.47 ninputs	21
3.17.1.48 nleases	21
3.17.1.49 nnets	21
3.17.1.50 nparallels	21
3.17.1.51 nredirdevs	21
3.17.1.52 ns	21
3.17.1.53 nseclabels	21
3.17.1.54 nserials	21
3.17.1.55 nsmartcards	21
3.17.1.56 nsounds	21
3.17.1.57 numatune	21
3.17.1.58 nvcpupin	21
3.17.1.59 nvideos	21
3.17.1.60 onCrash	21
3.17.1.61 onPoweroff	21
3.17.1.62 onReboot	22
3.17.1.63 os	22
3.17.1.64 parallels	22
3.17.1.65 period	22
3.17.1.66 placement_mode	22
3.17.1.67 pm	22
3.17.1.68 quota	22
3.17.1.69 redirdevs	22
3.17.1.70 redirfilter	22
3.17.1.71 s3	22
3.17.1.72 s4	22
3.17.1.73 seclabels	22
3.17.1.74 serials	22
3.17.1.75 shares	22
3.17.1.76 smartcards	22
3.17.1.77 soft_limit	22
3.17.1.78 sounds	22
3.17.1.79 swap_hard_limit	22
3.17.1.80 sysinfo	22
3.17.1.81 title	22
3.17.1.82 uuid	22
3.17.1.83 vcpupin	22
3.17.1.84 vcpus	22
3.17.1.85 videos	22



3.17.1.86 virtType . . . . .	22
3.17.1.87 watchdog . . . . .	22
3.17.1.88 weight . . . . .	22
3.18 _virDomainDeviceCcidAddress Struct Reference . . . . .	23
3.18.1 Field Documentation . . . . .	23
3.18.1.1 controller . . . . .	23
3.18.1.2 slot . . . . .	23
3.19 _virDomainDeviceDef Struct Reference . . . . .	23
3.19.1 Field Documentation . . . . .	23
3.19.1.1 chr . . . . .	23
3.19.1.2 controller . . . . .	23
3.19.1.3 data . . . . .	24
3.19.1.4 disk . . . . .	24
3.19.1.5 fs . . . . .	24
3.19.1.6 graphics . . . . .	24
3.19.1.7 hostdev . . . . .	24
3.19.1.8 hub . . . . .	24
3.19.1.9 input . . . . .	24
3.19.1.10 lease . . . . .	24
3.19.1.11 memballoon . . . . .	24
3.19.1.12 net . . . . .	24
3.19.1.13 redirdev . . . . .	24
3.19.1.14 smartcard . . . . .	24
3.19.1.15 sound . . . . .	24
3.19.1.16 type . . . . .	24
3.19.1.17 video . . . . .	24
3.19.1.18 watchdog . . . . .	24
3.20 _virDomainDeviceDriveAddress Struct Reference . . . . .	24
3.20.1 Field Documentation . . . . .	24
3.20.1.1 bus . . . . .	24
3.20.1.2 controller . . . . .	24
3.20.1.3 target . . . . .	25
3.20.1.4 unit . . . . .	25
3.21 _virDomainDeviceInfo Struct Reference . . . . .	25
3.21.1 Field Documentation . . . . .	25
3.21.1.1 addr . . . . .	25
3.21.1.2 alias . . . . .	25
3.21.1.3 bootIndex . . . . .	25
3.21.1.4 ccid . . . . .	25
3.21.1.5 drive . . . . .	25

3.21.1.6	master	25
3.21.1.7	mastertype	25
3.21.1.8	pci	25
3.21.1.9	rombar	25
3.21.1.10	romfile	26
3.21.1.11	spaprvio	26
3.21.1.12	type	26
3.21.1.13	usb	26
3.21.1.14	usb	26
3.21.1.15	vioserial	26
3.22	_virDomainDeviceSpaprVioAddress Struct Reference	26
3.22.1	Field Documentation	26
3.22.1.1	has_reg	26
3.22.1.2	reg	26
3.23	_virDomainDeviceUSBAddress Struct Reference	26
3.23.1	Field Documentation	26
3.23.1.1	bus	26
3.23.1.2	port	26
3.24	_virDomainDeviceUSBMaster Struct Reference	27
3.24.1	Field Documentation	27
3.24.1.1	startport	27
3.25	_virDomainDeviceVirtioSerialAddress Struct Reference	27
3.25.1	Field Documentation	27
3.25.1.1	bus	27
3.25.1.2	controller	27
3.25.1.3	port	27
3.26	_virDomainDiskDef Struct Reference	27
3.26.1	Field Documentation	29
3.26.1.1	auth	29
3.26.1.2	blkdeviotune	29
3.26.1.3	blockio	29
3.26.1.4	bus	29
3.26.1.5	cachemode	29
3.26.1.6	copy_on_read	29
3.26.1.7	cylinders	29
3.26.1.8	device	29
3.26.1.9	driverName	29
3.26.1.10	driverType	29
3.26.1.11	dst	29
3.26.1.12	encryption	29

3.26.1.13 error_policy	29
3.26.1.14 event_idx	29
3.26.1.15 geometry	29
3.26.1.16 heads	29
3.26.1.17 hosts	29
3.26.1.18 info	29
3.26.1.19 ioeventfd	29
3.26.1.20 iomode	29
3.26.1.21 logical_block_size	29
3.26.1.22 mirror	29
3.26.1.23 mirrorFormat	29
3.26.1.24 mirroring	29
3.26.1.25 nhosts	29
3.26.1.26 nseclabels	29
3.26.1.27 physical_block_size	29
3.26.1.28 protocol	30
3.26.1.29 rawio	30
3.26.1.30 rawio_specified	30
3.26.1.31 readonly	30
3.26.1.32 rerror_policy	30
3.26.1.33 seclabels	30
3.26.1.34 secret	30
3.26.1.35 secretType	30
3.26.1.36 sectors	30
3.26.1.37 serial	30
3.26.1.38 shared	30
3.26.1.39 snapshot	30
3.26.1.40 src	30
3.26.1.41 startupPolicy	30
3.26.1.42 trans	30
3.26.1.43 transient	30
3.26.1.44 tray_status	30
3.26.1.45 type	30
3.26.1.46 usage	30
3.26.1.47 username	30
3.26.1.48 uuid	30
3.26.1.49 wwn	30
3.27 _virDomainDiskError Struct Reference	30
3.27.1 Field Documentation	31
3.27.1.1 disk	31

3.27.1.2	error	31
3.28	<a href="#">_virDomainDiskHostDef Struct Reference</a>	31
3.28.1	Field Documentation	31
3.28.1.1	name	31
3.28.1.2	port	31
3.29	<a href="#">_virDomainEventGraphicsAddress Struct Reference</a>	31
3.29.1	Detailed Description	31
3.29.2	Field Documentation	32
3.29.2.1	family	32
3.29.2.2	node	32
3.29.2.3	service	32
3.30	<a href="#">_virDomainEventGraphicsSubject Struct Reference</a>	32
3.30.1	Detailed Description	32
3.30.2	Field Documentation	32
3.30.2.1	identities	32
3.30.2.2	nidentity	32
3.31	<a href="#">_virDomainEventGraphicsSubjectIdentity Struct Reference</a>	32
3.31.1	Detailed Description	33
3.31.2	Field Documentation	33
3.31.2.1	name	33
3.31.2.2	type	33
3.32	<a href="#">_virDomainFSDef Struct Reference</a>	33
3.32.1	Field Documentation	33
3.32.1.1	accessmode	33
3.32.1.2	dst	33
3.32.1.3	fsdriver	33
3.32.1.4	info	33
3.32.1.5	readonly	33
3.32.1.6	space_hard_limit	33
3.32.1.7	space_soft_limit	33
3.32.1.8	src	34
3.32.1.9	type	34
3.32.1.10	usage	34
3.32.1.11	wrpolicy	34
3.33	<a href="#">_virDomainGraphicsAuthDef Struct Reference</a>	34
3.33.1	Field Documentation	34
3.33.1.1	connected	34
3.33.1.2	expires	34
3.33.1.3	passwd	34
3.33.1.4	validTo	34

3.34 <a href="#">_virDomainGraphicsDef Struct Reference</a>	34
3.34.1 <a href="#">Field Documentation</a>	35
3.34.1.1 <a href="#">auth</a>	35
3.34.1.2 <a href="#">autoport</a>	35
3.34.1.3 <a href="#">channels</a>	35
3.34.1.4 <a href="#">cypypaste</a>	35
3.34.1.5 <a href="#">data</a>	35
3.34.1.6 <a href="#">defaultMode</a>	35
3.34.1.7 <a href="#">desktop</a>	35
3.34.1.8 <a href="#">display</a>	35
3.34.1.9 <a href="#">fullscreen</a>	35
3.34.1.10 <a href="#">fullscreen</a>	36
3.34.1.11 <a href="#">image</a>	36
3.34.1.12 <a href="#">jpeg</a>	36
3.34.1.13 <a href="#">keymap</a>	36
3.34.1.14 <a href="#">listens</a>	36
3.34.1.15 <a href="#">mousemode</a>	36
3.34.1.16 <a href="#">multiUser</a>	36
3.34.1.17 <a href="#">nListens</a>	36
3.34.1.18 <a href="#">playback</a>	36
3.34.1.19 <a href="#">port</a>	36
3.34.1.20 <a href="#">rdp</a>	36
3.34.1.21 <a href="#">replaceUser</a>	36
3.34.1.22 <a href="#">sdl</a>	36
3.34.1.23 <a href="#">socket</a>	36
3.34.1.24 <a href="#">spice</a>	36
3.34.1.25 <a href="#">streaming</a>	36
3.34.1.26 <a href="#">tlsPort</a>	36
3.34.1.27 <a href="#">type</a>	36
3.34.1.28 <a href="#">vnc</a>	36
3.34.1.29 <a href="#">xauth</a>	36
3.34.1.30 <a href="#">zlib</a>	36
3.35 <a href="#">_virDomainGraphicsListenDef Struct Reference</a>	36
3.35.1 <a href="#">Field Documentation</a>	37
3.35.1.1 <a href="#">address</a>	37
3.35.1.2 <a href="#">network</a>	37
3.35.1.3 <a href="#">type</a>	37
3.36 <a href="#">_virDomainHostdevDef Struct Reference</a>	37
3.36.1 <a href="#">Field Documentation</a>	37
3.36.1.1 <a href="#">caps</a>	37

3.36.1.2	dummy	37
3.36.1.3	info	37
3.36.1.4	managed	37
3.36.1.5	mode	37
3.36.1.6	origstates	37
3.36.1.7	parent	37
3.36.1.8	source	37
3.36.1.9	subsys	37
3.37	_virDomainHostdevOrigStates Struct Reference	38
3.37.1	Field Documentation	38
3.37.1.1	pci	38
3.37.1.2	remove_slot	38
3.37.1.3	reprobe	38
3.37.1.4	states	38
3.37.1.5	unbind_from_stub	38
3.38	_virDomainHostdevSubsys Struct Reference	38
3.38.1	Field Documentation	38
3.38.1.1	bus	38
3.38.1.2	device	39
3.38.1.3	pci	39
3.38.1.4	product	39
3.38.1.5	type	39
3.38.1.6	u	39
3.38.1.7	usb	39
3.38.1.8	vendor	39
3.39	_virDomainHubDef Struct Reference	39
3.39.1	Field Documentation	39
3.39.1.1	info	39
3.39.1.2	type	39
3.40	_virDomainInfo Struct Reference	39
3.40.1	Field Documentation	40
3.40.1.1	cpuTime	40
3.40.1.2	maxMem	40
3.40.1.3	memory	40
3.40.1.4	nrVirtCpu	40
3.40.1.5	state	40
3.41	_virDomainInputDef Struct Reference	40
3.41.1	Field Documentation	40
3.41.1.1	bus	40
3.41.1.2	info	40

3.41.1.3	type	40
3.42	<a href="#">_virDomainInterfaceStats Struct Reference</a>	40
3.42.1	Field Documentation	41
3.42.1.1	rx_bytes	41
3.42.1.2	rx_drop	41
3.42.1.3	rx_errs	41
3.42.1.4	rx_packets	41
3.42.1.5	tx_bytes	41
3.42.1.6	tx_drop	41
3.42.1.7	tx_errs	41
3.42.1.8	tx_packets	41
3.43	<a href="#">_virDomainJobInfo Struct Reference</a>	41
3.43.1	Field Documentation	41
3.43.1.1	dataProcessed	41
3.43.1.2	dataRemaining	41
3.43.1.3	dataTotal	41
3.43.1.4	fileProcessed	41
3.43.1.5	fileRemaining	41
3.43.1.6	fileTotal	42
3.43.1.7	memProcessed	42
3.43.1.8	memRemaining	42
3.43.1.9	memTotal	42
3.43.1.10	timeElapsed	42
3.43.1.11	timeRemaining	42
3.43.1.12	type	42
3.44	<a href="#">_virDomainLeaseDef Struct Reference</a>	42
3.44.1	Field Documentation	42
3.44.1.1	key	42
3.44.1.2	lockspace	42
3.44.1.3	offset	42
3.44.1.4	path	42
3.45	<a href="#">_virDomainMemballoonDef Struct Reference</a>	42
3.45.1	Field Documentation	43
3.45.1.1	info	43
3.45.1.2	model	43
3.46	<a href="#">_virDomainMemoryStat Struct Reference</a>	43
3.46.1	Field Documentation	43
3.46.1.1	tag	43
3.46.1.2	val	43
3.47	<a href="#">_virDomainNetDef Struct Reference</a>	43

3.47.1	Field Documentation	44
3.47.1.1	actual	44
3.47.1.2	address	44
3.47.1.3	bandwidth	44
3.47.1.4	bridge	44
3.47.1.5	brname	44
3.47.1.6	<b>POL New</b> brtype	44
3.47.1.7	data	45
3.47.1.8	def	45
3.47.1.9	dev	45
3.47.1.10	direct	45
3.47.1.11	driver	45
3.47.1.12	ethernet	45
3.47.1.13	event_idx	45
3.47.1.14	filter	45
3.47.1.15	filterparams	45
3.47.1.16	hostdev	45
3.47.1.17	ifname	45
3.47.1.18	info	45
3.47.1.19	internal	45
3.47.1.20	ioeventfd	45
3.47.1.21	ipaddr	45
3.47.1.22	linkdev	45
3.47.1.23	linkstate	45
3.47.1.24	mac	45
3.47.1.25	mode	45
3.47.1.26	model	45
3.47.1.27	name	45
3.47.1.28	name	45
3.47.1.29	network	45
3.47.1.30	port	45
3.47.1.31	portgroup	45
3.47.1.32	script	45
3.47.1.33	sndbuf	45
3.47.1.34	sndbuf_specified	45
3.47.1.35	socket	46
3.47.1.36	tune	46
3.47.1.37	txmode	46
3.47.1.38	type	46
3.47.1.39	virtio	46



3.47.1.40	virtPortProfile	46
3.47.1.41	vlan	46
3.48	_virDomainNumatuneDef Struct Reference	46
3.48.1	Field Documentation	46
3.48.1.1	memory	46
3.48.1.2	mode	46
3.48.1.3	nodemask	46
3.48.1.4	placement_mode	46
3.49	_virDomainObj Struct Reference	46
3.49.1	Field Documentation	47
3.49.1.1	autostart	47
3.49.1.2	current_snapshot	47
3.49.1.3	def	47
3.49.1.4	hasManagedSave	47
3.49.1.5	lock	47
3.49.1.6	newDef	47
3.49.1.7	object	47
3.49.1.8	persistent	47
3.49.1.9	pid	47
3.49.1.10	privateData	47
3.49.1.11	privateDataFreeFunc	47
3.49.1.12	snapshots	47
3.49.1.13	state	47
3.49.1.14	taint	47
3.49.1.15	updated	47
3.50	_virDomainObjList Struct Reference	47
3.50.1	Field Documentation	48
3.50.1.1	objs	48
3.51	_virDomainOSDef Struct Reference	48
3.51.1	Field Documentation	48
3.51.1.1	arch	48
3.51.1.2	bios	48
3.51.1.3	bootDevs	48
3.51.1.4	bootloader	48
3.51.1.5	bootloaderArgs	48
3.51.1.6	bootmenu	48
3.51.1.7	cmdline	48
3.51.1.8	init	49
3.51.1.9	initargv	49
3.51.1.10	initrd	49

3.51.1.11 kernel . . . . .	49
3.51.1.12 loader . . . . .	49
3.51.1.13 machine . . . . .	49
3.51.1.14 nBootDevs . . . . .	49
3.51.1.15 root . . . . .	49
3.51.1.16 smbios_mode . . . . .	49
3.51.1.17 type . . . . .	49
3.52 _virDomainRedirdevDef Struct Reference . . . . .	49
3.52.1 Field Documentation . . . . .	49
3.52.1.1 bus . . . . .	49
3.52.1.2 chr . . . . .	49
3.52.1.3 info . . . . .	49
3.52.1.4 source . . . . .	49
3.53 _virDomainRedirFilterDef Struct Reference . . . . .	50
3.53.1 Field Documentation . . . . .	50
3.53.1.1 nusbdevs . . . . .	50
3.53.1.2 usbdevs . . . . .	50
3.54 _virDomainRedirFilterUsbDevDef Struct Reference . . . . .	50
3.54.1 Field Documentation . . . . .	50
3.54.1.1 allow . . . . .	50
3.54.1.2 product . . . . .	50
3.54.1.3 usbClass . . . . .	50
3.54.1.4 vendor . . . . .	50
3.54.1.5 version . . . . .	50
3.55 _virDomainSmartcardDef Struct Reference . . . . .	50
3.55.1 Field Documentation . . . . .	51
3.55.1.1 cert . . . . .	51
3.55.1.2 data . . . . .	51
3.55.1.3 database . . . . .	51
3.55.1.4 file . . . . .	51
3.55.1.5 info . . . . .	51
3.55.1.6 passthru . . . . .	51
3.55.1.7 type . . . . .	51
3.56 _virDomainSoundCodecDef Struct Reference . . . . .	51
3.56.1 Field Documentation . . . . .	51
3.56.1.1 cad . . . . .	51
3.56.1.2 type . . . . .	51
3.57 _virDomainSoundDef Struct Reference . . . . .	52
3.57.1 Field Documentation . . . . .	52
3.57.1.1 codecs . . . . .	52

3.57.1.2	info	52
3.57.1.3	model	52
3.57.1.4	ncodecs	52
3.58	_virDomainStateReason Struct Reference	52
3.58.1	Field Documentation	52
3.58.1.1	reason	52
3.58.1.2	state	52
3.59	_virDomainTimerCatchupDef Struct Reference	52
3.59.1	Field Documentation	53
3.59.1.1	limit	53
3.59.1.2	slew	53
3.59.1.3	threshold	53
3.60	_virDomainTimerDef Struct Reference	53
3.60.1	Field Documentation	53
3.60.1.1	catchup	53
3.60.1.2	frequency	53
3.60.1.3	mode	53
3.60.1.4	name	53
3.60.1.5	present	53
3.60.1.6	tickpolicy	53
3.60.1.7	track	53
3.61	_virDomainVcpuPinDef Struct Reference	54
3.61.1	Field Documentation	54
3.61.1.1	cpumask	54
3.61.1.2	vcpuid	54
3.62	_virDomainVideoAccelDef Struct Reference	54
3.62.1	Field Documentation	54
3.62.1.1	support2d	54
3.62.1.2	support3d	54
3.63	_virDomainVideoDef Struct Reference	54
3.63.1	Field Documentation	55
3.63.1.1	accel	55
3.63.1.2	heads	55
3.63.1.3	info	55
3.63.1.4	type	55
3.63.1.5	vram	55
3.64	_virDomainVirtioSerialOpts Struct Reference	55
3.64.1	Field Documentation	55
3.64.1.1	ports	55
3.64.1.2	vectors	55

3.65	<a href="#">_virDomainWatchdogDef Struct Reference</a>	55
3.65.1	<a href="#">Field Documentation</a>	55
3.65.1.1	<a href="#">action</a>	55
3.65.1.2	<a href="#">info</a>	55
3.65.1.3	<a href="#">model</a>	56
3.66	<a href="#">_virDriver Struct Reference</a>	56
3.66.1	<a href="#">Detailed Description</a>	59
3.66.2	<a href="#">Field Documentation</a>	59
3.66.2.1	<a href="#">close</a>	59
3.66.2.2	<a href="#">cpuBaseline</a>	59
3.66.2.3	<a href="#">cpuCompare</a>	59
3.66.2.4	<a href="#">domainAbortJob</a>	59
3.66.2.5	<a href="#">domainAttachDevice</a>	59
3.66.2.6	<a href="#">domainAttachDeviceFlags</a>	59
3.66.2.7	<a href="#">domainBlockCommit</a>	60
3.66.2.8	<a href="#">domainBlockJobAbort</a>	60
3.66.2.9	<a href="#">domainBlockJobSetSpeed</a>	60
3.66.2.10	<a href="#">domainBlockPeek</a>	60
3.66.2.11	<a href="#">domainBlockPull</a>	60
3.66.2.12	<a href="#">domainBlockRebase</a>	60
3.66.2.13	<a href="#">domainBlockResize</a>	60
3.66.2.14	<a href="#">domainBlockStats</a>	60
3.66.2.15	<a href="#">domainBlockStatsFlags</a>	60
3.66.2.16	<a href="#">domainCoreDump</a>	60
3.66.2.17	<a href="#">domainCreate</a>	60
3.66.2.18	<a href="#">domainCreateWithFlags</a>	60
3.66.2.19	<a href="#">domainCreateXML</a>	60
3.66.2.20	<a href="#">domainDefineXML</a>	60
3.66.2.21	<a href="#">domainDestroy</a>	60
3.66.2.22	<a href="#">domainDestroyFlags</a>	60
3.66.2.23	<a href="#">domainDetachDevice</a>	60
3.66.2.24	<a href="#">domainDetachDeviceFlags</a>	60
3.66.2.25	<a href="#">domainEventDeregister</a>	60
3.66.2.26	<a href="#">domainEventDeregisterAny</a>	60
3.66.2.27	<a href="#">domainEventRegister</a>	60
3.66.2.28	<a href="#">domainEventRegisterAny</a>	60
3.66.2.29	<a href="#">domainGetAutostart</a>	60
3.66.2.30	<a href="#">domainGetBlkioParameters</a>	60
3.66.2.31	<a href="#">domainGetBlockInfo</a>	60
3.66.2.32	<a href="#">domainGetBlockIoTune</a>	60

3.66.2.33 domainGetBlockJobInfo . . . . .	60
3.66.2.34 domainGetControlInfo . . . . .	60
3.66.2.35 domainGetCPUStats . . . . .	61
3.66.2.36 domainGetDiskErrors . . . . .	61
3.66.2.37 domainGetEmulatorPinInfo . . . . .	61
3.66.2.38 domainGetHostname . . . . .	61
3.66.2.39 domainGetInfo . . . . .	61
3.66.2.40 domainGetInterfaceParameters . . . . .	61
3.66.2.41 domainGetJobInfo . . . . .	61
3.66.2.42 domainGetMaxMemory . . . . .	61
3.66.2.43 domainGetMaxVcpus . . . . .	61
3.66.2.44 domainGetMemoryParameters . . . . .	61
3.66.2.45 domainGetMetadata . . . . .	61
3.66.2.46 domainGetNumaParameters . . . . .	61
3.66.2.47 domainGetOSType . . . . .	61
3.66.2.48 domainGetSchedulerParameters . . . . .	61
3.66.2.49 domainGetSchedulerParametersFlags . . . . .	61
3.66.2.50 domainGetSchedulerType . . . . .	61
3.66.2.51 domainGetSecurityLabel . . . . .	61
3.66.2.52 domainGetSecurityLabelList . . . . .	61
3.66.2.53 domainGetState . . . . .	61
3.66.2.54 domainGetVcpuPinInfo . . . . .	61
3.66.2.55 domainGetVcpus . . . . .	61
3.66.2.56 domainGetVcpusFlags . . . . .	61
3.66.2.57 domainGetXMLDesc . . . . .	61
3.66.2.58 domainHasCurrentSnapshot . . . . .	61
3.66.2.59 domainHasManagedSavelImage . . . . .	61
3.66.2.60 domainInjectNMI . . . . .	61
3.66.2.61 domainInterfaceStats . . . . .	61
3.66.2.62 domainIsActive . . . . .	61
3.66.2.63 domainIsPersistent . . . . .	62
3.66.2.64 domainIsUpdated . . . . .	62
3.66.2.65 domainListAllSnapshots . . . . .	62
3.66.2.66 domainLookupByID . . . . .	62
3.66.2.67 domainLookupByName . . . . .	62
3.66.2.68 domainLookupByUUID . . . . .	62
3.66.2.69 domainManagedSave . . . . .	62
3.66.2.70 domainManagedSaveRemove . . . . .	62
3.66.2.71 domainMemoryPeek . . . . .	62
3.66.2.72 domainMemoryStats . . . . .	62

3.66.2.73 domainMigrateBegin3 . . . . .	62
3.66.2.74 domainMigrateConfirm3 . . . . .	62
3.66.2.75 domainMigrateFinish . . . . .	62
3.66.2.76 domainMigrateFinish2 . . . . .	62
3.66.2.77 domainMigrateFinish3 . . . . .	62
3.66.2.78 domainMigrateGetMaxSpeed . . . . .	62
3.66.2.79 domainMigratePerform . . . . .	62
3.66.2.80 domainMigratePerform3 . . . . .	62
3.66.2.81 domainMigratePrepare . . . . .	62
3.66.2.82 domainMigratePrepare2 . . . . .	62
3.66.2.83 domainMigratePrepare3 . . . . .	62
3.66.2.84 domainMigratePrepareTunnel . . . . .	62
3.66.2.85 domainMigratePrepareTunnel3 . . . . .	62
3.66.2.86 domainMigrateSetMaxDowntime . . . . .	62
3.66.2.87 domainMigrateSetMaxSpeed . . . . .	62
3.66.2.88 domainOpenConsole . . . . .	62
3.66.2.89 domainOpenGraphics . . . . .	62
3.66.2.90 domainPinEmulator . . . . .	62
3.66.2.91 domainPinVcpu . . . . .	63
3.66.2.92 domainPinVcpuFlags . . . . .	63
3.66.2.93 domainPMSuspendForDuration . . . . .	63
3.66.2.94 domainPMWakeup . . . . .	63
3.66.2.95 domainReboot . . . . .	63
3.66.2.96 domainReset . . . . .	63
3.66.2.97 domainRestore . . . . .	63
3.66.2.98 domainRestoreFlags . . . . .	63
3.66.2.99 domainResume . . . . .	63
3.66.2.100domainRevertToSnapshot . . . . .	63
3.66.2.101domainSave . . . . .	63
3.66.2.102domainSaveFlags . . . . .	63
3.66.2.103domainSavelImageDefineXML . . . . .	63
3.66.2.104domainSavelImageGetXMLDesc . . . . .	63
3.66.2.105domainScreenshot . . . . .	63
3.66.2.106domainSendKey . . . . .	63
3.66.2.107domainSetAutostart . . . . .	63
3.66.2.108domainSetBlkioParameters . . . . .	63
3.66.2.109domainSetBlockIoTune . . . . .	63
3.66.2.110domainSetInterfaceParameters . . . . .	63
3.66.2.111domainSetMaxMemory . . . . .	63
3.66.2.112domainSetMemory . . . . .	63

3.66.2.113domainSetMemoryFlags . . . . .	63
3.66.2.114domainSetMemoryParameters . . . . .	63
3.66.2.115domainSetMetadata . . . . .	63
3.66.2.116domainSetNumaParameters . . . . .	63
3.66.2.117domainSetSchedulerParameters . . . . .	63
3.66.2.118domainSetSchedulerParametersFlags . . . . .	63
3.66.2.119domainSetVcpus . . . . .	64
3.66.2.120domainSetVcpusFlags . . . . .	64
3.66.2.121domainShutdown . . . . .	64
3.66.2.122domainShutdownFlags . . . . .	64
3.66.2.123domainSnapshotCreateXML . . . . .	64
3.66.2.124domainSnapshotCurrent . . . . .	64
3.66.2.125domainSnapshotDelete . . . . .	64
3.66.2.126domainSnapshotGetParent . . . . .	64
3.66.2.127domainSnapshotGetXMLDesc . . . . .	64
3.66.2.128domainSnapshotHasMetadata . . . . .	64
3.66.2.129domainSnapshotIsCurrent . . . . .	64
3.66.2.130domainSnapshotListAllChildren . . . . .	64
3.66.2.131domainSnapshotListChildrenNames . . . . .	64
3.66.2.132domainSnapshotListNames . . . . .	64
3.66.2.133domainSnapshotLookupByName . . . . .	64
3.66.2.134domainSnapshotNum . . . . .	64
3.66.2.135domainSnapshotNumChildren . . . . .	64
3.66.2.136domainSuspend . . . . .	64
3.66.2.137domainUndefine . . . . .	64
3.66.2.138domainUndefineFlags . . . . .	64
3.66.2.139domainUpdateDeviceFlags . . . . .	64
3.66.2.140domainXMLFromNative . . . . .	64
3.66.2.141domainXMLToNative . . . . .	64
3.66.2.142getCapabilities . . . . .	64
3.66.2.143getHostname . . . . .	64
3.66.2.144getMaxVcpus . . . . .	64
3.66.2.145getSysinfo . . . . .	64
3.66.2.146sAlive . . . . .	64
3.66.2.147sEncrypted . . . . .	65
3.66.2.148sSecure . . . . .	65
3.66.2.149ibvirtVersion . . . . .	65
3.66.2.150istAllDomains . . . . .	65
3.66.2.151listDefinedDomains . . . . .	65
3.66.2.152istDomains . . . . .	65

3.66.2.153name	65
3.66.2.154no	65
3.66.2.155nodeDeviceDettach	65
3.66.2.156nodeDeviceReAttach	65
3.66.2.157nodeDeviceReset	65
3.66.2.158nodeGetCellsFreeMemory	65
3.66.2.159nodeGetCPUStats	65
3.66.2.160nodeGetFreeMemory	65
3.66.2.161nodeGetInfo	65
3.66.2.162nodeGetMemoryParameters	65
3.66.2.163nodeGetMemoryStats	65
3.66.2.164nodeGetSecurityModel	65
3.66.2.165nodeSetMemoryParameters	65
3.66.2.166nodeSuspendForDuration	65
3.66.2.167numOfDefinedDomains	65
3.66.2.168numOfDomains	65
3.66.2.169open	65
3.66.2.170qemuDomainArbitraryAgentCommand	65
3.66.2.171qemuDomainAttach	65
3.66.2.172qemuDomainMonitorCommand	65
3.66.2.173setKeepAlive	65
3.66.2.174supports_feature	65
3.66.2.175type	66
3.66.2.176version	66
3.67 _virInterfaceDriver Struct Reference	66
3.67.1 Detailed Description	66
3.67.2 Field Documentation	66
3.67.2.1 close	66
3.67.2.2 interfaceChangeBegin	66
3.67.2.3 interfaceChangeCommit	66
3.67.2.4 interfaceChangeRollback	67
3.67.2.5 interfaceCreate	67
3.67.2.6 interfaceDefineXML	67
3.67.2.7 interfaceDestroy	67
3.67.2.8 interfaceGetXMLDesc	67
3.67.2.9 interfaceIsActive	67
3.67.2.10 interfaceLookupByMACString	67
3.67.2.11 interfaceLookupByName	67
3.67.2.12 interfaceUndefine	67
3.67.2.13 listAllInterfaces	67



3.67.2.14	listDefinedInterfaces	67
3.67.2.15	listInterfaces	67
3.67.2.16	name	67
3.67.2.17	numOfDefinedInterfaces	67
3.67.2.18	numOfInterfaces	67
3.67.2.19	open	67
3.68	POL Mod _virNetworkDef Struct Reference	67
3.68.1	Field Documentation	68
3.68.1.1	bandwidth	68
3.68.1.2	bridge	68
3.68.1.3	POL New bridge_type	68
3.68.1.4	connections	68
3.68.1.5	delay	68
3.68.1.6	dns	68
3.68.1.7	domain	68
3.68.1.8	forwardIfs	68
3.68.1.9	forwardPfs	68
3.68.1.10	forwardType	68
3.68.1.11	ips	68
3.68.1.12	mac	68
3.68.1.13	mac_specified	68
3.68.1.14	managed	68
3.68.1.15	name	68
3.68.1.16	nForwardIfs	68
3.68.1.17	nForwardPfs	68
3.68.1.18	nips	68
3.68.1.19	nPortGroups	69
3.68.1.20	POL New nTunnels	69
3.68.1.21	portGroups	69
3.68.1.22	POL New source_bridge	69
3.68.1.23	stp	69
3.68.1.24	POL New tunnels	69
3.68.1.25	uuid	69
3.68.1.26	uuid_specified	69
3.68.1.27	virtPortProfile	69
3.68.1.28	vlan	69
3.69	_virNetworkDHCPHostDef Struct Reference	69
3.69.1	Field Documentation	69
3.69.1.1	ip	69
3.69.1.2	mac	69

3.69.1.3	name	69
3.70	<a href="#">_virNetworkDHCPRangeDef Struct Reference</a>	69
3.70.1	Field Documentation	70
3.70.1.1	end	70
3.70.1.2	start	70
3.71	<a href="#">_virNetworkDNSDef Struct Reference</a>	70
3.71.1	Field Documentation	70
3.71.1.1	hosts	70
3.71.1.2	nhosts	70
3.71.1.3	nsrvrecords	70
3.71.1.4	ntxtrecords	70
3.71.1.5	srvrecords	70
3.71.1.6	txtrecords	70
3.72	<a href="#">_virNetworkDNSHostsDef Struct Reference</a>	70
3.72.1	Field Documentation	71
3.72.1.1	ip	71
3.72.1.2	names	71
3.72.1.3	nnames	71
3.73	<a href="#">_virNetworkDNSSrvRecordsDef Struct Reference</a>	71
3.73.1	Field Documentation	71
3.73.1.1	domain	71
3.73.1.2	port	71
3.73.1.3	priority	71
3.73.1.4	protocol	71
3.73.1.5	service	71
3.73.1.6	target	71
3.73.1.7	weight	71
3.74	<a href="#">_virNetworkDNSTxtRecordsDef Struct Reference</a>	72
3.74.1	Field Documentation	72
3.74.1.1	name	72
3.74.1.2	value	72
3.75	<a href="#">_virNetworkDriver Struct Reference</a>	72
3.75.1	Detailed Description	73
3.75.2	Field Documentation	73
3.75.2.1	close	73
3.75.2.2	listAllNetworks	73
3.75.2.3	listDefinedNetworks	73
3.75.2.4	listNetworks	73
3.75.2.5	name	73
3.75.2.6	networkCreate	73

3.75.2.7	<a href="#">networkCreateXML</a>	73
3.75.2.8	<a href="#">networkDefineXML</a>	73
3.75.2.9	<a href="#">networkDestroy</a>	73
3.75.2.10	<a href="#">networkGetAutostart</a>	73
3.75.2.11	<a href="#">networkGetBridgeName</a>	73
3.75.2.12	<a href="#">networkGetBridgeType</a>	73
3.75.2.13	<a href="#">networkGetXMLDesc</a>	73
3.75.2.14	<a href="#">networkIsActive</a>	73
3.75.2.15	<a href="#">networkIsPersistent</a>	73
3.75.2.16	<a href="#">networkLookupByName</a>	73
3.75.2.17	<a href="#">networkLookupByUUID</a>	73
3.75.2.18	<a href="#">networkSetAutostart</a>	73
3.75.2.19	<a href="#">networkUndefine</a>	73
3.75.2.20	<a href="#">networkUpdate</a>	73
3.75.2.21	<a href="#">numOfDefinedNetworks</a>	73
3.75.2.22	<a href="#">numOfNetworks</a>	73
3.75.2.23	<a href="#">open</a>	74
3.76	<a href="#">_virNetworkForwardIfDef Struct Reference</a>	74
3.76.1	<a href="#">Field Documentation</a>	74
3.76.1.1	<a href="#">connections</a>	74
3.76.1.2	<a href="#">dev</a>	74
3.76.1.3	<a href="#">device</a>	74
3.76.1.4	<a href="#">pci</a>	74
3.76.1.5	<a href="#">type</a>	74
3.77	<a href="#">_virNetworkForwardPfDef Struct Reference</a>	74
3.77.1	<a href="#">Field Documentation</a>	74
3.77.1.1	<a href="#">connections</a>	74
3.77.1.2	<a href="#">dev</a>	75
3.78	<a href="#">_virNetworkIpDef Struct Reference</a>	75
3.78.1	<a href="#">Field Documentation</a>	75
3.78.1.1	<a href="#">address</a>	75
3.78.1.2	<a href="#">bootfile</a>	75
3.78.1.3	<a href="#">bootserver</a>	75
3.78.1.4	<a href="#">family</a>	75
3.78.1.5	<a href="#">hosts</a>	75
3.78.1.6	<a href="#">netmask</a>	75
3.78.1.7	<a href="#">nhosts</a>	75
3.78.1.8	<a href="#">nranges</a>	75
3.78.1.9	<a href="#">prefix</a>	75
3.78.1.10	<a href="#">ranges</a>	75

3.78.1.11 tftproot . . . . .	75
3.79 <a href="#">_virNetworkObj Struct Reference</a> . . . . .	76
3.79.1 Field Documentation . . . . .	76
3.79.1.1 active . . . . .	76
3.79.1.2 autostart . . . . .	76
3.79.1.3 def . . . . .	76
3.79.1.4 dnsmasqPid . . . . .	76
3.79.1.5 lock . . . . .	76
3.79.1.6 newDef . . . . .	76
3.79.1.7 persistent . . . . .	76
3.79.1.8 radvdPid . . . . .	76
3.80 <a href="#">_virNetworkObjList Struct Reference</a> . . . . .	76
3.80.1 Field Documentation . . . . .	76
3.80.1.1 count . . . . .	76
3.80.1.2 objs . . . . .	77
3.81 <a href="#">_virNodeCPUStats Struct Reference</a> . . . . .	77
3.81.1 Field Documentation . . . . .	77
3.81.1.1 field . . . . .	77
3.81.1.2 value . . . . .	77
3.82 <a href="#">_virNodeInfo Struct Reference</a> . . . . .	77
3.82.1 Field Documentation . . . . .	77
3.82.1.1 cores . . . . .	77
3.82.1.2 cpus . . . . .	77
3.82.1.3 memory . . . . .	77
3.82.1.4 mhz . . . . .	77
3.82.1.5 model . . . . .	78
3.82.1.6 nodes . . . . .	78
3.82.1.7 sockets . . . . .	78
3.82.1.8 threads . . . . .	78
3.83 <a href="#">_virNodeMemoryStats Struct Reference</a> . . . . .	78
3.83.1 Field Documentation . . . . .	78
3.83.1.1 field . . . . .	78
3.83.1.2 value . . . . .	78
3.84 <a href="#">_virNWFilterDriver Struct Reference</a> . . . . .	78
3.84.1 Detailed Description . . . . .	79
3.84.2 Field Documentation . . . . .	79
3.84.2.1 close . . . . .	79
3.84.2.2 defineXML . . . . .	79
3.84.2.3 getXMLDesc . . . . .	79
3.84.2.4 listAllNWFilters . . . . .	79

3.84.2.5	listNWFilters	79
3.84.2.6	name	79
3.84.2.7	numOfNWFilters	79
3.84.2.8	nwfilterLookupByName	79
3.84.2.9	nwfilterLookupByUUID	79
3.84.2.10	open	79
3.84.2.11	undefine	79
3.85	_virPortGroupDef Struct Reference	79
3.85.1	Field Documentation	80
3.85.1.1	bandwidth	80
3.85.1.2	isDefault	80
3.85.1.3	name	80
3.85.1.4	virtPortProfile	80
3.85.1.5	vlan	80
3.86	_virSecretDriver Struct Reference	80
3.86.1	Detailed Description	80
3.86.2	Field Documentation	80
3.86.2.1	close	80
3.86.2.2	defineXML	80
3.86.2.3	getValue	81
3.86.2.4	getXMLDesc	81
3.86.2.5	listAllSecrets	81
3.86.2.6	listSecrets	81
3.86.2.7	lookupByUsage	81
3.86.2.8	lookupByUUID	81
3.86.2.9	name	81
3.86.2.10	numOfSecrets	81
3.86.2.11	open	81
3.86.2.12	setValue	81
3.86.2.13	undefine	81
3.87	_virSecurityDeviceLabelDef Struct Reference	81
3.87.1	Field Documentation	81
3.87.1.1	label	81
3.87.1.2	model	81
3.87.1.3	norelabel	81
3.88	_virSecurityLabel Struct Reference	81
3.88.1	Detailed Description	82
3.88.2	Field Documentation	82
3.88.2.1	enforcing	82
3.88.2.2	label	82

3.89	<a href="#">_virSecurityLabelDef Struct Reference</a>	82
3.89.1	<a href="#">Field Documentation</a>	82
3.89.1.1	<a href="#">baselabel</a>	82
3.89.1.2	<a href="#">imagelabel</a>	82
3.89.1.3	<a href="#">implicit</a>	82
3.89.1.4	<a href="#">label</a>	82
3.89.1.5	<a href="#">model</a>	82
3.89.1.6	<a href="#">norelabel</a>	82
3.89.1.7	<a href="#">type</a>	82
3.90	<a href="#">_virSecurityModel Struct Reference</a>	83
3.90.1	<a href="#">Detailed Description</a>	83
3.90.2	<a href="#">Field Documentation</a>	83
3.90.2.1	<a href="#">doi</a>	83
3.90.2.2	<a href="#">model</a>	83
3.91	<a href="#">_virStorageDriver Struct Reference</a>	83
3.91.1	<a href="#">Detailed Description</a>	84
3.91.2	<a href="#">Field Documentation</a>	84
3.91.2.1	<a href="#">close</a>	84
3.91.2.2	<a href="#">findPoolSources</a>	84
3.91.2.3	<a href="#">listAllPools</a>	84
3.91.2.4	<a href="#">listDefinedPools</a>	84
3.91.2.5	<a href="#">listPools</a>	84
3.91.2.6	<a href="#">name</a>	84
3.91.2.7	<a href="#">numOfDefinedPools</a>	84
3.91.2.8	<a href="#">numOfPools</a>	85
3.91.2.9	<a href="#">open</a>	85
3.91.2.10	<a href="#">poolBuild</a>	85
3.91.2.11	<a href="#">poolCreate</a>	85
3.91.2.12	<a href="#">poolCreateXML</a>	85
3.91.2.13	<a href="#">poolDefineXML</a>	85
3.91.2.14	<a href="#">poolDelete</a>	85
3.91.2.15	<a href="#">poolDestroy</a>	85
3.91.2.16	<a href="#">poolGetAutostart</a>	85
3.91.2.17	<a href="#">poolGetInfo</a>	85
3.91.2.18	<a href="#">poolGetXMLDesc</a>	85
3.91.2.19	<a href="#">poolsActive</a>	85
3.91.2.20	<a href="#">poolsPersistent</a>	85
3.91.2.21	<a href="#">poolListAllVolumes</a>	85
3.91.2.22	<a href="#">poolListVolumes</a>	85
3.91.2.23	<a href="#">poolLookupByName</a>	85

3.91.2.24 poolLookupByUUID . . . . .	85
3.91.2.25 poolLookupByVolume . . . . .	85
3.91.2.26 poolNumOfVolumes . . . . .	85
3.91.2.27 poolRefresh . . . . .	85
3.91.2.28 poolSetAutostart . . . . .	85
3.91.2.29 poolUndefine . . . . .	85
3.91.2.30 volCreateXML . . . . .	85
3.91.2.31 volCreateXMLFrom . . . . .	85
3.91.2.32 volDelete . . . . .	85
3.91.2.33 volDownload . . . . .	85
3.91.2.34 volGetInfo . . . . .	85
3.91.2.35 volGetPath . . . . .	85
3.91.2.36 volGetXMLDesc . . . . .	86
3.91.2.37 volLookupByKey . . . . .	86
3.91.2.38 volLookupByName . . . . .	86
3.91.2.39 volLookupByPath . . . . .	86
3.91.2.40 volResize . . . . .	86
3.91.2.41 volUpload . . . . .	86
3.91.2.42 volWipe . . . . .	86
3.91.2.43 volWipePattern . . . . .	86
3.92 _virStoragePoolInfo Struct Reference . . . . .	86
3.92.1 Field Documentation . . . . .	86
3.92.1.1 allocation . . . . .	86
3.92.1.2 available . . . . .	86
3.92.1.3 capacity . . . . .	86
3.92.1.4 state . . . . .	86
3.93 _virStorageVolInfo Struct Reference . . . . .	86
3.93.1 Field Documentation . . . . .	87
3.93.1.1 allocation . . . . .	87
3.93.1.2 capacity . . . . .	87
3.93.1.3 type . . . . .	87
3.94 _virStreamDriver Struct Reference . . . . .	87
3.94.1 Field Documentation . . . . .	87
3.94.1.1 streamAbort . . . . .	87
3.94.1.2 streamAddCallback . . . . .	87
3.94.1.3 streamFinish . . . . .	87
3.94.1.4 streamRecv . . . . .	87
3.94.1.5 streamRemoveCallback . . . . .	87
3.94.1.6 streamSend . . . . .	87
3.94.1.7 streamUpdateCallback . . . . .	87

3.95	<b>POL New</b> <a href="#">_virTunnelDef Struct Reference</a>	88
3.95.1	<a href="#">Field Documentation</a>	88
3.95.1.1	<a href="#">device</a>	88
3.95.1.2	<a href="#">remote_ip</a>	88
3.95.1.3	<a href="#">trunks</a>	88
3.96	<a href="#">_virTypedParameter Struct Reference</a>	88
3.96.1	<a href="#">Field Documentation</a>	88
3.96.1.1	<a href="#">b</a>	88
3.96.1.2	<a href="#">d</a>	88
3.96.1.3	<a href="#">field</a>	88
3.96.1.4	<a href="#">i</a>	88
3.96.1.5	<a href="#">l</a>	89
3.96.1.6	<a href="#">s</a>	89
3.96.1.7	<a href="#">type</a>	89
3.96.1.8	<a href="#">ui</a>	89
3.96.1.9	<a href="#">ul</a>	89
3.96.1.10	<a href="#">value</a>	89
3.97	<a href="#">_virVcpuInfo Struct Reference</a>	89
3.97.1	<a href="#">Field Documentation</a>	89
3.97.1.1	<a href="#">cpu</a>	89
3.97.1.2	<a href="#">cpuTime</a>	89
3.97.1.3	<a href="#">number</a>	89
3.97.1.4	<a href="#">state</a>	89
3.98	<a href="#">network_driver Struct Reference</a>	89
3.98.1	<a href="#">Field Documentation</a>	90
3.98.1.1	<a href="#">dnsmasqCaps</a>	90
3.98.1.2	<a href="#">iptables</a>	90
3.98.1.3	<a href="#">lock</a>	90
3.98.1.4	<a href="#">logDir</a>	90
3.98.1.5	<a href="#">networkAutostartDir</a>	90
3.98.1.6	<a href="#">networkConfigDir</a>	90
3.98.1.7	<a href="#">networks</a>	90
3.99	<a href="#">virDomainIDDData Struct Reference</a>	90
3.99.1	<a href="#">Field Documentation</a>	90
3.99.1.1	<a href="#">ids</a>	90
3.99.1.2	<a href="#">maxids</a>	90
3.99.1.3	<a href="#">numids</a>	90
3.100	<a href="#">virDomainListData Struct Reference</a>	90
3.100.1	<a href="#">Field Documentation</a>	91
3.100.1.1	<a href="#">conn</a>	91



3.100.1.2 domains	91
3.100.1.3 error	91
3.100.1.4 flags	91
3.100.1.5 ndomains	91
3.101 virDomainNameData Struct Reference	91
3.101.1 Field Documentation	91
3.101.1.1 maxnames	91
3.101.1.2 names	91
3.101.1.3 numnames	91
3.101.1.4 oom	91
3.102 vshNetworkList Struct Reference	91
3.102.1 Field Documentation	91
3.102.1.1 nets	91
3.102.1.2 nnets	91
<b>4 File Documentation</b>	<b>93</b>
4.1 include/libvirt/libvirt.h File Reference	93
4.1.1 Macro Definition Documentation	111
4.1.1.1 _virBlkioParameter	111
4.1.1.2 _virMemoryParameter	111
4.1.1.3 _virSchedParameter	111
4.1.1.4 LIBVIR_VERSION_NUMBER	111
4.1.1.5 VIR_COPY_CPUMAP	111
4.1.1.6 VIR_CPU_MAPLEN	111
4.1.1.7 VIR_CPU_USABLE	112
4.1.1.8 VIR_DEPRECATED	112
4.1.1.9 VIR_DOMAIN_BANDWIDTH_IN_AVERAGE	112
4.1.1.10 VIR_DOMAIN_BANDWIDTH_IN_BURST	112
4.1.1.11 VIR_DOMAIN_BANDWIDTH_IN_PEAK	112
4.1.1.12 VIR_DOMAIN_BANDWIDTH_OUT_AVERAGE	112
4.1.1.13 VIR_DOMAIN_BANDWIDTH_OUT_BURST	112
4.1.1.14 VIR_DOMAIN_BANDWIDTH_OUT_PEAK	112
4.1.1.15 VIR_DOMAIN_BLKIO_DEVICE_WEIGHT	112
4.1.1.16 VIR_DOMAIN_BLKIO_FIELD_LENGTH	113
4.1.1.17 VIR_DOMAIN_BLKIO_WEIGHT	113
4.1.1.18 VIR_DOMAIN_BLOCK_IOTUNE_READ_BYTES_SEC	113
4.1.1.19 VIR_DOMAIN_BLOCK_IOTUNE_READ_IOPS_SEC	113
4.1.1.20 VIR_DOMAIN_BLOCK_IOTUNE_TOTAL_BYTES_SEC	113
4.1.1.21 VIR_DOMAIN_BLOCK_IOTUNE_TOTAL_IOPS_SEC	113
4.1.1.22 VIR_DOMAIN_BLOCK_IOTUNE_WRITE_BYTES_SEC	113

4.1.1.23	VIR_DOMAIN_BLOCK_IOTUNE_WRITE_IOPS_SEC	113
4.1.1.24	VIR_DOMAIN_BLOCK_STATS_ERRS	114
4.1.1.25	VIR_DOMAIN_BLOCK_STATS_FIELD_LENGTH	114
4.1.1.26	VIR_DOMAIN_BLOCK_STATS_FLUSH_REQ	114
4.1.1.27	VIR_DOMAIN_BLOCK_STATS_FLUSH_TOTAL_TIMES	114
4.1.1.28	VIR_DOMAIN_BLOCK_STATS_READ_BYTES	114
4.1.1.29	VIR_DOMAIN_BLOCK_STATS_READ_REQ	114
4.1.1.30	VIR_DOMAIN_BLOCK_STATS_READ_TOTAL_TIMES	114
4.1.1.31	VIR_DOMAIN_BLOCK_STATS_WRITE_BYTES	114
4.1.1.32	VIR_DOMAIN_BLOCK_STATS_WRITE_REQ	114
4.1.1.33	VIR_DOMAIN_BLOCK_STATS_WRITE_TOTAL_TIMES	115
4.1.1.34	VIR_DOMAIN_CPU_STATS_CPUTIME	115
4.1.1.35	VIR_DOMAIN_CPU_STATS_SYSTEMTIME	115
4.1.1.36	VIR_DOMAIN_CPU_STATS_USERTIME	115
4.1.1.37	VIR_DOMAIN_CPU_STATS_VCPUTIME	115
4.1.1.38	VIR_DOMAIN_EVENT_CALLBACK	115
4.1.1.39	VIR_DOMAIN_MEMORY_FIELD_LENGTH	115
4.1.1.40	VIR_DOMAIN_MEMORY_HARD_LIMIT	115
4.1.1.41	VIR_DOMAIN_MEMORY_MIN_GUARANTEE	115
4.1.1.42	VIR_DOMAIN_MEMORY_PARAM_UNLIMITED	115
4.1.1.43	VIR_DOMAIN_MEMORY_SOFT_LIMIT	116
4.1.1.44	VIR_DOMAIN_MEMORY_SWAP_HARD_LIMIT	116
4.1.1.45	VIR_DOMAIN_NUMA_MODE	116
4.1.1.46	VIR_DOMAIN_NUMA_NODESET	116
4.1.1.47	VIR_DOMAIN_SCHED_FIELD_LENGTH	116
4.1.1.48	VIR_DOMAIN_SCHEDULER_CAP	116
4.1.1.49	VIR_DOMAIN_SCHEDULER_CPU_SHARES	116
4.1.1.50	VIR_DOMAIN_SCHEDULER_EMULATOR_PERIOD	116
4.1.1.51	VIR_DOMAIN_SCHEDULER_EMULATOR_QUOTA	117
4.1.1.52	VIR_DOMAIN_SCHEDULER_LIMIT	117
4.1.1.53	VIR_DOMAIN_SCHEDULER_RESERVATION	117
4.1.1.54	VIR_DOMAIN_SCHEDULER_SHARES	117
4.1.1.55	VIR_DOMAIN_SCHEDULER_VCPU_PERIOD	117
4.1.1.56	VIR_DOMAIN_SCHEDULER_VCPU_QUOTA	117
4.1.1.57	VIR_DOMAIN_SCHEDULER_WEIGHT	117
4.1.1.58	VIR_DOMAIN_SEND_KEY_MAX_KEYS	117
4.1.1.59	VIR_EXPORT_VAR	117
4.1.1.60	VIR_GET_CPUMAP	118
4.1.1.61	VIR_NODE_CPU_STATS_FIELD_LENGTH	118
4.1.1.62	VIR_NODE_CPU_STATS_IDLE	118

4.1.1.63	VIR_NODE_CPU_STATS_IOWAIT	118
4.1.1.64	VIR_NODE_CPU_STATS_KERNEL	118
4.1.1.65	VIR_NODE_CPU_STATS_USER	118
4.1.1.66	VIR_NODE_CPU_STATS_UTILIZATION	118
4.1.1.67	VIR_NODE_MEMORY_SHARED_FULL_SCANS	118
4.1.1.68	VIR_NODE_MEMORY_SHARED_PAGES_SHARED	118
4.1.1.69	VIR_NODE_MEMORY_SHARED_PAGES_SHARING	118
4.1.1.70	VIR_NODE_MEMORY_SHARED_PAGES_TO_SCAN	118
4.1.1.71	VIR_NODE_MEMORY_SHARED_PAGES_UNSHARED	118
4.1.1.72	VIR_NODE_MEMORY_SHARED_PAGES_VOLATILE	118
4.1.1.73	VIR_NODE_MEMORY_SHARED_SLEEP_MILLISECS	119
4.1.1.74	VIR_NODE_MEMORY_STATS_BUFFERS	119
4.1.1.75	VIR_NODE_MEMORY_STATS_CACHED	119
4.1.1.76	VIR_NODE_MEMORY_STATS_FIELD_LENGTH	119
4.1.1.77	VIR_NODE_MEMORY_STATS_FREE	119
4.1.1.78	VIR_NODE_MEMORY_STATS_TOTAL	119
4.1.1.79	VIR_NODEINFO_MAXCPUS	119
4.1.1.80	VIR_SECURITY_DOI_BUFLen	119
4.1.1.81	VIR_SECURITY_LABEL_BUFLen	119
4.1.1.82	VIR_SECURITY_MODEL_BUFLen	120
4.1.1.83	VIR_TYPED_PARAM_FIELD_LENGTH	120
4.1.1.84	VIR_UNUSE_CPU	120
4.1.1.85	VIR_USE_CPU	120
4.1.1.86	VIR_UUID_BUFLen	120
4.1.1.87	VIR_UUID_STRING_BUFLen	120
4.1.2	Typedef Documentation	120
4.1.2.1	virBlkioParameter	120
4.1.2.2	virBlkioParameterPtr	120
4.1.2.3	virConnect	120
4.1.2.4	virConnectAuth	121
4.1.2.5	virConnectAuthCallbackPtr	121
4.1.2.6	virConnectAuthPtr	121
4.1.2.7	virConnectCloseFunc	121
4.1.2.8	virConnectCredential	121
4.1.2.9	virConnectCredentialPtr	121
4.1.2.10	virConnectDomainEventBalloonChangeCallback	121
4.1.2.11	virConnectDomainEventBlockJobCallback	121
4.1.2.12	virConnectDomainEventCallback	121
4.1.2.13	virConnectDomainEventDiskChangeCallback	121
4.1.2.14	virConnectDomainEventGenericCallback	122

4.1.2.15	<a href="#">virConnectDomainEventGraphicsCallback</a>	122
4.1.2.16	<a href="#">virConnectDomainEventIOErrorCallback</a>	122
4.1.2.17	<a href="#">virConnectDomainEventIOErrorReasonCallback</a>	122
4.1.2.18	<a href="#">virConnectDomainEventPMSuspendCallback</a>	122
4.1.2.19	<a href="#">virConnectDomainEventPMWakeupCallback</a>	122
4.1.2.20	<a href="#">virConnectDomainEventRTCChangeCallback</a>	123
4.1.2.21	<a href="#">virConnectDomainEventTrayChangeCallback</a>	123
4.1.2.22	<a href="#">virConnectDomainEventWatchdogCallback</a>	123
4.1.2.23	<a href="#">virConnectPtr</a>	123
4.1.2.24	<a href="#">virDomain</a>	123
4.1.2.25	<a href="#">virDomainBlockInfo</a>	123
4.1.2.26	<a href="#">virDomainBlockInfoPtr</a>	124
4.1.2.27	<a href="#">virDomainBlockJobCursor</a>	124
4.1.2.28	<a href="#">virDomainBlockJobInfo</a>	124
4.1.2.29	<a href="#">virDomainBlockJobInfoPtr</a>	124
4.1.2.30	<a href="#">virDomainBlockStatsPtr</a>	124
4.1.2.31	<a href="#">virDomainBlockStatsStruct</a>	124
4.1.2.32	<a href="#">virDomainControlInfo</a>	124
4.1.2.33	<a href="#">virDomainControlInfoPtr</a>	124
4.1.2.34	<a href="#">virDomainDiskError</a>	124
4.1.2.35	<a href="#">virDomainDiskErrorPtr</a>	125
4.1.2.36	<a href="#">virDomainEventGraphicsAddress</a>	125
4.1.2.37	<a href="#">virDomainEventGraphicsAddressPtr</a>	125
4.1.2.38	<a href="#">virDomainEventGraphicsSubject</a>	125
4.1.2.39	<a href="#">virDomainEventGraphicsSubjectIdentity</a>	125
4.1.2.40	<a href="#">virDomainEventGraphicsSubjectIdentityPtr</a>	125
4.1.2.41	<a href="#">virDomainEventGraphicsSubjectPtr</a>	125
4.1.2.42	<a href="#">virDomainInfo</a>	125
4.1.2.43	<a href="#">virDomainInfoPtr</a>	125
4.1.2.44	<a href="#">virDomainInterfaceStatsPtr</a>	125
4.1.2.45	<a href="#">virDomainInterfaceStatsStruct</a>	125
4.1.2.46	<a href="#">virDomainJobInfo</a>	125
4.1.2.47	<a href="#">virDomainJobInfoPtr</a>	125
4.1.2.48	<a href="#">virDomainMemoryStatPtr</a>	125
4.1.2.49	<a href="#">virDomainMemoryStatStruct</a>	125
4.1.2.50	<a href="#">virDomainPtr</a>	125
4.1.2.51	<a href="#">virDomainSnapshot</a>	126
4.1.2.52	<a href="#">virDomainSnapshotPtr</a>	126
4.1.2.53	<a href="#">virEventAddHandleFunc</a>	126
4.1.2.54	<a href="#">virEventAddTimeoutFunc</a>	126

4.1.2.55	<a href="#">virEventHandleCallback</a>	126
4.1.2.56	<a href="#">virEventRemoveHandleFunc</a>	126
4.1.2.57	<a href="#">virEventRemoveTimeoutFunc</a>	127
4.1.2.58	<a href="#">virEventTimeoutCallback</a>	127
4.1.2.59	<a href="#">virEventUpdateHandleFunc</a>	127
4.1.2.60	<a href="#">virEventUpdateTimeoutFunc</a>	127
4.1.2.61	<a href="#">virFreeCallback</a>	127
4.1.2.62	<a href="#">virInterface</a>	127
4.1.2.63	<a href="#">virInterfacePtr</a>	127
4.1.2.64	<a href="#">virMemoryParameter</a>	127
4.1.2.65	<a href="#">virMemoryParameterPtr</a>	127
4.1.2.66	<a href="#">virNetwork</a>	127
4.1.2.67	<a href="#">virNetworkPtr</a>	128
4.1.2.68	<a href="#">virNodeCPUStats</a>	128
4.1.2.69	<a href="#">virNodeCPUStatsPtr</a>	128
4.1.2.70	<a href="#">virNodeDevice</a>	128
4.1.2.71	<a href="#">virNodeDevicePtr</a>	128
4.1.2.72	<a href="#">virNodeInfo</a>	128
4.1.2.73	<a href="#">virNodeInfoPtr</a>	128
4.1.2.74	<a href="#">virNodeMemoryStats</a>	128
4.1.2.75	<a href="#">virNodeMemoryStatsPtr</a>	129
4.1.2.76	<a href="#">virNWFilter</a>	129
4.1.2.77	<a href="#">virNWFilterPtr</a>	129
4.1.2.78	<a href="#">virSchedParameter</a>	129
4.1.2.79	<a href="#">virSchedParameterPtr</a>	129
4.1.2.80	<a href="#">virSecret</a>	129
4.1.2.81	<a href="#">virSecretPtr</a>	129
4.1.2.82	<a href="#">virSecurityLabel</a>	129
4.1.2.83	<a href="#">virSecurityLabelPtr</a>	129
4.1.2.84	<a href="#">virSecurityModel</a>	129
4.1.2.85	<a href="#">virSecurityModelPtr</a>	130
4.1.2.86	<a href="#">virStoragePool</a>	130
4.1.2.87	<a href="#">virStoragePoolInfo</a>	130
4.1.2.88	<a href="#">virStoragePoolInfoPtr</a>	130
4.1.2.89	<a href="#">virStoragePoolPtr</a>	130
4.1.2.90	<a href="#">virStorageVol</a>	130
4.1.2.91	<a href="#">virStorageVolInfo</a>	130
4.1.2.92	<a href="#">virStorageVolInfoPtr</a>	130
4.1.2.93	<a href="#">virStorageVolPtr</a>	130
4.1.2.94	<a href="#">virStream</a>	130

4.1.2.95	<a href="#">virStreamEventCallback</a>	130
4.1.2.96	<a href="#">virStreamPtr</a>	131
4.1.2.97	<a href="#">virStreamSinkFunc</a>	131
4.1.2.98	<a href="#">virStreamSourceFunc</a>	131
4.1.2.99	<a href="#">virTypedParameter</a>	131
4.1.2.100	<a href="#">virTypedParameterPtr</a>	131
4.1.2.101	<a href="#">virVcpuInfo</a>	131
4.1.2.102	<a href="#">virVcpuInfoPtr</a>	131
4.1.3	Enumeration Type Documentation	132
4.1.3.1	<a href="#">virBlkioParameterType</a>	132
4.1.3.2	<a href="#">virConnectCloseReason</a>	132
4.1.3.3	<a href="#">virConnectCredentialType</a>	132
4.1.3.4	<a href="#">virConnectDomainEventBlockJobStatus</a>	132
4.1.3.5	<a href="#">virConnectDomainEventDiskChangeReason</a>	133
4.1.3.6	<a href="#">virConnectFlags</a>	133
4.1.3.7	<a href="#">virConnectListAllDomainsFlags</a>	133
4.1.3.8	<a href="#">virConnectListAllInterfacesFlags</a>	133
4.1.3.9	<a href="#">virConnectListAllNetworksFlags</a>	134
4.1.3.10	<a href="#">virConnectListAllNodeDeviceFlags</a>	134
4.1.3.11	<a href="#">virConnectListAllSecretsFlags</a>	134
4.1.3.12	<a href="#">virConnectListAllStoragePoolsFlags</a>	134
4.1.3.13	<a href="#">virCPUCompareResult</a>	135
4.1.3.14	<a href="#">virDomainBlockCommitFlags</a>	135
4.1.3.15	<a href="#">virDomainBlockedReason</a>	135
4.1.3.16	<a href="#">virDomainBlockJobAbortFlags</a>	135
4.1.3.17	<a href="#">virDomainBlockJobType</a>	135
4.1.3.18	<a href="#">virDomainBlockRebaseFlags</a>	136
4.1.3.19	<a href="#">virDomainBlockResizeFlags</a>	136
4.1.3.20	<a href="#">virDomainConsoleFlags</a>	136
4.1.3.21	<a href="#">virDomainControlState</a>	136
4.1.3.22	<a href="#">virDomainCoreDumpFlags</a>	137
4.1.3.23	<a href="#">virDomainCrashedReason</a>	137
4.1.3.24	<a href="#">virDomainCreateFlags</a>	137
4.1.3.25	<a href="#">virDomainDestroyFlagsValues</a>	137
4.1.3.26	<a href="#">virDomainDeviceModifyFlags</a>	137
4.1.3.27	<a href="#">virDomainDiskErrorCode</a>	138
4.1.3.28	<a href="#">virDomainEventDefinedDetailType</a>	138
4.1.3.29	<a href="#">virDomainEventGraphicsAddressType</a>	138
4.1.3.30	<a href="#">virDomainEventGraphicsPhase</a>	138
4.1.3.31	<a href="#">virDomainEventID</a>	138

4.1.3.32	<a href="#">virDomainEventIOErrorAction</a>	139
4.1.3.33	<a href="#">virDomainEventPMSuspendedDetailType</a>	139
4.1.3.34	<a href="#">virDomainEventResumedDetailType</a>	139
4.1.3.35	<a href="#">virDomainEventShutdownDetailType</a>	139
4.1.3.36	<a href="#">virDomainEventStartedDetailType</a>	140
4.1.3.37	<a href="#">virDomainEventStoppedDetailType</a>	140
4.1.3.38	<a href="#">virDomainEventSuspendedDetailType</a>	140
4.1.3.39	<a href="#">virDomainEventTrayChangeReason</a>	140
4.1.3.40	<a href="#">virDomainEventType</a>	141
4.1.3.41	<a href="#">virDomainEventUndefinedDetailType</a>	141
4.1.3.42	<a href="#">virDomainEventWatchdogAction</a>	141
4.1.3.43	<a href="#">virDomainJobType</a>	141
4.1.3.44	<a href="#">virDomainMemoryFlags</a>	142
4.1.3.45	<a href="#">virDomainMemoryModFlags</a>	142
4.1.3.46	<a href="#">virDomainMemoryStatTags</a>	142
4.1.3.47	<a href="#">virDomainMetadataType</a>	142
4.1.3.48	<a href="#">virDomainMigrateFlags</a>	142
4.1.3.49	<a href="#">virDomainModificationImpact</a>	143
4.1.3.50	<a href="#">virDomainNostateReason</a>	143
4.1.3.51	<a href="#">virDomainNumatuneMemMode</a>	143
4.1.3.52	<a href="#">virDomainOpenGraphicsFlags</a>	143
4.1.3.53	<a href="#">virDomainPausedReason</a>	143
4.1.3.54	<a href="#">virDomainPMSuspendedReason</a>	144
4.1.3.55	<a href="#">virDomainRebootFlagValues</a>	144
4.1.3.56	<a href="#">virDomainRunningReason</a>	144
4.1.3.57	<a href="#">virDomainSaveRestoreFlags</a>	144
4.1.3.58	<a href="#">virDomainShutdownFlagValues</a>	145
4.1.3.59	<a href="#">virDomainShutdownReason</a>	145
4.1.3.60	<a href="#">virDomainShutoffReason</a>	145
4.1.3.61	<a href="#">virDomainSnapshotCreateFlags</a>	145
4.1.3.62	<a href="#">virDomainSnapshotDeleteFlags</a>	145
4.1.3.63	<a href="#">virDomainSnapshotListFlags</a>	146
4.1.3.64	<a href="#">virDomainSnapshotRevertFlags</a>	146
4.1.3.65	<a href="#">virDomainState</a>	146
4.1.3.66	<a href="#">virDomainUndefineFlagsValues</a>	146
4.1.3.67	<a href="#">virDomainVcpuFlags</a>	147
4.1.3.68	<a href="#">virDomainXMLFlags</a>	147
4.1.3.69	<a href="#">virEventHandleType</a>	147
4.1.3.70	<a href="#">virInterfaceXMLFlags</a>	147
4.1.3.71	<a href="#">virKeycodeSet</a>	147

4.1.3.72	<a href="#">virMemoryParameterType</a>	148
4.1.3.73	<a href="#">virNetworkUpdateCommand</a>	148
4.1.3.74	<a href="#">virNetworkUpdateFlags</a>	148
4.1.3.75	<a href="#">virNetworkUpdateSection</a>	149
4.1.3.76	<a href="#">virNetworkXMLFlags</a>	149
4.1.3.77	<a href="#">virNodeGetCPUStatsAllCPUs</a>	149
4.1.3.78	<a href="#">virNodeGetMemoryStatsAllCells</a>	149
4.1.3.79	<a href="#">virNodeSuspendTarget</a>	150
4.1.3.80	<a href="#">virSchedParameterType</a>	150
4.1.3.81	<a href="#">virSecretUsageType</a>	150
4.1.3.82	<a href="#">virStoragePoolBuildFlags</a>	150
4.1.3.83	<a href="#">virStoragePoolDeleteFlags</a>	150
4.1.3.84	<a href="#">virStoragePoolState</a>	151
4.1.3.85	<a href="#">virStorageVolDeleteFlags</a>	151
4.1.3.86	<a href="#">virStorageVolResizeFlags</a>	151
4.1.3.87	<a href="#">virStorageVolType</a>	151
4.1.3.88	<a href="#">virStorageVolWipeAlgorithm</a>	151
4.1.3.89	<a href="#">virStorageXMLFlags</a>	152
4.1.3.90	<a href="#">virStreamEventType</a>	152
4.1.3.91	<a href="#">virStreamFlags</a>	152
4.1.3.92	<a href="#">virTypedParameterFlags</a>	152
4.1.3.93	<a href="#">virTypedParameterType</a>	152
4.1.3.94	<a href="#">virVcpuState</a>	153
4.1.4	<a href="#">Function Documentation</a>	153
4.1.4.1	<a href="#">virConnectBaselineCPU</a>	153
4.1.4.2	<a href="#">virConnectClose</a>	153
4.1.4.3	<a href="#">virConnectCompareCPU</a>	153
4.1.4.4	<a href="#">virConnectDomainEventDeregister</a>	154
4.1.4.5	<a href="#">virConnectDomainEventDeregisterAny</a>	154
4.1.4.6	<a href="#">virConnectDomainEventRegister</a>	154
4.1.4.7	<a href="#">virConnectDomainEventRegisterAny</a>	154
4.1.4.8	<a href="#">virConnectDomainXMLFromNative</a>	155
4.1.4.9	<a href="#">virConnectDomainXMLToNative</a>	155
4.1.4.10	<a href="#">virConnectFindStoragePoolSources</a>	155
4.1.4.11	<a href="#">virConnectGetCapabilities</a>	155
4.1.4.12	<a href="#">virConnectGetHostname</a>	155
4.1.4.13	<a href="#">virConnectGetLibVersion</a>	156
4.1.4.14	<a href="#">virConnectGetMaxVcpus</a>	156
4.1.4.15	<a href="#">virConnectGetSysinfo</a>	156
4.1.4.16	<a href="#">virConnectGetType</a>	156



4.1.4.17	<a href="#">virConnectGetURI</a>	156
4.1.4.18	<a href="#">virConnectGetVersion</a>	156
4.1.4.19	<a href="#">virConnectIsAlive</a>	157
4.1.4.20	<a href="#">virConnectIsEncrypted</a>	157
4.1.4.21	<a href="#">virConnectIsSecure</a>	157
4.1.4.22	<a href="#">virConnectListAllDomains</a>	157
4.1.4.23	<a href="#">virConnectListAllInterfaces</a>	158
4.1.4.24	<a href="#">virConnectListAllNetworks</a>	158
4.1.4.25	<a href="#">virConnectListAllNodeDevices</a>	159
4.1.4.26	<a href="#">virConnectListAllNWFilters</a>	159
4.1.4.27	<a href="#">virConnectListAllSecrets</a>	159
4.1.4.28	<a href="#">virConnectListAllStoragePools</a>	160
4.1.4.29	<a href="#">virConnectListDefinedDomains</a>	160
4.1.4.30	<a href="#">virConnectListDefinedInterfaces</a>	160
4.1.4.31	<a href="#">virConnectListDefinedNetworks</a>	161
4.1.4.32	<a href="#">virConnectListDefinedStoragePools</a>	161
4.1.4.33	<a href="#">virConnectListDomains</a>	161
4.1.4.34	<a href="#">virConnectListInterfaces</a>	161
4.1.4.35	<a href="#">virConnectListNetworks</a>	162
4.1.4.36	<a href="#">virConnectListNWFilters</a>	162
4.1.4.37	<a href="#">virConnectListSecrets</a>	162
4.1.4.38	<a href="#">virConnectListStoragePools</a>	162
4.1.4.39	<a href="#">virConnectNumOfDefinedDomains</a>	162
4.1.4.40	<a href="#">virConnectNumOfDefinedInterfaces</a>	162
4.1.4.41	<a href="#">virConnectNumOfDefinedNetworks</a>	163
4.1.4.42	<a href="#">virConnectNumOfDefinedStoragePools</a>	163
4.1.4.43	<a href="#">virConnectNumOfDomains</a>	163
4.1.4.44	<a href="#">virConnectNumOfInterfaces</a>	163
4.1.4.45	<a href="#">virConnectNumOfNetworks</a>	163
4.1.4.46	<a href="#">virConnectNumOfNWFilters</a>	163
4.1.4.47	<a href="#">virConnectNumOfSecrets</a>	163
4.1.4.48	<a href="#">virConnectNumOfStoragePools</a>	163
4.1.4.49	<a href="#">virConnectOpen</a>	164
4.1.4.50	<a href="#">virConnectOpenAuth</a>	164
4.1.4.51	<a href="#">virConnectOpenReadOnly</a>	164
4.1.4.52	<a href="#">virConnectRef</a>	164
4.1.4.53	<a href="#">virConnectRegisterCloseCallback</a>	164
4.1.4.54	<a href="#">virConnectSetKeepAlive</a>	164
4.1.4.55	<a href="#">virConnectUnregisterCloseCallback</a>	164
4.1.4.56	<a href="#">virDomainAbortJob</a>	165

4.1.4.57	<a href="#">virDomainAttachDevice</a>	165
4.1.4.58	<a href="#">virDomainAttachDeviceFlags</a>	165
4.1.4.59	<a href="#">virDomainBlockCommit</a>	165
4.1.4.60	<a href="#">virDomainBlockJobAbort</a>	166
4.1.4.61	<a href="#">virDomainBlockJobSetSpeed</a>	167
4.1.4.62	<a href="#">virDomainBlockPeek</a>	167
4.1.4.63	<a href="#">virDomainBlockPull</a>	167
4.1.4.64	<a href="#">virDomainBlockRebase</a>	168
4.1.4.65	<a href="#">virDomainBlockResize</a>	169
4.1.4.66	<a href="#">virDomainBlockStats</a>	169
4.1.4.67	<a href="#">virDomainBlockStatsFlags</a>	169
4.1.4.68	<a href="#">virDomainCoreDump</a>	170
4.1.4.69	<a href="#">virDomainCreate</a>	170
4.1.4.70	<a href="#">virDomainCreateLinux</a>	170
4.1.4.71	<a href="#">virDomainCreateWithFlags</a>	170
4.1.4.72	<a href="#">virDomainCreateXML</a>	171
4.1.4.73	<a href="#">virDomainDefineXML</a>	171
4.1.4.74	<a href="#">virDomainDestroy</a>	171
4.1.4.75	<a href="#">virDomainDestroyFlags</a>	171
4.1.4.76	<a href="#">virDomainDetachDevice</a>	172
4.1.4.77	<a href="#">virDomainDetachDeviceFlags</a>	172
4.1.4.78	<a href="#">virDomainFree</a>	172
4.1.4.79	<a href="#">virDomainGetAutostart</a>	172
4.1.4.80	<a href="#">virDomainGetBlkioParameters</a>	172
4.1.4.81	<a href="#">virDomainGetBlockInfo</a>	173
4.1.4.82	<a href="#">virDomainGetBlockIoTune</a>	173
4.1.4.83	<a href="#">virDomainGetBlockJobInfo</a>	173
4.1.4.84	<a href="#">virDomainGetConnect</a>	174
4.1.4.85	<a href="#">virDomainGetControllInfo</a>	174
4.1.4.86	<a href="#">virDomainGetCPUStats</a>	174
4.1.4.87	<a href="#">virDomainGetDiskErrors</a>	175
4.1.4.88	<a href="#">virDomainGetEmulatorPinInfo</a>	175
4.1.4.89	<a href="#">virDomainGetHostname</a>	175
4.1.4.90	<a href="#">virDomainGetID</a>	175
4.1.4.91	<a href="#">virDomainGetInfo</a>	175
4.1.4.92	<a href="#">virDomainGetInterfaceParameters</a>	176
4.1.4.93	<a href="#">virDomainGetJobInfo</a>	176
4.1.4.94	<a href="#">virDomainGetMaxMemory</a>	176
4.1.4.95	<a href="#">virDomainGetMaxVcpus</a>	176
4.1.4.96	<a href="#">virDomainGetMemoryParameters</a>	176

4.1.4.97	<a href="#">virDomainGetMetadata</a>	177
4.1.4.98	<a href="#">virDomainGetName</a>	177
4.1.4.99	<a href="#">virDomainGetNumaParameters</a>	177
4.1.4.100	<a href="#">virDomainGetOSType</a>	177
4.1.4.101	<a href="#">virDomainGetSchedulerParameters</a>	178
4.1.4.102	<a href="#">virDomainGetSchedulerParametersFlags</a>	178
4.1.4.103	<a href="#">virDomainGetSchedulerType</a>	178
4.1.4.104	<a href="#">virDomainGetSecurityLabel</a>	178
4.1.4.105	<a href="#">virDomainGetSecurityLabelList</a>	178
4.1.4.106	<a href="#">virDomainGetState</a>	179
4.1.4.107	<a href="#">virDomainGetUUID</a>	179
4.1.4.108	<a href="#">virDomainGetUUIDString</a>	179
4.1.4.109	<a href="#">virDomainGetVcpuPinInfo</a>	179
4.1.4.110	<a href="#">virDomainGetVcpus</a>	179
4.1.4.111	<a href="#">virDomainGetVcpusFlags</a>	180
4.1.4.112	<a href="#">virDomainGetXMLDesc</a>	180
4.1.4.113	<a href="#">virDomainHasCurrentSnapshot</a>	180
4.1.4.114	<a href="#">virDomainHasManagedSaveImage</a>	180
4.1.4.115	<a href="#">virDomainInjectNMI</a>	180
4.1.4.116	<a href="#">virDomainInterfaceStats</a>	181
4.1.4.117	<a href="#">virDomainIsActive</a>	181
4.1.4.118	<a href="#">virDomainIsPersistent</a>	181
4.1.4.119	<a href="#">virDomainIsUpdated</a>	181
4.1.4.120	<a href="#">virDomainListAllSnapshots</a>	181
4.1.4.121	<a href="#">virDomainLookupByID</a>	182
4.1.4.122	<a href="#">virDomainLookupByName</a>	182
4.1.4.123	<a href="#">virDomainLookupByUUID</a>	182
4.1.4.124	<a href="#">virDomainLookupByUUIDString</a>	182
4.1.4.125	<a href="#">virDomainManagedSave</a>	182
4.1.4.126	<a href="#">virDomainManagedSaveRemove</a>	183
4.1.4.127	<a href="#">virDomainMemoryPeek</a>	183
4.1.4.128	<a href="#">virDomainMemoryStats</a>	183
4.1.4.129	<a href="#">virDomainMigrate</a>	183
4.1.4.130	<a href="#">virDomainMigrate2</a>	184
4.1.4.131	<a href="#">virDomainMigrateGetMaxSpeed</a>	185
4.1.4.132	<a href="#">virDomainMigrateSetMaxDowntime</a>	185
4.1.4.133	<a href="#">virDomainMigrateSetMaxSpeed</a>	185
4.1.4.134	<a href="#">virDomainMigrateToURI</a>	186
4.1.4.135	<a href="#">virDomainMigrateToURI2</a>	186
4.1.4.136	<a href="#">virDomainOpenConsole</a>	187

4.1.4.137 virDomainOpenGraphics . . . . .	187
4.1.4.138 virDomainPinEmulator . . . . .	188
4.1.4.139 virDomainPinVcpu . . . . .	188
4.1.4.140 virDomainPinVcpuFlags . . . . .	188
4.1.4.141 virDomainPMSuspendForDuration . . . . .	189
4.1.4.142 virDomainPMWakeup . . . . .	189
4.1.4.143 virDomainReboot . . . . .	189
4.1.4.144 virDomainRef . . . . .	189
4.1.4.145 virDomainReset . . . . .	189
4.1.4.146 virDomainRestore . . . . .	190
4.1.4.147 virDomainRestoreFlags . . . . .	190
4.1.4.148 virDomainResume . . . . .	190
4.1.4.149 virDomainRevertToSnapshot . . . . .	190
4.1.4.150 virDomainSave . . . . .	191
4.1.4.151 virDomainSaveFlags . . . . .	191
4.1.4.152 virDomainSaveImageDefineXML . . . . .	191
4.1.4.153 virDomainSaveImageGetXMLDesc . . . . .	191
4.1.4.154 virDomainScreenshot . . . . .	191
4.1.4.155 virDomainSendKey . . . . .	192
4.1.4.156 virDomainSetAutostart . . . . .	192
4.1.4.157 virDomainSetBlkioParameters . . . . .	192
4.1.4.158 virDomainSetBlockIoTune . . . . .	192
4.1.4.159 virDomainSetInterfaceParameters . . . . .	192
4.1.4.160 virDomainSetMaxMemory . . . . .	193
4.1.4.161 virDomainSetMemory . . . . .	193
4.1.4.162 virDomainSetMemoryFlags . . . . .	193
4.1.4.163 virDomainSetMemoryParameters . . . . .	193
4.1.4.164 virDomainSetMetadata . . . . .	194
4.1.4.165 virDomainSetNumaParameters . . . . .	194
4.1.4.166 virDomainSetSchedulerParameters . . . . .	194
4.1.4.167 virDomainSetSchedulerParametersFlags . . . . .	194
4.1.4.168 virDomainSetVcpus . . . . .	195
4.1.4.169 virDomainSetVcpusFlags . . . . .	195
4.1.4.170 virDomainShutdown . . . . .	195
4.1.4.171 virDomainShutdownFlags . . . . .	195
4.1.4.172 virDomainSnapshotCreateXML . . . . .	196
4.1.4.173 virDomainSnapshotCurrent . . . . .	197
4.1.4.174 virDomainSnapshotDelete . . . . .	197
4.1.4.175 virDomainSnapshotFree . . . . .	197
4.1.4.176 virDomainSnapshotGetConnect . . . . .	197

4.1.4.177 virDomainSnapshotGetDomain . . . . .	197
4.1.4.178 virDomainSnapshotGetName . . . . .	198
4.1.4.179 virDomainSnapshotGetParent . . . . .	198
4.1.4.180 virDomainSnapshotGetXMLDesc . . . . .	198
4.1.4.181 virDomainSnapshotHasMetadata . . . . .	198
4.1.4.182 virDomainSnapshotIsCurrent . . . . .	198
4.1.4.183 virDomainSnapshotListAllChildren . . . . .	198
4.1.4.184 virDomainSnapshotListChildrenNames . . . . .	199
4.1.4.185 virDomainSnapshotListNames . . . . .	199
4.1.4.186 virDomainSnapshotLookupByName . . . . .	200
4.1.4.187 virDomainSnapshotNum . . . . .	200
4.1.4.188 virDomainSnapshotNumChildren . . . . .	200
4.1.4.189 virDomainSnapshotRef . . . . .	201
4.1.4.190 virDomainSuspend . . . . .	201
4.1.4.191 virDomainUndefine . . . . .	201
4.1.4.192 virDomainUndefineFlags . . . . .	201
4.1.4.193 virDomainUpdateDeviceFlags . . . . .	202
4.1.4.194 virEventAddHandle . . . . .	202
4.1.4.195 virEventAddTimeout . . . . .	202
4.1.4.196 virEventRegisterDefaultImpl . . . . .	202
4.1.4.197 virEventRegisterImpl . . . . .	202
4.1.4.198 virEventRemoveHandle . . . . .	202
4.1.4.199 virEventRemoveTimeout . . . . .	202
4.1.4.200 virEventRunDefaultImpl . . . . .	202
4.1.4.201 virEventUpdateHandle . . . . .	202
4.1.4.202 virEventUpdateTimeout . . . . .	202
4.1.4.203 virGetVersion . . . . .	202
4.1.4.204 virInitialize . . . . .	202
4.1.4.205 virInterfaceChangeBegin . . . . .	202
4.1.4.206 virInterfaceChangeCommit . . . . .	203
4.1.4.207 virInterfaceChangeRollback . . . . .	203
4.1.4.208 virInterfaceCreate . . . . .	203
4.1.4.209 virInterfaceDefineXML . . . . .	203
4.1.4.210 virInterfaceDestroy . . . . .	203
4.1.4.211 virInterfaceFree . . . . .	204
4.1.4.212 virInterfaceGetConnect . . . . .	204
4.1.4.213 virInterfaceGetMACString . . . . .	204
4.1.4.214 virInterfaceGetName . . . . .	204
4.1.4.215 virInterfaceGetXMLDesc . . . . .	204
4.1.4.216 virInterfacelsActive . . . . .	205

4.1.4.217 virInterfaceLookupByMACString . . . . .	205
4.1.4.218 virInterfaceLookupByName . . . . .	205
4.1.4.219 virInterfaceRef . . . . .	205
4.1.4.220 virInterfaceUndefine . . . . .	205
4.1.4.221 virNetworkCreate . . . . .	205
4.1.4.222 virNetworkCreateXML . . . . .	205
4.1.4.223 virNetworkDefineXML . . . . .	206
4.1.4.224 virNetworkDestroy . . . . .	206
4.1.4.225 virNetworkFree . . . . .	206
4.1.4.226 virNetworkGetAutostart . . . . .	206
4.1.4.227 virNetworkGetBridgeName . . . . .	206
4.1.4.228 <b>POL New</b> virNetworkGetBridgeType . . . . .	206
4.1.4.229 virNetworkGetConnect . . . . .	207
4.1.4.230 virNetworkGetName . . . . .	207
4.1.4.231 virNetworkGetUUID . . . . .	207
4.1.4.232 virNetworkGetUUIDString . . . . .	207
4.1.4.233 virNetworkGetXMLDesc . . . . .	207
4.1.4.234 virNetworkIsActive . . . . .	207
4.1.4.235 virNetworkIsPersistent . . . . .	207
4.1.4.236 virNetworkLookupByName . . . . .	208
4.1.4.237 virNetworkLookupByUUID . . . . .	208
4.1.4.238 virNetworkLookupByUUIDString . . . . .	208
4.1.4.239 virNetworkRef . . . . .	208
4.1.4.240 virNetworkSetAutostart . . . . .	208
4.1.4.241 virNetworkUndefine . . . . .	208
4.1.4.242 virNetworkUpdate . . . . .	208
4.1.4.243 virNodeDeviceCreateXML . . . . .	209
4.1.4.244 virNodeDeviceDestroy . . . . .	209
4.1.4.245 virNodeDeviceDetach . . . . .	209
4.1.4.246 virNodeDeviceFree . . . . .	209
4.1.4.247 virNodeDeviceGetName . . . . .	209
4.1.4.248 virNodeDeviceGetParent . . . . .	209
4.1.4.249 virNodeDeviceGetXMLDesc . . . . .	209
4.1.4.250 virNodeDeviceListCaps . . . . .	210
4.1.4.251 virNodeDeviceLookupByName . . . . .	210
4.1.4.252 virNodeDeviceNumOfCaps . . . . .	210
4.1.4.253 virNodeDeviceReAttach . . . . .	210
4.1.4.254 virNodeDeviceRef . . . . .	210
4.1.4.255 virNodeDeviceReset . . . . .	210
4.1.4.256 virNodeGetCellsFreeMemory . . . . .	211

4.1.4.257 virNodeGetCPUStats . . . . .	211
4.1.4.258 virNodeGetFreeMemory . . . . .	211
4.1.4.259 virNodeGetInfo . . . . .	211
4.1.4.260 virNodeGetMemoryParameters . . . . .	212
4.1.4.261 virNodeGetMemoryStats . . . . .	212
4.1.4.262 virNodeGetSecurityModel . . . . .	212
4.1.4.263 virNodeListDevices . . . . .	212
4.1.4.264 virNodeNumOfDevices . . . . .	213
4.1.4.265 virNodeSetMemoryParameters . . . . .	213
4.1.4.266 virNodeSuspendForDuration . . . . .	213
4.1.4.267 virNWFilterDefineXML . . . . .	213
4.1.4.268 virNWFilterFree . . . . .	213
4.1.4.269 virNWFilterGetName . . . . .	213
4.1.4.270 virNWFilterGetUUID . . . . .	213
4.1.4.271 virNWFilterGetUUIDString . . . . .	214
4.1.4.272 virNWFilterGetXMLDesc . . . . .	214
4.1.4.273 virNWFilterLookupByName . . . . .	214
4.1.4.274 virNWFilterLookupByUUID . . . . .	214
4.1.4.275 virNWFilterLookupByUUIDString . . . . .	214
4.1.4.276 virNWFilterRef . . . . .	214
4.1.4.277 virNWFilterUndefine . . . . .	214
4.1.4.278 virSecretDefineXML . . . . .	215
4.1.4.279 virSecretFree . . . . .	215
4.1.4.280 virSecretGetConnect . . . . .	215
4.1.4.281 virSecretGetUsageID . . . . .	215
4.1.4.282 virSecretGetUsageType . . . . .	215
4.1.4.283 virSecretGetUUID . . . . .	215
4.1.4.284 virSecretGetUUIDString . . . . .	216
4.1.4.285 virSecretGetValue . . . . .	216
4.1.4.286 virSecretGetXMLDesc . . . . .	216
4.1.4.287 virSecretLookupByUsage . . . . .	216
4.1.4.288 virSecretLookupByUUID . . . . .	216
4.1.4.289 virSecretLookupByUUIDString . . . . .	216
4.1.4.290 virSecretRef . . . . .	216
4.1.4.291 virSecretSetValue . . . . .	217
4.1.4.292 virSecretUndefine . . . . .	217
4.1.4.293 virStoragePoolBuild . . . . .	217
4.1.4.294 virStoragePoolCreate . . . . .	217
4.1.4.295 virStoragePoolCreateXML . . . . .	217
4.1.4.296 virStoragePoolDefineXML . . . . .	217

4.1.4.297 virStoragePoolDelete . . . . .	218
4.1.4.298 virStoragePoolDestroy . . . . .	218
4.1.4.299 virStoragePoolFree . . . . .	218
4.1.4.300 virStoragePoolGetAutostart . . . . .	218
4.1.4.301 virStoragePoolGetConnect . . . . .	218
4.1.4.302 virStoragePoolGetInfo . . . . .	218
4.1.4.303 virStoragePoolGetName . . . . .	219
4.1.4.304 virStoragePoolGetUUID . . . . .	219
4.1.4.305 virStoragePoolGetUUIDString . . . . .	219
4.1.4.306 virStoragePoolGetXMLDesc . . . . .	219
4.1.4.307 virStoragePoolsActive . . . . .	219
4.1.4.308 virStoragePoolsPersistent . . . . .	219
4.1.4.309 virStoragePoolListAllVolumes . . . . .	219
4.1.4.310 virStoragePoolListVolumes . . . . .	220
4.1.4.311 virStoragePoolLookupByName . . . . .	220
4.1.4.312 virStoragePoolLookupByUUID . . . . .	220
4.1.4.313 virStoragePoolLookupByUUIDString . . . . .	220
4.1.4.314 virStoragePoolLookupByVolume . . . . .	220
4.1.4.315 virStoragePoolNumOfVolumes . . . . .	220
4.1.4.316 virStoragePoolRef . . . . .	220
4.1.4.317 virStoragePoolRefresh . . . . .	220
4.1.4.318 virStoragePoolSetAutostart . . . . .	221
4.1.4.319 virStoragePoolUndefine . . . . .	221
4.1.4.320 virStorageVolCreateXML . . . . .	221
4.1.4.321 virStorageVolCreateXMLFrom . . . . .	221
4.1.4.322 virStorageVolDelete . . . . .	221
4.1.4.323 virStorageVolDownload . . . . .	221
4.1.4.324 virStorageVolFree . . . . .	222
4.1.4.325 virStorageVolGetConnect . . . . .	222
4.1.4.326 virStorageVolGetInfo . . . . .	222
4.1.4.327 virStorageVolGetKey . . . . .	222
4.1.4.328 virStorageVolGetName . . . . .	222
4.1.4.329 virStorageVolGetPath . . . . .	222
4.1.4.330 virStorageVolGetXMLDesc . . . . .	223
4.1.4.331 virStorageVolLookupByKey . . . . .	223
4.1.4.332 virStorageVolLookupByName . . . . .	223
4.1.4.333 virStorageVolLookupByPath . . . . .	223
4.1.4.334 virStorageVolRef . . . . .	223
4.1.4.335 virStorageVolResize . . . . .	223
4.1.4.336 virStorageVolUpload . . . . .	224



4.1.4.337	virStorageVolWipe	224
4.1.4.338	virStorageVolWipePattern	224
4.1.4.339	virStreamAbort	224
4.1.4.340	virStreamEventAddCallback	224
4.1.4.341	virStreamEventRemoveCallback	224
4.1.4.342	virStreamEventUpdateCallback	225
4.1.4.343	virStreamFinish	225
4.1.4.344	virStreamFree	225
4.1.4.345	virStreamNew	225
4.1.4.346	virStreamRecv	225
4.1.4.347	virStreamRecvAll	226
4.1.4.348	virStreamRef	226
4.1.4.349	virStreamSend	226
4.1.4.350	virStreamSendAll	227
4.1.5	Variable Documentation	227
4.1.5.1	virConnectAuthPtrDefault	227
4.2	src/conf/domain_conf.c File Reference	227
4.2.1	Macro Definition Documentation	235
4.2.1.1	DUMPTXML_FLAGS	235
4.2.1.2	MATCH	236
4.2.1.3	NET_MODEL_CHARS	236
4.2.1.4	VIR_DOMAIN_BLOCKED_LAST	236
4.2.1.5	VIR_DOMAIN_CRASHED_LAST	236
4.2.1.6	VIR_DOMAIN_NOSTATE_LAST	236
4.2.1.7	VIR_DOMAIN_PAUSED_LAST	236
4.2.1.8	VIR_DOMAIN_RUNNING_LAST	236
4.2.1.9	VIR_DOMAIN_SHUTDOWN_LAST	236
4.2.1.10	VIR_DOMAIN_SHUTOFF_LAST	236
4.2.1.11	VIR_DOMAIN_XML_READ_FLAGS	236
4.2.1.12	VIR_DOMAIN_XML_WRITE_FLAGS	236
4.2.1.13	VIR_FROM_THIS	236
4.2.2	Enumeration Type Documentation	236
4.2.2.1	virDomainXMLInternalFlags	236
4.2.3	Function Documentation	236
4.2.3.1	ATTRIBUTE_NONNULL	236
4.2.3.2	ATTRIBUTE_NONNULL	236
4.2.3.3	verify	236
4.2.3.4	verify	236
4.2.3.5	VIR_ENUM_IMPL	236
4.2.3.6	VIR_ENUM_IMPL	236

4.2.3.7	<a href="#">virBlkioDeviceWeightArrayClear</a>	237
4.2.3.8	<a href="#">virDiskNameToBusDeviceIndex</a>	237
4.2.3.9	<a href="#">POL Mod virDomainActualNetDefFormat</a>	237
4.2.3.10	<a href="#">POL Mod virDomainActualNetDefFree</a>	237
4.2.3.11	<a href="#">virDomainActualNetDefParseXML</a>	237
4.2.3.12	<a href="#">virDomainAssignDef</a>	237
4.2.3.13	<a href="#">virDomainBlkioDeviceWeightParseXML</a>	237
4.2.3.14	<a href="#">virDomainChannelDefCheckABIStability</a>	237
4.2.3.15	<a href="#">virDomainChrDefaultTargetType</a>	237
4.2.3.16	<a href="#">virDomainChrDefForeach</a>	238
4.2.3.17	<a href="#">virDomainChrDefFormat</a>	238
4.2.3.18	<a href="#">virDomainChrDefFree</a>	238
4.2.3.19	<a href="#">virDomainChrDefGetSecurityLabelDef</a>	238
4.2.3.20	<a href="#">virDomainChrDefNew</a>	238
4.2.3.21	<a href="#">virDomainChrDefParseTargetXML</a>	238
4.2.3.22	<a href="#">virDomainChrDefParseXML</a>	238
4.2.3.23	<a href="#">virDomainChrSourceDefCopy</a>	238
4.2.3.24	<a href="#">virDomainChrSourceDefFormat</a>	238
4.2.3.25	<a href="#">virDomainChrSourceDefFree</a>	238
4.2.3.26	<a href="#">virDomainChrSourceDefIsEqual</a>	238
4.2.3.27	<a href="#">virDomainChrSourceDefParseXML</a>	238
4.2.3.28	<a href="#">virDomainChrTargetTypeFromString</a>	238
4.2.3.29	<a href="#">virDomainChrTargetTypeToString</a>	238
4.2.3.30	<a href="#">virDomainClockDefClear</a>	238
4.2.3.31	<a href="#">virDomainConfigFile</a>	238
4.2.3.32	<a href="#">virDomainConsoleDefCheckABIStability</a>	238
4.2.3.33	<a href="#">virDomainControllerDefCheckABIStability</a>	238
4.2.3.34	<a href="#">virDomainControllerDefFormat</a>	238
4.2.3.35	<a href="#">virDomainControllerDefFree</a>	238
4.2.3.36	<a href="#">virDomainControllerDefParseXML</a>	238
4.2.3.37	<a href="#">virDomainControllerFind</a>	239
4.2.3.38	<a href="#">virDomainControllerInsert</a>	239
4.2.3.39	<a href="#">virDomainControllerInsertPreAlloced</a>	239
4.2.3.40	<a href="#">virDomainControllerModelTypeFromString</a>	239
4.2.3.41	<a href="#">virDomainControllerModelTypeToString</a>	239
4.2.3.42	<a href="#">virDomainControllerRemove</a>	239
4.2.3.43	<a href="#">virDomainDefAddDiskControllersForType</a>	239
4.2.3.44	<a href="#">virDomainDefAddImplicitControllers</a>	239
4.2.3.45	<a href="#">virDomainDefAddSecurityLabelDef</a>	239
4.2.3.46	<a href="#">virDomainDefCheckABIStability</a>	239

4.2.3.47	<a href="#">virDomainDefClearDeviceAliases</a>	239
4.2.3.48	<a href="#">virDomainDefClearPCIAddresses</a>	239
4.2.3.49	<a href="#">virDomainDefCompatibleDevice</a>	239
4.2.3.50	<a href="#">virDomainDefDefaultEmulator</a>	239
4.2.3.51	<a href="#">virDomainDefFormat</a>	239
4.2.3.52	<a href="#">virDomainDefFormatInternal</a>	239
4.2.3.53	<a href="#">virDomainDefFree</a>	239
4.2.3.54	<a href="#">virDomainDefGetSecurityLabelDef</a>	239
4.2.3.55	<a href="#">virDomainDefHasUSB</a>	239
4.2.3.56	<a href="#">virDomainDefMaybeAddController</a>	239
4.2.3.57	<a href="#">virDomainDefMaybeAddSmartcardController</a>	239
4.2.3.58	<a href="#">virDomainDefMaybeAddVirtioSerialController</a>	239
4.2.3.59	<a href="#">virDomainDefParse</a>	239
4.2.3.60	<a href="#">virDomainDefParseBootXML</a>	239
4.2.3.61	<a href="#">virDomainDefParseFile</a>	239
4.2.3.62	<a href="#">virDomainDefParseNode</a>	240
4.2.3.63	<a href="#">virDomainDefParseString</a>	240
4.2.3.64	<a href="#">virDomainDefParseXML</a>	240
4.2.3.65	<a href="#">virDomainDeleteConfig</a>	240
4.2.3.66	<a href="#">virDomainDeviceAddressIsValid</a>	240
4.2.3.67	<a href="#">virDomainDeviceBootParseXML</a>	240
4.2.3.68	<a href="#">virDomainDeviceCcidAddressParseXML</a>	240
4.2.3.69	<a href="#">virDomainDeviceDefCopy</a>	240
4.2.3.70	<a href="#">virDomainDeviceDefFree</a>	240
4.2.3.71	<a href="#">virDomainDeviceDefParse</a>	240
4.2.3.72	<a href="#">virDomainDeviceDriveAddressParseXML</a>	240
4.2.3.73	<a href="#">virDomainDeviceInfoCheckABIStability</a>	240
4.2.3.74	<a href="#">virDomainDeviceInfoClear</a>	240
4.2.3.75	<a href="#">virDomainDeviceInfoClearAlias</a>	240
4.2.3.76	<a href="#">virDomainDeviceInfoClearPCIAddress</a>	240
4.2.3.77	<a href="#">virDomainDeviceInfoFree</a>	240
4.2.3.78	<a href="#">virDomainDeviceInfoIsSet</a>	240
4.2.3.79	<a href="#">virDomainDeviceInfoIterate</a>	240
4.2.3.80	<a href="#">virDomainDeviceInfoParseXML</a>	240
4.2.3.81	<a href="#">virDomainDevicesIsUSB</a>	240
4.2.3.82	<a href="#">virDomainDeviceSpaprVioAddressParseXML</a>	240
4.2.3.83	<a href="#">virDomainDeviceUSBAddressParseXML</a>	241
4.2.3.84	<a href="#">virDomainDeviceUSBMasterParseXML</a>	241
4.2.3.85	<a href="#">virDomainDeviceVirtioSerialAddressParseXML</a>	241
4.2.3.86	<a href="#">virDomainDiskBlockioDefFormat</a>	241

4.2.3.87	<a href="#">virDomainDiskDefAssignAddress</a>	241
4.2.3.88	<a href="#">virDomainDiskDefCheckABIStability</a>	241
4.2.3.89	<a href="#">virDomainDiskDefForeachPath</a>	241
4.2.3.90	<a href="#">virDomainDiskDefFormat</a>	241
4.2.3.91	<a href="#">virDomainDiskDefFree</a>	241
4.2.3.92	<a href="#">virDomainDiskDefGetSecurityLabelDef</a>	241
4.2.3.93	<a href="#">virDomainDiskDefParseXML</a>	241
4.2.3.94	<a href="#">virDomainDiskFindControllerModel</a>	241
4.2.3.95	<a href="#">virDomainDiskGeometryDefFormat</a>	241
4.2.3.96	<a href="#">virDomainDiskHostDefFree</a>	241
4.2.3.97	<a href="#">virDomainDiskIndexByName</a>	241
4.2.3.98	<a href="#">virDomainDiskInsert</a>	241
4.2.3.99	<a href="#">virDomainDiskInsertPreAlloced</a>	241
4.2.3.100	<a href="#">virDomainDiskPathByName</a>	241
4.2.3.101	<a href="#">virDomainDiskRemove</a>	241
4.2.3.102	<a href="#">virDomainDiskRemoveByName</a>	241
4.2.3.103	<a href="#">virDomainEmulatorPinAdd</a>	241
4.2.3.104	<a href="#">virDomainEmulatorPinDel</a>	241
4.2.3.105	<a href="#">virDomainFindByID</a>	241
4.2.3.106	<a href="#">virDomainFindByName</a>	241
4.2.3.107	<a href="#">virDomainFindByUUID</a>	242
4.2.3.108	<a href="#">virDomainFsDefCheckABIStability</a>	242
4.2.3.109	<a href="#">virDomainFSDefFormat</a>	242
4.2.3.110	<a href="#">virDomainFSDefFree</a>	242
4.2.3.111	<a href="#">virDomainFSDefParseXML</a>	242
4.2.3.112	<a href="#">virDomainFSIndexByName</a>	242
4.2.3.113	<a href="#">virDomainGetRootFilesystem</a>	242
4.2.3.114	<a href="#">virDomainGraphicsAuthDefClear</a>	242
4.2.3.115	<a href="#">virDomainGraphicsAuthDefFormatAttr</a>	242
4.2.3.116	<a href="#">virDomainGraphicsAuthDefParseXML</a>	242
4.2.3.117	<a href="#">virDomainGraphicsDefFormat</a>	242
4.2.3.118	<a href="#">virDomainGraphicsDefFree</a>	242
4.2.3.119	<a href="#">virDomainGraphicsDefParseXML</a>	242
4.2.3.120	<a href="#">virDomainGraphicsGetListen</a>	242
4.2.3.121	<a href="#">virDomainGraphicsListenDefClear</a>	242
4.2.3.122	<a href="#">virDomainGraphicsListenDefFormat</a>	242
4.2.3.123	<a href="#">virDomainGraphicsListenDefParseXML</a>	242
4.2.3.124	<a href="#">virDomainGraphicsListenGetAddress</a>	242
4.2.3.125	<a href="#">virDomainGraphicsListenGetNetwork</a>	242
4.2.3.126	<a href="#">virDomainGraphicsListenGetType</a>	242

4.2.3.127 virDomainGraphicsListenSetAddress . . . . .	242
4.2.3.128 virDomainGraphicsListenSetNetwork . . . . .	242
4.2.3.129 virDomainGraphicsListenSetType . . . . .	242
4.2.3.130 virDomainHostdevDefAlloc . . . . .	243
4.2.3.131 virDomainHostdevDefCheckABIStability . . . . .	243
4.2.3.132 virDomainHostdevDefClear . . . . .	243
4.2.3.133 virDomainHostdevDefFormat . . . . .	243
4.2.3.134 virDomainHostdevDefFree . . . . .	243
4.2.3.135 virDomainHostdevDefParseXML . . . . .	243
4.2.3.136 virDomainHostdevFind . . . . .	243
4.2.3.137 virDomainHostdevInsert . . . . .	243
4.2.3.138 virDomainHostdevPartsParse . . . . .	243
4.2.3.139 virDomainHostdevRemove . . . . .	243
4.2.3.140 virDomainHostdevSourceFormat . . . . .	243
4.2.3.141 virDomainHostdevSubsysPciDefParseXML . . . . .	243
4.2.3.142 virDomainHostdevSubsysPciOrigStatesDefParseXML . . . . .	243
4.2.3.143 virDomainHostdevSubsysUsbDefParseXML . . . . .	243
4.2.3.144 virDomainHubDefCheckABIStability . . . . .	243
4.2.3.145 virDomainHubDefFormat . . . . .	243
4.2.3.146 virDomainHubDefFree . . . . .	243
4.2.3.147 virDomainHubDefParseXML . . . . .	243
4.2.3.148 virDomainInputDefCheckABIStability . . . . .	243
4.2.3.149 virDomainInputDefFormat . . . . .	243
4.2.3.150 virDomainInputDefFree . . . . .	243
4.2.3.151 virDomainInputDefParseXML . . . . .	244
4.2.3.152 virDomainLeaseDefFormat . . . . .	244
4.2.3.153 virDomainLeaseDefFree . . . . .	244
4.2.3.154 virDomainLeaseDefParseXML . . . . .	244
4.2.3.155 virDomainLeaseIndex . . . . .	244
4.2.3.156 virDomainLeaseInsert . . . . .	244
4.2.3.157 virDomainLeaseInsertPreAlloc . . . . .	244
4.2.3.158 virDomainLeaseInsertPreAlloced . . . . .	244
4.2.3.159 virDomainLeaseRemove . . . . .	244
4.2.3.160 virDomainLeaseRemoveAt . . . . .	244
4.2.3.161 virDomainLifecycleDefFormat . . . . .	244
4.2.3.162 virDomainLifecycleParseXML . . . . .	244
4.2.3.163 virDomainList . . . . .	244
4.2.3.164 virDomainListPopulate . . . . .	244
4.2.3.165 virDomainLiveConfigHelperMethod . . . . .	244
4.2.3.166 virDomainLoadAllConfigs . . . . .	244

4.2.3.167	virDomainLoadConfig	244
4.2.3.168	virDomainLoadStatus	244
4.2.3.169	virDomainMemballoonDefCheckABIStability	244
4.2.3.170	virDomainMemballoonDefFormat	244
4.2.3.171	virDomainMemballoonDefFree	244
4.2.3.172	virDomainMemballoonDefParseXML	245
4.2.3.173	virDomainNetDefCheckABIStability	245
4.2.3.174	virDomainNetDefFormat	245
4.2.3.175	<b>POL Mod</b> virDomainNetDefFree	245
4.2.3.176	<b>POL Mod</b> virDomainNetDefParseXML	245
4.2.3.177	virDomainNetFind	245
4.2.3.178	virDomainNetGetActualBandwidth	246
4.2.3.179	virDomainNetGetActualBridgeName	246
4.2.3.180	<b>POL New</b> virDomainNetGetActualBridgeType	246
4.2.3.181	virDomainNetGetActualDirectDev	246
4.2.3.182	virDomainNetGetActualDirectMode	246
4.2.3.183	virDomainNetGetActualHostdev	246
4.2.3.184	virDomainNetGetActualType	246
4.2.3.185	<b>POL Mod</b> virDomainNetGetActualVirtPortProfile	246
4.2.3.186	virDomainNetGetActualVlan	246
4.2.3.187	virDomainNetIndexByMac	246
4.2.3.188	virDomainNetInsert	246
4.2.3.189	virDomainNetRemove	246
4.2.3.190	virDomainNetRemoveByMac	246
4.2.3.191	virDomainObjAssignDef	246
4.2.3.192	virDomainObjCopyPersistentDef	246
4.2.3.193	virDomainObjDispose	246
4.2.3.194	virDomainObjFormat	246
4.2.3.195	virDomainObjGetPersistentDef	246
4.2.3.196	virDomainObjGetState	247
4.2.3.197	virDomainObjIsDuplicate	247
4.2.3.198	virDomainObjListCopyActiveIDs	247
4.2.3.199	virDomainObjListCopyInactiveNames	247
4.2.3.200	virDomainObjListCountActive	247
4.2.3.201	virDomainObjListCountInactive	247
4.2.3.202	virDomainObjListDataFree	247
4.2.3.203	virDomainObjListDeinit	247
4.2.3.204	virDomainObjListGetActiveIDs	247
4.2.3.205	virDomainObjListGetInactiveNames	247
4.2.3.206	virDomainObjListInit	247

4.2.3.207 virDomainObjListNumOfDomains . . . . .	247
4.2.3.208 virDomainObjListSearchID . . . . .	247
4.2.3.209 virDomainObjListSearchName . . . . .	247
4.2.3.210 virDomainObjLock . . . . .	247
4.2.3.211 virDomainObjNew . . . . .	247
4.2.3.212 virDomainObjParseFile . . . . .	247
4.2.3.213 virDomainObjParseNode . . . . .	247
4.2.3.214 virDomainObjParseXML . . . . .	247
4.2.3.215 virDomainObjSetDefTransient . . . . .	247
4.2.3.216 virDomainObjSetState . . . . .	247
4.2.3.217 virDomainObjTaint . . . . .	247
4.2.3.218 virDomainObjUnlock . . . . .	247
4.2.3.219 virDomainParallelDefCheckABIStability . . . . .	248
4.2.3.220 virDomainParseLegacyDeviceAddress . . . . .	248
4.2.3.221 virDomainParseMemory . . . . .	248
4.2.3.222 virDomainParseScaledValue . . . . .	248
4.2.3.223 virDomainPMStateParseXML . . . . .	248
4.2.3.224 virDomainRedirdevDefFormat . . . . .	248
4.2.3.225 virDomainRedirdevDefFree . . . . .	248
4.2.3.226 virDomainRedirdevDefParseXML . . . . .	248
4.2.3.227 virDomainRedirFilterDefCheckABIStability . . . . .	248
4.2.3.228 virDomainRedirFilterDefFormat . . . . .	248
4.2.3.229 virDomainRedirFilterDefFree . . . . .	248
4.2.3.230 virDomainRedirFilterDefParseXML . . . . .	248
4.2.3.231 virDomainRedirFilterUsbDevDefParseXML . . . . .	248
4.2.3.232 virDomainRedirFilterUsbVersionHelper . . . . .	248
4.2.3.233 virDomainRemoveInactive . . . . .	248
4.2.3.234 virDomainSaveConfig . . . . .	248
4.2.3.235 virDomainSaveStatus . . . . .	248
4.2.3.236 virDomainSaveXML . . . . .	248
4.2.3.237 virDomainSerialDefCheckABIStability . . . . .	248
4.2.3.238 virDomainSmartcardDefCheckABIStability . . . . .	248
4.2.3.239 virDomainSmartcardDefForeach . . . . .	248
4.2.3.240 virDomainSmartcardDefFormat . . . . .	248
4.2.3.241 virDomainSmartcardDefFree . . . . .	249
4.2.3.242 virDomainSmartcardDefParseXML . . . . .	249
4.2.3.243 virDomainSoundCodecDefFormat . . . . .	249
4.2.3.244 virDomainSoundCodecDefFree . . . . .	249
4.2.3.245 virDomainSoundCodecDefParseXML . . . . .	249
4.2.3.246 virDomainSoundDefCheckABIStability . . . . .	249

4.2.3.247	<a href="#">virDomainSoundDefFormat</a>	249
4.2.3.248	<a href="#">virDomainSoundDefFree</a>	249
4.2.3.249	<a href="#">virDomainSoundDefParseXML</a>	249
4.2.3.250	<a href="#">virDomainStateReasonFromString</a>	249
4.2.3.251	<a href="#">virDomainStateReasonToString</a>	249
4.2.3.252	<a href="#">virDomainSysinfoDefFormat</a>	249
4.2.3.253	<a href="#">virDomainTimerDefCheckABIStability</a>	249
4.2.3.254	<a href="#">virDomainTimerDefFormat</a>	249
4.2.3.255	<a href="#">virDomainTimerDefParseXML</a>	249
4.2.3.256	<a href="#">virDomainVcpuPinAdd</a>	249
4.2.3.257	<a href="#">virDomainVcpuPinDefArrayFree</a>	249
4.2.3.258	<a href="#">virDomainVcpuPinDefCopy</a>	249
4.2.3.259	<a href="#">virDomainVcpuPinDefFree</a>	249
4.2.3.260	<a href="#">virDomainVcpuPinDefParseXML</a>	249
4.2.3.261	<a href="#">virDomainVcpuPinDel</a>	249
4.2.3.262	<a href="#">virDomainVcpuPinFindByVcpu</a>	249
4.2.3.263	<a href="#">virDomainVcpuPinIsDuplicate</a>	249
4.2.3.264	<a href="#">virDomainVideoAccelDefFormat</a>	250
4.2.3.265	<a href="#">virDomainVideoAccelDefParseXML</a>	250
4.2.3.266	<a href="#">virDomainVideoDefaultRAM</a>	250
4.2.3.267	<a href="#">virDomainVideoDefaultType</a>	250
4.2.3.268	<a href="#">virDomainVideoDefCheckABIStability</a>	250
4.2.3.269	<a href="#">virDomainVideoDefFormat</a>	250
4.2.3.270	<a href="#">virDomainVideoDefFree</a>	250
4.2.3.271	<a href="#">virDomainVideoDefParseXML</a>	250
4.2.3.272	<a href="#">virDomainWatchdogDefCheckABIStability</a>	250
4.2.3.273	<a href="#">virDomainWatchdogDefFormat</a>	250
4.2.3.274	<a href="#">virDomainWatchdogDefFree</a>	250
4.2.3.275	<a href="#">virDomainWatchdogDefParseXML</a>	250
4.2.3.276	<a href="#">virSecurityDeviceLabelDefFormat</a>	250
4.2.3.277	<a href="#">virSecurityDeviceLabelDefFree</a>	250
4.2.3.278	<a href="#">virSecurityDeviceLabelDefParseXML</a>	250
4.2.3.279	<a href="#">virSecurityLabelDefFormat</a>	250
4.2.3.280	<a href="#">virSecurityLabelDefFree</a>	250
4.2.3.281	<a href="#">virSecurityLabelDefParseXML</a>	250
4.2.3.282	<a href="#">virSecurityLabelDefsParseXML</a>	250
4.2.3.283	<a href="#">virSysinfoParseXML</a>	250
4.3	<a href="#">src/conf/domain_conf.h File Reference</a>	251
4.3.1	<a href="#">Macro Definition Documentation</a>	264
4.3.1.1	<a href="#">VIR_CONNECT_LIST_DOMAINS_FILTERS_ACTIVE</a>	264



4.3.1.2	VIR_CONNECT_LIST_DOMAINS_FILTERS_ALL . . . . .	264
4.3.1.3	VIR_CONNECT_LIST_DOMAINS_FILTERS_AUTOSTART . . . . .	264
4.3.1.4	VIR_CONNECT_LIST_DOMAINS_FILTERS_MANAGEDSAVE . . . . .	264
4.3.1.5	VIR_CONNECT_LIST_DOMAINS_FILTERS_PERSISTENT . . . . .	264
4.3.1.6	VIR_CONNECT_LIST_DOMAINS_FILTERS_SNAPSHOT . . . . .	265
4.3.1.7	VIR_CONNECT_LIST_DOMAINS_FILTERS_STATE . . . . .	265
4.3.1.8	VIR_DOMAIN_CPUMASK_LEN . . . . .	265
4.3.1.9	VIR_DOMAIN_FS_RAM_DEFAULT_USAGE . . . . .	265
4.3.1.10	VIR_DOMAIN_MAX_BOOT_DEVS . . . . .	265
4.3.1.11	VIR_DOMAIN_SMARTCARD_DEFAULT_DATABASE . . . . .	265
4.3.1.12	VIR_DOMAIN_SMARTCARD_NUM_CERTIFICATES . . . . .	265
4.3.1.13	VIR_NET_GENERATED_PREFIX . . . . .	265
4.3.2	Typedef Documentation . . . . .	265
4.3.2.1	virBlkioDeviceWeight . . . . .	265
4.3.2.2	virBlkioDeviceWeightPtr . . . . .	265
4.3.2.3	virDomainActualNetDef . . . . .	265
4.3.2.4	virDomainActualNetDefPtr . . . . .	265
4.3.2.5	virDomainBIOSDef . . . . .	265
4.3.2.6	virDomainBIOSDefPtr . . . . .	265
4.3.2.7	virDomainBlockIoTuneInfo . . . . .	265
4.3.2.8	virDomainBlockIoTuneInfoPtr . . . . .	265
4.3.2.9	virDomainChrDef . . . . .	265
4.3.2.10	virDomainChrDefIterator . . . . .	265
4.3.2.11	virDomainChrDefPtr . . . . .	265
4.3.2.12	virDomainChrSourceDef . . . . .	265
4.3.2.13	virDomainChrSourceDefPtr . . . . .	265
4.3.2.14	virDomainClockDef . . . . .	266
4.3.2.15	virDomainClockDefPtr . . . . .	266
4.3.2.16	virDomainControllerDef . . . . .	266
4.3.2.17	virDomainControllerDefPtr . . . . .	266
4.3.2.18	virDomainDef . . . . .	266
4.3.2.19	virDomainDefPtr . . . . .	266
4.3.2.20	virDomainDeviceCcidAddress . . . . .	266
4.3.2.21	virDomainDeviceCcidAddressPtr . . . . .	266
4.3.2.22	virDomainDeviceDef . . . . .	266
4.3.2.23	virDomainDeviceDefPtr . . . . .	266
4.3.2.24	virDomainDeviceDriveAddress . . . . .	266
4.3.2.25	virDomainDeviceDriveAddressPtr . . . . .	266
4.3.2.26	virDomainDeviceInfo . . . . .	266
4.3.2.27	virDomainDeviceInfoCallback . . . . .	266

4.3.2.28	<a href="#">virDomainDeviceInfoPtr</a>	266
4.3.2.29	<a href="#">virDomainDeviceSpaprVioAddress</a>	266
4.3.2.30	<a href="#">virDomainDeviceSpaprVioAddressPtr</a>	266
4.3.2.31	<a href="#">virDomainDeviceUSBAddress</a>	266
4.3.2.32	<a href="#">virDomainDeviceUSBAddressPtr</a>	266
4.3.2.33	<a href="#">virDomainDeviceUSBMaster</a>	266
4.3.2.34	<a href="#">virDomainDeviceUSBMasterPtr</a>	266
4.3.2.35	<a href="#">virDomainDeviceVirtioSerialAddress</a>	266
4.3.2.36	<a href="#">virDomainDeviceVirtioSerialAddressPtr</a>	266
4.3.2.37	<a href="#">virDomainDiskDef</a>	266
4.3.2.38	<a href="#">virDomainDiskDefPathIterator</a>	266
4.3.2.39	<a href="#">virDomainDiskDefPtr</a>	266
4.3.2.40	<a href="#">virDomainDiskHostDef</a>	266
4.3.2.41	<a href="#">virDomainDiskHostDefPtr</a>	267
4.3.2.42	<a href="#">virDomainFSDef</a>	267
4.3.2.43	<a href="#">virDomainFSDefPtr</a>	267
4.3.2.44	<a href="#">virDomainGraphicsAuthDef</a>	267
4.3.2.45	<a href="#">virDomainGraphicsAuthDefPtr</a>	267
4.3.2.46	<a href="#">virDomainGraphicsDef</a>	267
4.3.2.47	<a href="#">virDomainGraphicsDefPtr</a>	267
4.3.2.48	<a href="#">virDomainGraphicsListenDef</a>	267
4.3.2.49	<a href="#">virDomainGraphicsListenDefPtr</a>	267
4.3.2.50	<a href="#">virDomainHostdevDef</a>	267
4.3.2.51	<a href="#">virDomainHostdevDefPtr</a>	267
4.3.2.52	<a href="#">virDomainHostdevOrigStates</a>	267
4.3.2.53	<a href="#">virDomainHostdevOrigStatesPtr</a>	267
4.3.2.54	<a href="#">virDomainHostdevSubsys</a>	267
4.3.2.55	<a href="#">virDomainHostdevSubsysPtr</a>	267
4.3.2.56	<a href="#">virDomainHubDef</a>	267
4.3.2.57	<a href="#">virDomainHubDefPtr</a>	267
4.3.2.58	<a href="#">virDomainInputDef</a>	267
4.3.2.59	<a href="#">virDomainInputDefPtr</a>	267
4.3.2.60	<a href="#">virDomainLeaseDef</a>	267
4.3.2.61	<a href="#">virDomainLeaseDefPtr</a>	267
4.3.2.62	<a href="#">virDomainLoadConfigNotify</a>	267
4.3.2.63	<a href="#">virDomainMemballoonDef</a>	267
4.3.2.64	<a href="#">virDomainMemballoonDefPtr</a>	267
4.3.2.65	<a href="#">virDomainNetDef</a>	267
4.3.2.66	<a href="#">virDomainNetDefPtr</a>	267
4.3.2.67	<a href="#">virDomainNumatuneDef</a>	267

4.3.2.68	<a href="#">virDomainNumatuneDefPtr</a>	267
4.3.2.69	<a href="#">virDomainObj</a>	268
4.3.2.70	<a href="#">virDomainObjList</a>	268
4.3.2.71	<a href="#">virDomainObjListPtr</a>	268
4.3.2.72	<a href="#">virDomainObjPtr</a>	268
4.3.2.73	<a href="#">virDomainOSDef</a>	268
4.3.2.74	<a href="#">virDomainOSDefPtr</a>	268
4.3.2.75	<a href="#">virDomainRedirdevDef</a>	268
4.3.2.76	<a href="#">virDomainRedirdevDefPtr</a>	268
4.3.2.77	<a href="#">virDomainRedirFilterDef</a>	268
4.3.2.78	<a href="#">virDomainRedirFilterDefPtr</a>	268
4.3.2.79	<a href="#">virDomainRedirFilterUsbDevDef</a>	268
4.3.2.80	<a href="#">virDomainRedirFilterUsbDevDefPtr</a>	268
4.3.2.81	<a href="#">virDomainSmartcardDef</a>	268
4.3.2.82	<a href="#">virDomainSmartcardDefIterator</a>	268
4.3.2.83	<a href="#">virDomainSmartcardDefPtr</a>	268
4.3.2.84	<a href="#">virDomainSnapshotObj</a>	268
4.3.2.85	<a href="#">virDomainSnapshotObjList</a>	268
4.3.2.86	<a href="#">virDomainSnapshotObjListPtr</a>	268
4.3.2.87	<a href="#">virDomainSnapshotObjPtr</a>	268
4.3.2.88	<a href="#">virDomainSoundCodecDef</a>	268
4.3.2.89	<a href="#">virDomainSoundCodecDefPtr</a>	268
4.3.2.90	<a href="#">virDomainSoundDef</a>	268
4.3.2.91	<a href="#">virDomainSoundDefPtr</a>	268
4.3.2.92	<a href="#">virDomainStateReason</a>	268
4.3.2.93	<a href="#">virDomainTimerCatchupDef</a>	268
4.3.2.94	<a href="#">virDomainTimerCatchupDefPtr</a>	268
4.3.2.95	<a href="#">virDomainTimerDef</a>	268
4.3.2.96	<a href="#">virDomainTimerDefPtr</a>	268
4.3.2.97	<a href="#">virDomainVcpuPinDef</a>	269
4.3.2.98	<a href="#">virDomainVcpuPinDefPtr</a>	269
4.3.2.99	<a href="#">virDomainVideoAccelDef</a>	269
4.3.2.100	<a href="#">virDomainVideoAccelDefPtr</a>	269
4.3.2.101	<a href="#">virDomainVideoDef</a>	269
4.3.2.102	<a href="#">virDomainVideoDefPtr</a>	269
4.3.2.103	<a href="#">virDomainVirtioSerialOpts</a>	269
4.3.2.104	<a href="#">virDomainVirtioSerialOptsPtr</a>	269
4.3.2.105	<a href="#">virDomainWatchdogDef</a>	269
4.3.2.106	<a href="#">virDomainWatchdogDefPtr</a>	269
4.3.2.107	<a href="#">virLifecycleFromStringFunc</a>	269

4.3.2.108	virLifecycleToStringFunc	269
4.3.2.109	virSecurityDeviceLabelDef	269
4.3.2.110	virSecurityDeviceLabelDefPtr	269
4.3.2.111	virSecurityLabelDef	269
4.3.2.112	virSecurityLabelDefPtr	269
4.3.3	Enumeration Type Documentation	269
4.3.3.1	anonymous enum	269
4.3.3.2	virDomainApicEoi	269
4.3.3.3	virDomainBIOSUseserial	270
4.3.3.4	virDomainBootMenu	270
4.3.3.5	virDomainBootOrder	270
4.3.3.6	virDomainChrChannelTargetType	270
4.3.3.7	virDomainChrConsoleTargetType	270
4.3.3.8	virDomainChrDeviceType	271
4.3.3.9	virDomainChrSpicevmcName	271
4.3.3.10	virDomainChrTcpProtocol	271
4.3.3.11	virDomainChrType	271
4.3.3.12	virDomainClockBasis	272
4.3.3.13	virDomainClockOffsetType	272
4.3.3.14	virDomainControllerMaster	272
4.3.3.15	virDomainControllerModelSCSI	272
4.3.3.16	virDomainControllerModelUSB	272
4.3.3.17	virDomainControllerType	273
4.3.3.18	virDomainCpuPlacementMode	273
4.3.3.19	virDomainDeviceAddressType	273
4.3.3.20	virDomainDeviceType	273
4.3.3.21	virDomainDiskBus	274
4.3.3.22	virDomainDiskCache	274
4.3.3.23	virDomainDiskCopyOnRead	274
4.3.3.24	virDomainDiskDevice	275
4.3.3.25	virDomainDiskErrorPolicy	275
4.3.3.26	virDomainDiskGeometryTrans	275
4.3.3.27	virDomainDiskIo	275
4.3.3.28	virDomainDiskProtocol	275
4.3.3.29	virDomainDiskSecretType	276
4.3.3.30	virDomainDiskTray	276
4.3.3.31	virDomainDiskType	276
4.3.3.32	virDomainFeature	276
4.3.3.33	virDomainFSAccessMode	276
4.3.3.34	virDomainFSDriverType	277

4.3.3.35	<a href="#">virDomainFSType</a>	277
4.3.3.36	<a href="#">virDomainFSWrpolicy</a>	277
4.3.3.37	<a href="#">virDomainGraphicsAuthConnectedType</a>	277
4.3.3.38	<a href="#">virDomainGraphicsListenType</a>	277
4.3.3.39	<a href="#">virDomainGraphicsSpiceChannelMode</a>	278
4.3.3.40	<a href="#">virDomainGraphicsSpiceChannelName</a>	278
4.3.3.41	<a href="#">virDomainGraphicsSpiceClipboardCypaste</a>	278
4.3.3.42	<a href="#">virDomainGraphicsSpiceImageCompression</a>	278
4.3.3.43	<a href="#">virDomainGraphicsSpiceJpegCompression</a>	279
4.3.3.44	<a href="#">virDomainGraphicsSpiceMouseMode</a>	279
4.3.3.45	<a href="#">virDomainGraphicsSpicePlaybackCompression</a>	279
4.3.3.46	<a href="#">virDomainGraphicsSpiceStreamingMode</a>	279
4.3.3.47	<a href="#">virDomainGraphicsSpiceZlibCompression</a>	279
4.3.3.48	<a href="#">virDomainGraphicsType</a>	280
4.3.3.49	<a href="#">virDomainHostdevMode</a>	280
4.3.3.50	<a href="#">virDomainHostdevSubsysType</a>	280
4.3.3.51	<a href="#">virDomainHubType</a>	280
4.3.3.52	<a href="#">virDomainInputBus</a>	280
4.3.3.53	<a href="#">virDomainInputType</a>	280
4.3.3.54	<a href="#">virDomainIoEventFd</a>	281
4.3.3.55	<a href="#">virDomainLifecycleAction</a>	281
4.3.3.56	<a href="#">virDomainLifecycleCrashAction</a>	281
4.3.3.57	<a href="#">virDomainMemDump</a>	281
4.3.3.58	<a href="#">virDomainNetBackendType</a>	281
4.3.3.59	<a href="#">virDomainNetInterfaceLinkState</a>	282
4.3.3.60	<a href="#">virDomainNetType</a>	282
4.3.3.61	<a href="#">virDomainNetVirtioTxModeType</a>	282
4.3.3.62	<a href="#">virDomainNumatuneMemPlacementMode</a>	282
4.3.3.63	<a href="#">virDomainPciRombarMode</a>	282
4.3.3.64	<a href="#">virDomainPMState</a>	283
4.3.3.65	<a href="#">virDomainRedirdevBus</a>	283
4.3.3.66	<a href="#">virDomainSeclabelType</a>	283
4.3.3.67	<a href="#">virDomainSmartcardType</a>	283
4.3.3.68	<a href="#">virDomainSmbiosMode</a>	283
4.3.3.69	<a href="#">virDomainSoundCodecType</a>	283
4.3.3.70	<a href="#">virDomainSoundModel</a>	284
4.3.3.71	<a href="#">virDomainStartupPolicy</a>	284
4.3.3.72	<a href="#">virDomainTaintFlags</a>	284
4.3.3.73	<a href="#">virDomainTimerModeType</a>	284
4.3.3.74	<a href="#">virDomainTimerNameType</a>	285

4.3.3.75	virDomainTimerTickpolicyType	285
4.3.3.76	virDomainTimerTrackType	285
4.3.3.77	virDomainVideoType	285
4.3.3.78	virDomainVirtioEventIdx	285
4.3.3.79	virDomainVirtType	286
4.3.3.80	virDomainWatchdogAction	286
4.3.3.81	virDomainWatchdogModel	286
4.3.4	Function Documentation	286
4.3.4.1	virBlkioDeviceWeightArrayClear	286
4.3.4.2	virDiskNameToBusDeviceIndex	286
4.3.4.3	<b>POL Mod</b> virDomainActualNetDefFree	286
4.3.4.4	virDomainAssignDef	287
4.3.4.5	virDomainChrDefForeach	287
4.3.4.6	virDomainChrDefFree	287
4.3.4.7	virDomainChrDefGetSecurityLabelDef	287
4.3.4.8	virDomainChrDefNew	287
4.3.4.9	virDomainChrSourceDefCopy	287
4.3.4.10	virDomainChrSourceDefFree	287
4.3.4.11	virDomainConfigFile	287
4.3.4.12	virDomainControllerDefFree	287
4.3.4.13	virDomainControllerFind	287
4.3.4.14	virDomainControllerInsert	287
4.3.4.15	virDomainControllerInsertPreAlloced	287
4.3.4.16	virDomainControllerRemove	287
4.3.4.17	virDomainDefAddImplicitControllers	287
4.3.4.18	virDomainDefAddSecurityLabelDef	287
4.3.4.19	virDomainDefCheckABIStability	287
4.3.4.20	virDomainDefClearDeviceAliases	287
4.3.4.21	virDomainDefClearPCIAddresses	287
4.3.4.22	virDomainDefCompatibleDevice	287
4.3.4.23	virDomainDefFormat	287
4.3.4.24	virDomainDefFormatInternal	287
4.3.4.25	virDomainDefFree	287
4.3.4.26	virDomainDefGetSecurityLabelDef	288
4.3.4.27	virDomainDefParseFile	288
4.3.4.28	virDomainDefParseNode	288
4.3.4.29	virDomainDefParseString	288
4.3.4.30	virDomainDeleteConfig	288
4.3.4.31	virDomainDeviceAddressIsValid	288
4.3.4.32	virDomainDeviceDefCopy	288

4.3.4.33	<a href="#">virDomainDeviceDefFree</a>	288
4.3.4.34	<a href="#">virDomainDeviceDefParse</a>	288
4.3.4.35	<a href="#">virDomainDeviceInfoClear</a>	288
4.3.4.36	<a href="#">virDomainDeviceInfoIterate</a>	288
4.3.4.37	<a href="#">virDomainDiskDefAssignAddress</a>	288
4.3.4.38	<a href="#">virDomainDiskDefForeachPath</a>	288
4.3.4.39	<a href="#">virDomainDiskDefFree</a>	288
4.3.4.40	<a href="#">virDomainDiskDefGetSecurityLabelDef</a>	288
4.3.4.41	<a href="#">virDomainDiskFindControllerModel</a>	288
4.3.4.42	<a href="#">virDomainDiskHostDefFree</a>	288
4.3.4.43	<a href="#">virDomainDiskIndexByName</a>	288
4.3.4.44	<a href="#">virDomainDiskInsert</a>	288
4.3.4.45	<a href="#">virDomainDiskInsertPreAlloced</a>	288
4.3.4.46	<a href="#">virDomainDiskPathByName</a>	288
4.3.4.47	<a href="#">virDomainDiskRemove</a>	288
4.3.4.48	<a href="#">virDomainDiskRemoveByName</a>	288
4.3.4.49	<a href="#">virDomainEmulatorPinAdd</a>	288
4.3.4.50	<a href="#">virDomainEmulatorPinDel</a>	288
4.3.4.51	<a href="#">virDomainFindByID</a>	289
4.3.4.52	<a href="#">virDomainFindByName</a>	289
4.3.4.53	<a href="#">virDomainFindByUUID</a>	289
4.3.4.54	<a href="#">virDomainFSDefFree</a>	289
4.3.4.55	<a href="#">virDomainFSIndexByName</a>	289
4.3.4.56	<a href="#">virDomainGetRootFilesystem</a>	289
4.3.4.57	<a href="#">virDomainGraphicsDefFree</a>	289
4.3.4.58	<a href="#">virDomainGraphicsListenGetAddress</a>	289
4.3.4.59	<a href="#">virDomainGraphicsListenGetNetwork</a>	289
4.3.4.60	<a href="#">virDomainGraphicsListenGetType</a>	289
4.3.4.61	<a href="#">virDomainGraphicsListenSetAddress</a>	289
4.3.4.62	<a href="#">virDomainGraphicsListenSetNetwork</a>	289
4.3.4.63	<a href="#">virDomainGraphicsListenSetType</a>	289
4.3.4.64	<a href="#">virDomainHostdevDefAlloc</a>	289
4.3.4.65	<a href="#">virDomainHostdevDefClear</a>	289
4.3.4.66	<a href="#">virDomainHostdevDefFree</a>	289
4.3.4.67	<a href="#">virDomainHostdevFind</a>	289
4.3.4.68	<a href="#">virDomainHostdevInsert</a>	289
4.3.4.69	<a href="#">virDomainHostdevRemove</a>	289
4.3.4.70	<a href="#">virDomainHubDefFree</a>	289
4.3.4.71	<a href="#">virDomainInputDefFree</a>	289
4.3.4.72	<a href="#">virDomainLeaseDefFree</a>	289

4.3.4.73	virDomainLeaseIndex	289
4.3.4.74	virDomainLeaseInsert	289
4.3.4.75	virDomainLeaseInsertPreAlloc	289
4.3.4.76	virDomainLeaseInsertPreAlloced	289
4.3.4.77	virDomainLeaseRemove	289
4.3.4.78	virDomainLeaseRemoveAt	290
4.3.4.79	virDomainList	290
4.3.4.80	virDomainLiveConfigHelperMethod	290
4.3.4.81	virDomainLoadAllConfigs	290
4.3.4.82	virDomainMemballoonDefFree	290
4.3.4.83	<b>POL Mod</b> virDomainNetDefFree	290
4.3.4.84	virDomainNetFind	290
4.3.4.85	virDomainNetGetActualBandwidth	290
4.3.4.86	virDomainNetGetActualBridgeName	290
4.3.4.87	<b>POL New</b> virDomainNetGetActualBridgeType	290
4.3.4.88	virDomainNetGetActualDirectDev	290
4.3.4.89	virDomainNetGetActualDirectMode	290
4.3.4.90	virDomainNetGetActualHostdev	290
4.3.4.91	virDomainNetGetActualType	290
4.3.4.92	<b>POL Mod</b> virDomainNetGetActualVirtPortProfile	290
4.3.4.93	virDomainNetGetActualVlan	291
4.3.4.94	virDomainNetIndexByMac	291
4.3.4.95	virDomainNetInsert	291
4.3.4.96	virDomainNetRemove	291
4.3.4.97	virDomainNetRemoveByMac	291
4.3.4.98	virDomainObjAssignDef	291
4.3.4.99	virDomainObjCopyPersistentDef	291
4.3.4.100	virDomainObjGetPersistentDef	291
4.3.4.101	virDomainObjGetState	291
4.3.4.102	virDomainObjsActive	291
4.3.4.103	virDomainObjsDuplicate	291
4.3.4.104	virDomainObjListDeinit	291
4.3.4.105	virDomainObjListGetActiveIDs	291
4.3.4.106	virDomainObjListGetInactiveNames	291
4.3.4.107	virDomainObjListInit	291
4.3.4.108	virDomainObjListNumOfDomains	291
4.3.4.109	virDomainObjLock	291
4.3.4.110	virDomainObjNew	291
4.3.4.111	virDomainObjSetDefTransient	291
4.3.4.112	virDomainObjSetState	291



4.3.4.113	virDomainObjTaint	291
4.3.4.114	virDomainObjUnlock	291
4.3.4.115	virDomainRedirdevDefFree	291
4.3.4.116	virDomainRedirFilterDefFree	292
4.3.4.117	virDomainRemoveInactive	292
4.3.4.118	virDomainSaveConfig	292
4.3.4.119	virDomainSaveStatus	292
4.3.4.120	virDomainSaveXML	292
4.3.4.121	virDomainSmartcardDefForeach	292
4.3.4.122	virDomainSmartcardDefFree	292
4.3.4.123	virDomainSoundCodecDefFree	292
4.3.4.124	virDomainSoundDefFree	292
4.3.4.125	virDomainStateReasonFromString	292
4.3.4.126	virDomainStateReasonToString	292
4.3.4.127	virDomainVcpuPinAdd	292
4.3.4.128	virDomainVcpuPinDefArrayFree	292
4.3.4.129	virDomainVcpuPinDefCopy	292
4.3.4.130	virDomainVcpuPinDefFree	292
4.3.4.131	virDomainVcpuPinDel	292
4.3.4.132	virDomainVcpuPinFindByVcpu	292
4.3.4.133	virDomainVcpuPinIsDuplicate	292
4.3.4.134	virDomainVideoDefaultRAM	292
4.3.4.135	virDomainVideoDefaultType	292
4.3.4.136	virDomainVideoDefFree	292
4.3.4.137	virDomainWatchdogDefFree	292
4.4	src/conf/network_conf.c File Reference	293
4.4.1	Macro Definition Documentation	295
4.4.1.1	MATCH	295
4.4.1.2	MAX_BRIDGE_ID	295
4.4.1.3	VIR_FROM_THIS	295
4.4.2	Function Documentation	295
4.4.2.1	VIR_ENUM_IMPL	295
4.4.2.2	virNetworkAllocateBridge	295
4.4.2.3	virNetworkAssignDef	295
4.4.2.4	virNetworkBridgeInUse	295
4.4.2.5	virNetworkConfigChangeSetup	296
4.4.2.6	virNetworkConfigFile	296
4.4.2.7	virNetworkDefCopy	296
4.4.2.8	POL Mod virNetworkDefFormat	296
4.4.2.9	POL Mod virNetworkDefFree	296

4.4.2.10	virNetworkDefGetIpByIndex	296
4.4.2.11	virNetworkDefParse	296
4.4.2.12	virNetworkDefParseFile	296
4.4.2.13	virNetworkDefParseNode	296
4.4.2.14	virNetworkDefParseString	296
4.4.2.15	<b>POL Mod</b> virNetworkDefParseXML	296
4.4.2.16	virNetworkDefUpdateBridge	297
4.4.2.17	virNetworkDefUpdateCheckElementName	297
4.4.2.18	virNetworkDefUpdateDNSHost	297
4.4.2.19	virNetworkDefUpdateDNSSrv	297
4.4.2.20	virNetworkDefUpdateDNSTxt	297
4.4.2.21	virNetworkDefUpdateDomain	297
4.4.2.22	virNetworkDefUpdateForward	297
4.4.2.23	virNetworkDefUpdateForwardInterface	297
4.4.2.24	virNetworkDefUpdateForwardPF	297
4.4.2.25	virNetworkDefUpdateIP	297
4.4.2.26	virNetworkDefUpdateIPDHCPHost	297
4.4.2.27	virNetworkDefUpdateIPDHCPRange	297
4.4.2.28	virNetworkDefUpdateNoSupport	297
4.4.2.29	virNetworkDefUpdatePortGroup	297
4.4.2.30	<b>POL Mod</b> virNetworkDefUpdateSection	297
4.4.2.31	<b>POL New</b> virNetworkDefUpdateTunnel	298
4.4.2.32	virNetworkDefUpdateUnknownCommand	298
4.4.2.33	virNetworkDeleteConfig	298
4.4.2.34	virNetworkDHCPDefParse	298
4.4.2.35	virNetworkDHCPHostDefClear	298
4.4.2.36	virNetworkDHCPHostDefParse	298
4.4.2.37	virNetworkDHCPRangeDefParse	298
4.4.2.38	virNetworkDNSDefFormat	298
4.4.2.39	virNetworkDNSDefFree	298
4.4.2.40	virNetworkDNSDefParseXML	298
4.4.2.41	virNetworkDNSHostsDefParseXML	298
4.4.2.42	virNetworkDNSSrvDefParseXML	298
4.4.2.43	virNetworkFindByName	299
4.4.2.44	virNetworkForwardIfDefClear	299
4.4.2.45	virNetworkForwardPfDefClear	299
4.4.2.46	virNetworkIpDefByIndex	299
4.4.2.47	virNetworkIpDefClear	299
4.4.2.48	virNetworkIpDefFormat	299
4.4.2.49	virNetworkIpDefNetmask	299

4.4.2.50	<a href="#">virNetworkIpDefPrefix</a>	299
4.4.2.51	<a href="#">virNetworkIPParseXML</a>	299
4.4.2.52	<a href="#">virNetworkList</a>	299
4.4.2.53	<a href="#">virNetworkLoadAllConfigs</a>	299
4.4.2.54	<a href="#">virNetworkLoadConfig</a>	299
4.4.2.55	<a href="#">virNetworkMatch</a>	299
4.4.2.56	<a href="#">virNetworkObjAssignDef</a>	299
4.4.2.57	<a href="#">virNetworkObjFree</a>	299
4.4.2.58	<a href="#">virNetworkObjGetPersistentDef</a>	299
4.4.2.59	<a href="#">virNetworkObjIsDuplicate</a>	299
4.4.2.60	<a href="#">virNetworkObjListFree</a>	299
4.4.2.61	<a href="#">virNetworkObjLock</a>	299
4.4.2.62	<a href="#">virNetworkObjReplacePersistentDef</a>	299
4.4.2.63	<a href="#">virNetworkObjSetDefTransient</a>	299
4.4.2.64	<a href="#">virNetworkObjUnlock</a>	299
4.4.2.65	<a href="#">virNetworkObjUnsetDefTransient</a>	299
4.4.2.66	<a href="#">virNetworkObjUpdate</a>	299
4.4.2.67	<a href="#">virNetworkPortGroupParseXML</a>	299
4.4.2.68	<a href="#">virNetworkRemoveInactive</a>	299
4.4.2.69	<a href="#">virNetworkSaveConfig</a>	300
4.4.2.70	<a href="#">virNetworkSaveStatus</a>	300
4.4.2.71	<a href="#">virNetworkSaveXML</a>	300
4.4.2.72	<a href="#">virNetworkSetBridgeMacAddr</a>	300
4.4.2.73	<a href="#">virNetworkSetBridgeName</a>	300
4.4.2.74	<b>POL New</b> <a href="#">virNetworkTunnelParseXML</a>	300
4.4.2.75	<a href="#">virPortGroupDefClear</a>	300
4.4.2.76	<a href="#">virPortGroupDefFormat</a>	300
4.4.2.77	<a href="#">virPortGroupFindByName</a>	300
4.4.2.78	<b>POL New</b> <a href="#">virTunnelDefFormat</a>	300
4.4.2.79	<b>POL New</b> <a href="#">virTunnelDefFree</a>	300
4.5	<a href="#">src/conf/network_conf.h File Reference</a>	301
4.5.1	<a href="#">Macro Definition Documentation</a>	303
4.5.1.1	<a href="#">DNS_RECORD_LENGTH_SRV</a>	303
4.5.1.2	<a href="#">VIR_CONNECT_LIST_NETWORKS_FILTERS_ACTIVE</a>	303
4.5.1.3	<a href="#">VIR_CONNECT_LIST_NETWORKS_FILTERS_ALL</a>	303
4.5.1.4	<a href="#">VIR_CONNECT_LIST_NETWORKS_FILTERS_AUTOSTART</a>	303
4.5.1.5	<a href="#">VIR_CONNECT_LIST_NETWORKS_FILTERS_PERSISTENT</a>	304
4.5.2	<a href="#">Typedef Documentation</a>	304
4.5.2.1	<a href="#">virNetworkDef</a>	304
4.5.2.2	<a href="#">virNetworkDefPtr</a>	304

4.5.2.3	<a href="#">virNetworkDHCPHostDef</a>	304
4.5.2.4	<a href="#">virNetworkDHCPHostDefPtr</a>	304
4.5.2.5	<a href="#">virNetworkDHCPRangeDef</a>	304
4.5.2.6	<a href="#">virNetworkDHCPRangeDefPtr</a>	304
4.5.2.7	<a href="#">virNetworkDNSDefPtr</a>	304
4.5.2.8	<a href="#">virNetworkDNSHostsDefPtr</a>	304
4.5.2.9	<a href="#">virNetworkDNSSrvRecordsDef</a>	304
4.5.2.10	<a href="#">virNetworkDNSSrvRecordsDefPtr</a>	304
4.5.2.11	<a href="#">virNetworkDNSTxtRecordsDef</a>	304
4.5.2.12	<a href="#">virNetworkDNSTxtRecordsDefPtr</a>	304
4.5.2.13	<a href="#">virNetworkForwardIfDef</a>	304
4.5.2.14	<a href="#">virNetworkForwardIfDefPtr</a>	304
4.5.2.15	<a href="#">virNetworkForwardPfDef</a>	304
4.5.2.16	<a href="#">virNetworkForwardPfDefPtr</a>	304
4.5.2.17	<a href="#">virNetworkIpDef</a>	304
4.5.2.18	<a href="#">virNetworkIpDefPtr</a>	304
4.5.2.19	<a href="#">virNetworkObj</a>	304
4.5.2.20	<a href="#">virNetworkObjList</a>	304
4.5.2.21	<a href="#">virNetworkObjListPtr</a>	304
4.5.2.22	<a href="#">virNetworkObjPtr</a>	304
4.5.2.23	<a href="#">virPortGroupDef</a>	304
4.5.2.24	<a href="#">virPortGroupDefPtr</a>	304
4.5.2.25	<a href="#">virTunnelDef</a>	305
4.5.2.26	<a href="#">virTunnelDefPtr</a>	305
4.5.3	<a href="#">Enumeration Type Documentation</a>	305
4.5.3.1	<a href="#">virNetworkForwardHostdevDeviceType</a>	305
4.5.3.2	<a href="#">virNetworkForwardType</a>	305
4.5.4	<a href="#">Function Documentation</a>	305
4.5.4.1	<a href="#">virNetworkAllocateBridge</a>	305
4.5.4.2	<a href="#">virNetworkAssignDef</a>	305
4.5.4.3	<a href="#">virNetworkBridgeInUse</a>	305
4.5.4.4	<a href="#">virNetworkConfigChangeSetup</a>	305
4.5.4.5	<a href="#">virNetworkConfigFile</a>	305
4.5.4.6	<a href="#">virNetworkDefCopy</a>	305
4.5.4.7	<a href="#">POL Mod <a href="#">virNetworkDefFormat</a></a>	305
4.5.4.8	<a href="#">virNetworkDefForwardIf</a>	306
4.5.4.9	<a href="#">POL Mod <a href="#">virNetworkDefFree</a></a>	306
4.5.4.10	<a href="#">virNetworkDefGetIpByIndex</a>	306
4.5.4.11	<a href="#">virNetworkDefParseFile</a>	306
4.5.4.12	<a href="#">virNetworkDefParseNode</a>	306

4.5.4.13	<a href="#">virNetworkDefParseString</a>	306
4.5.4.14	<a href="#">virNetworkDeleteConfig</a>	306
4.5.4.15	<a href="#">virNetworkFindByName</a>	306
4.5.4.16	<a href="#">virNetworkFindByUUID</a>	306
4.5.4.17	<a href="#">virNetworkIpDefNetmask</a>	306
4.5.4.18	<a href="#">virNetworkIpDefPrefix</a>	306
4.5.4.19	<a href="#">virNetworkList</a>	306
4.5.4.20	<a href="#">virNetworkLoadAllConfigs</a>	306
4.5.4.21	<a href="#">virNetworkLoadConfig</a>	306
4.5.4.22	<a href="#">virNetworkObjAssignDef</a>	306
4.5.4.23	<a href="#">virNetworkObjFree</a>	306
4.5.4.24	<a href="#">virNetworkObjGetPersistentDef</a>	306
4.5.4.25	<a href="#">virNetworkObjsActive</a>	306
4.5.4.26	<a href="#">virNetworkObjsDuplicate</a>	306
4.5.4.27	<a href="#">virNetworkObjListFree</a>	307
4.5.4.28	<a href="#">virNetworkObjLock</a>	307
4.5.4.29	<a href="#">virNetworkObjReplacePersistentDef</a>	307
4.5.4.30	<a href="#">virNetworkObjSetDefTransient</a>	307
4.5.4.31	<a href="#">virNetworkObjUnlock</a>	307
4.5.4.32	<a href="#">virNetworkObjUnsetDefTransient</a>	307
4.5.4.33	<a href="#">virNetworkObjUpdate</a>	307
4.5.4.34	<a href="#">virNetworkRemoveInactive</a>	307
4.5.4.35	<a href="#">virNetworkSaveConfig</a>	307
4.5.4.36	<a href="#">virNetworkSaveStatus</a>	307
4.5.4.37	<a href="#">virNetworkSaveXML</a>	307
4.5.4.38	<a href="#">virNetworkSetBridgeMacAddr</a>	307
4.5.4.39	<a href="#">virNetworkSetBridgeName</a>	307
4.5.4.40	<a href="#">virPortGroupFindByName</a>	307
4.6	<a href="#">src/driver.h File Reference</a>	307
4.6.1	<a href="#">Macro Definition Documentation</a>	315
4.6.1.1	<a href="#">VIR_DRV_SUPPORTS_FEATURE</a>	315
4.6.2	<a href="#">Typedef Documentation</a>	316
4.6.2.1	<a href="#">virDeviceMonitor</a>	316
4.6.2.2	<a href="#">virDeviceMonitorPtr</a>	316
4.6.2.3	<a href="#">virDevMonDeviceGetParent</a>	316
4.6.2.4	<a href="#">virDevMonDeviceGetXMLDesc</a>	316
4.6.2.5	<a href="#">virDevMonDeviceListCaps</a>	316
4.6.2.6	<a href="#">virDevMonDeviceLookupByName</a>	316
4.6.2.7	<a href="#">virDevMonDeviceNumOfCaps</a>	316
4.6.2.8	<a href="#">virDevMonListAllNodeDevices</a>	316

4.6.2.9	<a href="#">virDevMonListDevices</a>	316
4.6.2.10	<a href="#">virDevMonNumOfDevices</a>	316
4.6.2.11	<a href="#">virDriver</a>	316
4.6.2.12	<a href="#">virDriverPtr</a>	316
4.6.2.13	<a href="#">virDrvBaselineCPU</a>	316
4.6.2.14	<a href="#">virDrvClose</a>	316
4.6.2.15	<a href="#">virDrvCompareCPU</a>	316
4.6.2.16	<a href="#">virDrvConnectDomainXMLFromNative</a>	316
4.6.2.17	<a href="#">virDrvConnectDomainXMLToNative</a>	316
4.6.2.18	<a href="#">virDrvConnectFindStoragePoolSources</a>	316
4.6.2.19	<a href="#">virDrvConnectIsAlive</a>	316
4.6.2.20	<a href="#">virDrvConnectIsEncrypted</a>	316
4.6.2.21	<a href="#">virDrvConnectIsSecure</a>	316
4.6.2.22	<a href="#">virDrvConnectListAllNWFilters</a>	316
4.6.2.23	<a href="#">virDrvConnectListAllStoragePools</a>	316
4.6.2.24	<a href="#">virDrvConnectListDefinedStoragePools</a>	316
4.6.2.25	<a href="#">virDrvConnectListNWFilters</a>	317
4.6.2.26	<a href="#">virDrvConnectListStoragePools</a>	317
4.6.2.27	<a href="#">virDrvConnectNumOfDefinedStoragePools</a>	317
4.6.2.28	<a href="#">virDrvConnectNumOfNWFilters</a>	317
4.6.2.29	<a href="#">virDrvConnectNumOfStoragePools</a>	317
4.6.2.30	<a href="#">virDrvDomainAbortJob</a>	317
4.6.2.31	<a href="#">virDrvDomainAttachDevice</a>	317
4.6.2.32	<a href="#">virDrvDomainAttachDeviceFlags</a>	317
4.6.2.33	<a href="#">virDrvDomainBlockCommit</a>	317
4.6.2.34	<a href="#">virDrvDomainBlockJobAbort</a>	317
4.6.2.35	<a href="#">virDrvDomainBlockJobSetSpeed</a>	317
4.6.2.36	<a href="#">virDrvDomainBlockPeek</a>	317
4.6.2.37	<a href="#">virDrvDomainBlockPull</a>	317
4.6.2.38	<a href="#">virDrvDomainBlockRebase</a>	317
4.6.2.39	<a href="#">virDrvDomainBlockResize</a>	317
4.6.2.40	<a href="#">virDrvDomainBlockStats</a>	317
4.6.2.41	<a href="#">virDrvDomainBlockStatsFlags</a>	317
4.6.2.42	<a href="#">virDrvDomainCoreDump</a>	317
4.6.2.43	<a href="#">virDrvDomainCreate</a>	317
4.6.2.44	<a href="#">virDrvDomainCreateWithFlags</a>	317
4.6.2.45	<a href="#">virDrvDomainCreateXML</a>	317
4.6.2.46	<a href="#">virDrvDomainDefineXML</a>	317
4.6.2.47	<a href="#">virDrvDomainDestroy</a>	317
4.6.2.48	<a href="#">virDrvDomainDestroyFlags</a>	317

4.6.2.49	<a href="#">virDrvDomainDetachDevice</a>	318
4.6.2.50	<a href="#">virDrvDomainDetachDeviceFlags</a>	318
4.6.2.51	<a href="#">virDrvDomainEventDeregister</a>	318
4.6.2.52	<a href="#">virDrvDomainEventDeregisterAny</a>	318
4.6.2.53	<a href="#">virDrvDomainEventRegister</a>	318
4.6.2.54	<a href="#">virDrvDomainEventRegisterAny</a>	318
4.6.2.55	<a href="#">virDrvDomainGetAutostart</a>	318
4.6.2.56	<a href="#">virDrvDomainGetBlkioParameters</a>	318
4.6.2.57	<a href="#">virDrvDomainGetBlockInfo</a>	318
4.6.2.58	<a href="#">virDrvDomainGetBlockIoTune</a>	318
4.6.2.59	<a href="#">virDrvDomainGetBlockJobInfo</a>	318
4.6.2.60	<a href="#">virDrvDomainGetControllInfo</a>	318
4.6.2.61	<a href="#">virDrvDomainGetCPUStats</a>	318
4.6.2.62	<a href="#">virDrvDomainGetDiskErrors</a>	318
4.6.2.63	<a href="#">virDrvDomainGetEmulatorPinInfo</a>	318
4.6.2.64	<a href="#">virDrvDomainGetHostname</a>	318
4.6.2.65	<a href="#">virDrvDomainGetInfo</a>	318
4.6.2.66	<a href="#">virDrvDomainGetInterfaceParameters</a>	318
4.6.2.67	<a href="#">virDrvDomainGetJobInfo</a>	318
4.6.2.68	<a href="#">virDrvDomainGetMaxMemory</a>	318
4.6.2.69	<a href="#">virDrvDomainGetMaxVcpus</a>	318
4.6.2.70	<a href="#">virDrvDomainGetMemoryParameters</a>	318
4.6.2.71	<a href="#">virDrvDomainGetMetadata</a>	319
4.6.2.72	<a href="#">virDrvDomainGetNumaParameters</a>	319
4.6.2.73	<a href="#">virDrvDomainGetOSType</a>	319
4.6.2.74	<a href="#">virDrvDomainGetSchedulerParameters</a>	319
4.6.2.75	<a href="#">virDrvDomainGetSchedulerParametersFlags</a>	319
4.6.2.76	<a href="#">virDrvDomainGetSchedulerType</a>	319
4.6.2.77	<a href="#">virDrvDomainGetSecurityLabel</a>	319
4.6.2.78	<a href="#">virDrvDomainGetSecurityLabelList</a>	319
4.6.2.79	<a href="#">virDrvDomainGetState</a>	319
4.6.2.80	<a href="#">virDrvDomainGetVcpuPinInfo</a>	319
4.6.2.81	<a href="#">virDrvDomainGetVcpus</a>	319
4.6.2.82	<a href="#">virDrvDomainGetVcpusFlags</a>	319
4.6.2.83	<a href="#">virDrvDomainGetXMLDesc</a>	319
4.6.2.84	<a href="#">virDrvDomainHasCurrentSnapshot</a>	319
4.6.2.85	<a href="#">virDrvDomainHasManagedSaveImage</a>	319
4.6.2.86	<a href="#">virDrvDomainInjectNMI</a>	319
4.6.2.87	<a href="#">virDrvDomainInterfaceStats</a>	319
4.6.2.88	<a href="#">virDrvDomainIsActive</a>	319

4.6.2.89	<a href="#">virDrvDomainIsPersistent</a>	319
4.6.2.90	<a href="#">virDrvDomainIsUpdated</a>	319
4.6.2.91	<a href="#">virDrvDomainListAllSnapshots</a>	319
4.6.2.92	<a href="#">virDrvDomainLookupByID</a>	319
4.6.2.93	<a href="#">virDrvDomainLookupByName</a>	319
4.6.2.94	<a href="#">virDrvDomainLookupByUUID</a>	319
4.6.2.95	<a href="#">virDrvDomainManagedSave</a>	319
4.6.2.96	<a href="#">virDrvDomainManagedSaveRemove</a>	320
4.6.2.97	<a href="#">virDrvDomainMemoryPeek</a>	320
4.6.2.98	<a href="#">virDrvDomainMemoryStats</a>	320
4.6.2.99	<a href="#">virDrvDomainMigrateBegin3</a>	320
4.6.2.100	<a href="#">virDrvDomainMigrateConfirm3</a>	320
4.6.2.101	<a href="#">virDrvDomainMigrateFinish</a>	320
4.6.2.102	<a href="#">virDrvDomainMigrateFinish2</a>	320
4.6.2.103	<a href="#">virDrvDomainMigrateFinish3</a>	320
4.6.2.104	<a href="#">virDrvDomainMigrateGetMaxSpeed</a>	320
4.6.2.105	<a href="#">virDrvDomainMigratePerform</a>	320
4.6.2.106	<a href="#">virDrvDomainMigratePerform3</a>	320
4.6.2.107	<a href="#">virDrvDomainMigratePrepare</a>	320
4.6.2.108	<a href="#">virDrvDomainMigratePrepare2</a>	320
4.6.2.109	<a href="#">virDrvDomainMigratePrepare3</a>	320
4.6.2.110	<a href="#">virDrvDomainMigratePrepareTunnel</a>	320
4.6.2.111	<a href="#">virDrvDomainMigratePrepareTunnel3</a>	320
4.6.2.112	<a href="#">virDrvDomainMigrateSetMaxDowntime</a>	320
4.6.2.113	<a href="#">virDrvDomainMigrateSetMaxSpeed</a>	320
4.6.2.114	<a href="#">virDrvDomainOpenConsole</a>	321
4.6.2.115	<a href="#">virDrvDomainOpenGraphics</a>	321
4.6.2.116	<a href="#">virDrvDomainPinEmulator</a>	321
4.6.2.117	<a href="#">virDrvDomainPinVcpu</a>	321
4.6.2.118	<a href="#">virDrvDomainPinVcpuFlags</a>	321
4.6.2.119	<a href="#">virDrvDomainPMSuspendForDuration</a>	321
4.6.2.120	<a href="#">virDrvDomainPMWakeup</a>	321
4.6.2.121	<a href="#">virDrvDomainQemuAgentCommand</a>	321
4.6.2.122	<a href="#">virDrvDomainQemuAttach</a>	321
4.6.2.123	<a href="#">virDrvDomainQemuMonitorCommand</a>	321
4.6.2.124	<a href="#">virDrvDomainReboot</a>	321
4.6.2.125	<a href="#">virDrvDomainReset</a>	321
4.6.2.126	<a href="#">virDrvDomainRestore</a>	321
4.6.2.127	<a href="#">virDrvDomainRestoreFlags</a>	321
4.6.2.128	<a href="#">virDrvDomainResume</a>	321



4.6.2.129	<a href="#">virDrvDomainRevertToSnapshot</a>	321
4.6.2.130	<a href="#">virDrvDomainSave</a>	321
4.6.2.131	<a href="#">virDrvDomainSaveFlags</a>	321
4.6.2.132	<a href="#">virDrvDomainSaveImageDefineXML</a>	321
4.6.2.133	<a href="#">virDrvDomainSaveImageGetXMLDesc</a>	321
4.6.2.134	<a href="#">virDrvDomainScreenshot</a>	321
4.6.2.135	<a href="#">virDrvDomainSendKey</a>	321
4.6.2.136	<a href="#">virDrvDomainSetAutostart</a>	321
4.6.2.137	<a href="#">virDrvDomainSetBlkioParameters</a>	322
4.6.2.138	<a href="#">virDrvDomainSetBlockioTune</a>	322
4.6.2.139	<a href="#">virDrvDomainSetInterfaceParameters</a>	322
4.6.2.140	<a href="#">virDrvDomainSetMaxMemory</a>	322
4.6.2.141	<a href="#">virDrvDomainSetMemory</a>	322
4.6.2.142	<a href="#">virDrvDomainSetMemoryFlags</a>	322
4.6.2.143	<a href="#">virDrvDomainSetMemoryParameters</a>	322
4.6.2.144	<a href="#">virDrvDomainSetMetadata</a>	322
4.6.2.145	<a href="#">virDrvDomainSetNumaParameters</a>	322
4.6.2.146	<a href="#">virDrvDomainSetSchedulerParameters</a>	322
4.6.2.147	<a href="#">virDrvDomainSetSchedulerParametersFlags</a>	322
4.6.2.148	<a href="#">virDrvDomainSetVcpus</a>	322
4.6.2.149	<a href="#">virDrvDomainSetVcpusFlags</a>	322
4.6.2.150	<a href="#">virDrvDomainShutdown</a>	322
4.6.2.151	<a href="#">virDrvDomainShutdownFlags</a>	322
4.6.2.152	<a href="#">virDrvDomainSnapshotCreateXML</a>	322
4.6.2.153	<a href="#">virDrvDomainSnapshotCurrent</a>	322
4.6.2.154	<a href="#">virDrvDomainSnapshotDelete</a>	322
4.6.2.155	<a href="#">virDrvDomainSnapshotGetParent</a>	322
4.6.2.156	<a href="#">virDrvDomainSnapshotGetXMLDesc</a>	322
4.6.2.157	<a href="#">virDrvDomainSnapshotHasMetadata</a>	322
4.6.2.158	<a href="#">virDrvDomainSnapshotIsCurrent</a>	322
4.6.2.159	<a href="#">virDrvDomainSnapshotListAllChildren</a>	322
4.6.2.160	<a href="#">virDrvDomainSnapshotListChildrenNames</a>	323
4.6.2.161	<a href="#">virDrvDomainSnapshotListNames</a>	323
4.6.2.162	<a href="#">virDrvDomainSnapshotLookupByName</a>	323
4.6.2.163	<a href="#">virDrvDomainSnapshotNum</a>	323
4.6.2.164	<a href="#">virDrvDomainSnapshotNumChildren</a>	323
4.6.2.165	<a href="#">virDrvDomainSuspend</a>	323
4.6.2.166	<a href="#">virDrvDomainUndefine</a>	323
4.6.2.167	<a href="#">virDrvDomainUndefineFlags</a>	323
4.6.2.168	<a href="#">virDrvDomainUpdateDeviceFlags</a>	323

4.6.2.169 virDrvDrvSupportsFeature . . . . .	323
4.6.2.170 virDrvGetCapabilities . . . . .	323
4.6.2.171 virDrvGetHostname . . . . .	323
4.6.2.172 virDrvGetLibVersion . . . . .	323
4.6.2.173 virDrvGetMaxVcpus . . . . .	323
4.6.2.174 virDrvGetSysinfo . . . . .	323
4.6.2.175 virDrvGetType . . . . .	323
4.6.2.176 virDrvGetURI . . . . .	323
4.6.2.177 virDrvGetVersion . . . . .	323
4.6.2.178 virDrvInterfaceChangeBegin . . . . .	323
4.6.2.179 virDrvInterfaceChangeCommit . . . . .	323
4.6.2.180 virDrvInterfaceChangeRollback . . . . .	323
4.6.2.181 virDrvInterfaceCreate . . . . .	323
4.6.2.182 virDrvInterfaceDefineXML . . . . .	323
4.6.2.183 virDrvInterfaceDestroy . . . . .	323
4.6.2.184 virDrvInterfaceGetXMLDesc . . . . .	323
4.6.2.185 virDrvInterfaceIsActive . . . . .	323
4.6.2.186 virDrvInterfaceLookupByMACString . . . . .	324
4.6.2.187 virDrvInterfaceLookupByName . . . . .	324
4.6.2.188 virDrvInterfaceUndefine . . . . .	324
4.6.2.189 virDrvListAllDomains . . . . .	324
4.6.2.190 virDrvListAllInterfaces . . . . .	324
4.6.2.191 virDrvListAllNetworks . . . . .	324
4.6.2.192 virDrvListAllSecrets . . . . .	324
4.6.2.193 virDrvListDefinedDomains . . . . .	324
4.6.2.194 virDrvListDefinedInterfaces . . . . .	324
4.6.2.195 virDrvListDefinedNetworks . . . . .	324
4.6.2.196 virDrvListDomains . . . . .	324
4.6.2.197 virDrvListInterfaces . . . . .	324
4.6.2.198 virDrvListNetworks . . . . .	324
4.6.2.199 virDrvListSecrets . . . . .	324
4.6.2.200 virDrvNetworkCreate . . . . .	324
4.6.2.201 virDrvNetworkCreateXML . . . . .	324
4.6.2.202 virDrvNetworkDefineXML . . . . .	324
4.6.2.203 virDrvNetworkDestroy . . . . .	324
4.6.2.204 virDrvNetworkGetAutostart . . . . .	324
4.6.2.205 virDrvNetworkGetBridgeName . . . . .	324
4.6.2.206 <b>POL New</b> virDrvNetworkGetBridgeType . . . . .	324
4.6.2.207 virDrvNetworkGetXMLDesc . . . . .	324
4.6.2.208 virDrvNetworkIsActive . . . . .	324

4.6.2.209 virDrvNetworkIsPersistent . . . . .	324
4.6.2.210 virDrvNetworkLookupByName . . . . .	324
4.6.2.211 virDrvNetworkLookupByUUID . . . . .	324
4.6.2.212 virDrvNetworkSetAutostart . . . . .	324
4.6.2.213 virDrvNetworkUndefine . . . . .	324
4.6.2.214 virDrvNetworkUpdate . . . . .	325
4.6.2.215 virDrvNodeDeviceCreateXML . . . . .	325
4.6.2.216 virDrvNodeDeviceDestroy . . . . .	325
4.6.2.217 virDrvNodeDeviceDetach . . . . .	325
4.6.2.218 virDrvNodeDeviceReAttach . . . . .	325
4.6.2.219 virDrvNodeDeviceReset . . . . .	325
4.6.2.220 virDrvNodeGetCellsFreeMemory . . . . .	325
4.6.2.221 virDrvNodeGetCPUStats . . . . .	325
4.6.2.222 virDrvNodeGetFreeMemory . . . . .	325
4.6.2.223 virDrvNodeGetInfo . . . . .	325
4.6.2.224 virDrvNodeGetMemoryParameters . . . . .	325
4.6.2.225 virDrvNodeGetMemoryStats . . . . .	325
4.6.2.226 virDrvNodeGetSecurityModel . . . . .	325
4.6.2.227 virDrvNodeSetMemoryParameters . . . . .	325
4.6.2.228 virDrvNodeSuspendForDuration . . . . .	325
4.6.2.229 virDrvNumOfDefinedDomains . . . . .	325
4.6.2.230 virDrvNumOfDefinedInterfaces . . . . .	325
4.6.2.231 virDrvNumOfDefinedNetworks . . . . .	325
4.6.2.232 virDrvNumOfDomains . . . . .	325
4.6.2.233 virDrvNumOfInterfaces . . . . .	325
4.6.2.234 virDrvNumOfNetworks . . . . .	325
4.6.2.235 virDrvNumOfSecrets . . . . .	325
4.6.2.236 virDrvNWFilterDefineXML . . . . .	325
4.6.2.237 virDrvNWFilterGetXMLDesc . . . . .	325
4.6.2.238 virDrvNWFilterLookupByName . . . . .	326
4.6.2.239 virDrvNWFilterLookupByUUID . . . . .	326
4.6.2.240 virDrvNWFilterUndefine . . . . .	326
4.6.2.241 virDrvOpen . . . . .	326
4.6.2.242 virDrvSecretDefineXML . . . . .	326
4.6.2.243 virDrvSecretGetValue . . . . .	326
4.6.2.244 virDrvSecretGetXMLDesc . . . . .	326
4.6.2.245 virDrvSecretLookupByUsage . . . . .	326
4.6.2.246 virDrvSecretLookupByUUID . . . . .	326
4.6.2.247 virDrvSecretSetValue . . . . .	326
4.6.2.248 virDrvSecretUndefine . . . . .	326

4.6.2.249 virDrvSetKeepAlive . . . . .	326
4.6.2.250 virDrvStoragePoolBuild . . . . .	326
4.6.2.251 virDrvStoragePoolCreate . . . . .	326
4.6.2.252 virDrvStoragePoolCreateXML . . . . .	326
4.6.2.253 virDrvStoragePoolDefineXML . . . . .	326
4.6.2.254 virDrvStoragePoolDelete . . . . .	326
4.6.2.255 virDrvStoragePoolDestroy . . . . .	326
4.6.2.256 virDrvStoragePoolGetAutostart . . . . .	326
4.6.2.257 virDrvStoragePoolGetInfo . . . . .	326
4.6.2.258 virDrvStoragePoolGetXMLDesc . . . . .	326
4.6.2.259 virDrvStoragePoolsActive . . . . .	326
4.6.2.260 virDrvStoragePoolsPersistent . . . . .	326
4.6.2.261 virDrvStoragePoolListAllVolumes . . . . .	326
4.6.2.262 virDrvStoragePoolListVolumes . . . . .	326
4.6.2.263 virDrvStoragePoolLookupByName . . . . .	326
4.6.2.264 virDrvStoragePoolLookupByUUID . . . . .	327
4.6.2.265 virDrvStoragePoolLookupByVolume . . . . .	327
4.6.2.266 virDrvStoragePoolNumOfVolumes . . . . .	327
4.6.2.267 virDrvStoragePoolRefresh . . . . .	327
4.6.2.268 virDrvStoragePoolSetAutostart . . . . .	327
4.6.2.269 virDrvStoragePoolUndefine . . . . .	327
4.6.2.270 virDrvStorageVolCreateXML . . . . .	327
4.6.2.271 virDrvStorageVolCreateXMLFrom . . . . .	327
4.6.2.272 virDrvStorageVolDelete . . . . .	327
4.6.2.273 virDrvStorageVolDownload . . . . .	327
4.6.2.274 virDrvStorageVolGetInfo . . . . .	327
4.6.2.275 virDrvStorageVolGetPath . . . . .	327
4.6.2.276 virDrvStorageVolGetXMLDesc . . . . .	327
4.6.2.277 virDrvStorageVolLookupByKey . . . . .	327
4.6.2.278 virDrvStorageVolLookupByName . . . . .	327
4.6.2.279 virDrvStorageVolLookupByPath . . . . .	327
4.6.2.280 virDrvStorageVolResize . . . . .	327
4.6.2.281 virDrvStorageVolUpload . . . . .	327
4.6.2.282 virDrvStorageVolWipe . . . . .	327
4.6.2.283 virDrvStorageVolWipePattern . . . . .	327
4.6.2.284 virDrvStreamAbort . . . . .	327
4.6.2.285 virDrvStreamEventAddCallback . . . . .	327
4.6.2.286 virDrvStreamEventRemoveCallback . . . . .	327
4.6.2.287 virDrvStreamEventUpdateCallback . . . . .	327
4.6.2.288 virDrvStreamFinish . . . . .	327

4.6.2.289	<a href="#">virDrvStreamRecv</a>	328
4.6.2.290	<a href="#">virDrvStreamSend</a>	328
4.6.2.291	<a href="#">virInterfaceDriver</a>	328
4.6.2.292	<a href="#">virInterfaceDriverPtr</a>	328
4.6.2.293	<a href="#">virNetworkDriver</a>	328
4.6.2.294	<a href="#">virNetworkDriverPtr</a>	328
4.6.2.295	<a href="#">virNWFilterDriver</a>	328
4.6.2.296	<a href="#">virNWFilterDriverPtr</a>	328
4.6.2.297	<a href="#">virSecretDriver</a>	328
4.6.2.298	<a href="#">virSecretDriverPtr</a>	328
4.6.2.299	<a href="#">virStorageDriver</a>	328
4.6.2.300	<a href="#">virStorageDriverPtr</a>	328
4.6.2.301	<a href="#">virStreamDriver</a>	328
4.6.2.302	<a href="#">virStreamDriverPtr</a>	328
4.6.3	<a href="#">Enumeration Type Documentation</a>	328
4.6.3.1	<a href="#">anonymous enum</a>	328
4.6.3.2	<a href="#">virDrvNo</a>	328
4.6.3.3	<a href="#">virDrvOpenStatus</a>	329
4.6.4	<a href="#">Function Documentation</a>	329
4.6.4.1	<a href="#">virDriverLoadModule</a>	329
4.6.4.2	<a href="#">virDriverModuleInitialize</a>	329
4.6.4.3	<a href="#">virRegisterDeviceMonitor</a>	329
4.6.4.4	<a href="#">virRegisterDriver</a>	329
4.6.4.5	<a href="#">virRegisterInterfaceDriver</a>	329
4.6.4.6	<a href="#">virRegisterNetworkDriver</a>	329
4.6.4.7	<a href="#">virRegisterNWFilterDriver</a>	329
4.6.4.8	<a href="#">virRegisterSecretDriver</a>	330
4.6.4.9	<a href="#">virRegisterStorageDriver</a>	330
4.7	<a href="#">src/libvirt.c File Reference</a>	330
4.7.1	<a href="#">Macro Definition Documentation</a>	340
4.7.1.1	<a href="#">MAX_DRIVERS</a>	340
4.7.1.2	<a href="#">URI_ALIAS_CHARS</a>	340
4.7.1.3	<a href="#">VIR_ARG15</a>	340
4.7.1.4	<a href="#">VIR_DOMAIN_DEBUG</a>	340
4.7.1.5	<a href="#">VIR_DOMAIN_DEBUG_0</a>	341
4.7.1.6	<a href="#">VIR_DOMAIN_DEBUG_1</a>	341
4.7.1.7	<a href="#">VIR_DOMAIN_DEBUG_2</a>	341
4.7.1.8	<a href="#">VIR_DOMAIN_DEBUG_EXPAND</a>	341
4.7.1.9	<a href="#">VIR_DOMAIN_DEBUG_PASTE</a>	341
4.7.1.10	<a href="#">VIR_FROM_THIS</a>	341

4.7.1.11	<a href="#">VIR_HAS_COMMA</a>	341
4.7.1.12	<a href="#">VIR_UUID_DEBUG</a>	341
4.7.1.13	<a href="#">virLibConnError</a>	341
4.7.1.14	<a href="#">virLibDomainError</a>	341
4.7.1.15	<a href="#">virLibDomainSnapshotError</a>	342
4.7.1.16	<a href="#">virLibInterfaceError</a>	342
4.7.1.17	<a href="#">virLibNetworkError</a>	342
4.7.1.18	<a href="#">virLibNodeDeviceError</a>	342
4.7.1.19	<a href="#">virLibNWFilterError</a>	342
4.7.1.20	<a href="#">virLibSecretError</a>	342
4.7.1.21	<a href="#">virLibStoragePoolError</a>	342
4.7.1.22	<a href="#">virLibStorageVolError</a>	343
4.7.1.23	<a href="#">virLibStreamError</a>	343
4.7.2	<a href="#">Function Documentation</a>	343
4.7.2.1	<a href="#">do_open</a>	343
4.7.2.2	<a href="#">virConnectAuthCallbackDefault</a>	343
4.7.2.3	<a href="#">virConnectBaselineCPU</a>	343
4.7.2.4	<a href="#">virConnectClose</a>	343
4.7.2.5	<a href="#">virConnectCompareCPU</a>	343
4.7.2.6	<a href="#">virConnectDomainEventDeregister</a>	344
4.7.2.7	<a href="#">virConnectDomainEventDeregisterAny</a>	344
4.7.2.8	<a href="#">virConnectDomainEventRegister</a>	344
4.7.2.9	<a href="#">virConnectDomainEventRegisterAny</a>	344
4.7.2.10	<a href="#">virConnectDomainXMLFromNative</a>	345
4.7.2.11	<a href="#">virConnectDomainXMLToNative</a>	345
4.7.2.12	<a href="#">virConnectFindStoragePoolSources</a>	345
4.7.2.13	<a href="#">virConnectGetCapabilities</a>	345
4.7.2.14	<a href="#">virConnectGetConfigFile</a>	345
4.7.2.15	<a href="#">virConnectGetConfigFilePath</a>	345
4.7.2.16	<a href="#">virConnectGetDefaultURI</a>	345
4.7.2.17	<a href="#">virConnectGetHostname</a>	345
4.7.2.18	<a href="#">virConnectGetLibVersion</a>	346
4.7.2.19	<a href="#">virConnectGetMaxVcpus</a>	346
4.7.2.20	<a href="#">virConnectGetSysinfo</a>	346
4.7.2.21	<a href="#">virConnectGetType</a>	346
4.7.2.22	<a href="#">virConnectGetURI</a>	346
4.7.2.23	<a href="#">virConnectGetVersion</a>	347
4.7.2.24	<a href="#">virConnectIsAlive</a>	347
4.7.2.25	<a href="#">virConnectIsEncrypted</a>	347
4.7.2.26	<a href="#">virConnectIsSecure</a>	347

4.7.2.27	<a href="#">virConnectListAllDomains</a>	347
4.7.2.28	<a href="#">virConnectListAllInterfaces</a>	348
4.7.2.29	<a href="#">virConnectListAllNetworks</a>	348
4.7.2.30	<a href="#">virConnectListAllNodeDevices</a>	349
4.7.2.31	<a href="#">virConnectListAllNWFilters</a>	349
4.7.2.32	<a href="#">virConnectListAllSecrets</a>	349
4.7.2.33	<a href="#">virConnectListAllStoragePools</a>	350
4.7.2.34	<a href="#">virConnectListDefinedDomains</a>	350
4.7.2.35	<a href="#">virConnectListDefinedInterfaces</a>	350
4.7.2.36	<a href="#">virConnectListDefinedNetworks</a>	351
4.7.2.37	<a href="#">virConnectListDefinedStoragePools</a>	351
4.7.2.38	<a href="#">virConnectListDomains</a>	351
4.7.2.39	<a href="#">virConnectListInterfaces</a>	351
4.7.2.40	<a href="#">virConnectListNetworks</a>	352
4.7.2.41	<a href="#">virConnectListNWFilters</a>	352
4.7.2.42	<a href="#">virConnectListSecrets</a>	352
4.7.2.43	<a href="#">virConnectListStoragePools</a>	352
4.7.2.44	<a href="#">virConnectNumOfDefinedDomains</a>	352
4.7.2.45	<a href="#">virConnectNumOfDefinedInterfaces</a>	352
4.7.2.46	<a href="#">virConnectNumOfDefinedNetworks</a>	353
4.7.2.47	<a href="#">virConnectNumOfDefinedStoragePools</a>	353
4.7.2.48	<a href="#">virConnectNumOfDomains</a>	353
4.7.2.49	<a href="#">virConnectNumOfInterfaces</a>	353
4.7.2.50	<a href="#">virConnectNumOfNetworks</a>	353
4.7.2.51	<a href="#">virConnectNumOfNWFilters</a>	353
4.7.2.52	<a href="#">virConnectNumOfSecrets</a>	353
4.7.2.53	<a href="#">virConnectNumOfStoragePools</a>	354
4.7.2.54	<a href="#">virConnectOpen</a>	354
4.7.2.55	<a href="#">virConnectOpenAuth</a>	354
4.7.2.56	<a href="#">virConnectOpenFindURIAliasMatch</a>	354
4.7.2.57	<a href="#">virConnectOpenReadOnly</a>	354
4.7.2.58	<a href="#">virConnectOpenResolveURIAlias</a>	354
4.7.2.59	<a href="#">virConnectRef</a>	354
4.7.2.60	<a href="#">virConnectRegisterCloseCallback</a>	354
4.7.2.61	<a href="#">virConnectSetKeepAlive</a>	354
4.7.2.62	<a href="#">virConnectUnregisterCloseCallback</a>	355
4.7.2.63	<a href="#">virDomainAbortJob</a>	355
4.7.2.64	<a href="#">virDomainAttachDevice</a>	355
4.7.2.65	<a href="#">virDomainAttachDeviceFlags</a>	355
4.7.2.66	<a href="#">virDomainBlockCommit</a>	355

4.7.2.67	<a href="#">virDomainBlockJobAbort</a>	356
4.7.2.68	<a href="#">virDomainBlockJobSetSpeed</a>	357
4.7.2.69	<a href="#">virDomainBlockPeek</a>	357
4.7.2.70	<a href="#">virDomainBlockPull</a>	357
4.7.2.71	<a href="#">virDomainBlockRebase</a>	358
4.7.2.72	<a href="#">virDomainBlockResize</a>	359
4.7.2.73	<a href="#">virDomainBlockStats</a>	359
4.7.2.74	<a href="#">virDomainBlockStatsFlags</a>	359
4.7.2.75	<a href="#">virDomainCoreDump</a>	360
4.7.2.76	<a href="#">virDomainCreate</a>	360
4.7.2.77	<a href="#">virDomainCreateLinux</a>	360
4.7.2.78	<a href="#">virDomainCreateWithFlags</a>	360
4.7.2.79	<a href="#">virDomainCreateXML</a>	361
4.7.2.80	<a href="#">virDomainDefineXML</a>	361
4.7.2.81	<a href="#">virDomainDestroy</a>	361
4.7.2.82	<a href="#">virDomainDestroyFlags</a>	362
4.7.2.83	<a href="#">virDomainDetachDevice</a>	362
4.7.2.84	<a href="#">virDomainDetachDeviceFlags</a>	362
4.7.2.85	<a href="#">virDomainFree</a>	362
4.7.2.86	<a href="#">virDomainGetAutostart</a>	362
4.7.2.87	<a href="#">virDomainGetBlkioParameters</a>	363
4.7.2.88	<a href="#">virDomainGetBlockInfo</a>	363
4.7.2.89	<a href="#">virDomainGetBlockIoTune</a>	363
4.7.2.90	<a href="#">virDomainGetBlockJobInfo</a>	364
4.7.2.91	<a href="#">virDomainGetConnect</a>	364
4.7.2.92	<a href="#">virDomainGetControllInfo</a>	364
4.7.2.93	<a href="#">virDomainGetCPUStats</a>	364
4.7.2.94	<a href="#">virDomainGetDiskErrors</a>	365
4.7.2.95	<a href="#">virDomainGetEmulatorPinInfo</a>	365
4.7.2.96	<a href="#">virDomainGetHostname</a>	365
4.7.2.97	<a href="#">virDomainGetID</a>	366
4.7.2.98	<a href="#">virDomainGetInfo</a>	366
4.7.2.99	<a href="#">virDomainGetInterfaceParameters</a>	366
4.7.2.100	<a href="#">virDomainGetJobInfo</a>	366
4.7.2.101	<a href="#">virDomainGetMaxMemory</a>	366
4.7.2.102	<a href="#">virDomainGetMaxVcpus</a>	366
4.7.2.103	<a href="#">virDomainGetMemoryParameters</a>	367
4.7.2.104	<a href="#">virDomainGetMetadata</a>	367
4.7.2.105	<a href="#">virDomainGetName</a>	367
4.7.2.106	<a href="#">virDomainGetNumaParameters</a>	367



4.7.2.107 virDomainGetOSType . . . . .	368
4.7.2.108 virDomainGetSchedulerParameters . . . . .	368
4.7.2.109 virDomainGetSchedulerParametersFlags . . . . .	368
4.7.2.110 virDomainGetSchedulerType . . . . .	368
4.7.2.111 virDomainGetSecurityLabel . . . . .	368
4.7.2.112 virDomainGetSecurityLabelList . . . . .	369
4.7.2.113 virDomainGetState . . . . .	369
4.7.2.114 virDomainGetUUID . . . . .	369
4.7.2.115 virDomainGetUUIDString . . . . .	369
4.7.2.116 virDomainGetVcpuPinInfo . . . . .	369
4.7.2.117 virDomainGetVcpus . . . . .	369
4.7.2.118 virDomainGetVcpusFlags . . . . .	370
4.7.2.119 virDomainGetXMLDesc . . . . .	370
4.7.2.120 virDomainHasCurrentSnapshot . . . . .	370
4.7.2.121 virDomainHasManagedSaveImage . . . . .	370
4.7.2.122 virDomainInjectNMI . . . . .	371
4.7.2.123 virDomainInterfaceStats . . . . .	371
4.7.2.124 virDomainIsActive . . . . .	371
4.7.2.125 virDomainIsPersistent . . . . .	371
4.7.2.126 virDomainIsUpdated . . . . .	371
4.7.2.127 virDomainListAllSnapshots . . . . .	371
4.7.2.128 virDomainLookupByID . . . . .	372
4.7.2.129 virDomainLookupByName . . . . .	372
4.7.2.130 virDomainLookupByUUID . . . . .	372
4.7.2.131 virDomainLookupByUUIDString . . . . .	372
4.7.2.132 virDomainManagedSave . . . . .	372
4.7.2.133 virDomainManagedSaveRemove . . . . .	373
4.7.2.134 virDomainMemoryPeek . . . . .	373
4.7.2.135 virDomainMemoryStats . . . . .	373
4.7.2.136 virDomainMigrate . . . . .	374
4.7.2.137 virDomainMigrate2 . . . . .	374
4.7.2.138 virDomainMigrateBegin3 . . . . .	375
4.7.2.139 virDomainMigrateConfirm3 . . . . .	375
4.7.2.140 virDomainMigrateDirect . . . . .	375
4.7.2.141 virDomainMigrateFinish . . . . .	375
4.7.2.142 virDomainMigrateFinish2 . . . . .	375
4.7.2.143 virDomainMigrateFinish3 . . . . .	375
4.7.2.144 virDomainMigrateGetMaxSpeed . . . . .	376
4.7.2.145 virDomainMigratePeer2Peer . . . . .	376
4.7.2.146 virDomainMigratePerform . . . . .	376

4.7.2.147 virDomainMigratePerform3 . . . . .	376
4.7.2.148 virDomainMigratePrepare . . . . .	376
4.7.2.149 virDomainMigratePrepare2 . . . . .	376
4.7.2.150 virDomainMigratePrepare3 . . . . .	376
4.7.2.151 virDomainMigratePrepareTunnel . . . . .	376
4.7.2.152 virDomainMigratePrepareTunnel3 . . . . .	376
4.7.2.153 virDomainMigrateSetMaxDowntime . . . . .	376
4.7.2.154 virDomainMigrateSetMaxSpeed . . . . .	376
4.7.2.155 virDomainMigrateToURI . . . . .	377
4.7.2.156 virDomainMigrateToURI2 . . . . .	377
4.7.2.157 virDomainMigrateVersion1 . . . . .	378
4.7.2.158 virDomainMigrateVersion2 . . . . .	378
4.7.2.159 virDomainMigrateVersion3 . . . . .	378
4.7.2.160 virDomainOpenConsole . . . . .	378
4.7.2.161 virDomainOpenGraphics . . . . .	378
4.7.2.162 virDomainPinEmulator . . . . .	379
4.7.2.163 virDomainPinVcpu . . . . .	379
4.7.2.164 virDomainPinVcpuFlags . . . . .	379
4.7.2.165 virDomainPMSuspendForDuration . . . . .	380
4.7.2.166 virDomainPMWakeup . . . . .	380
4.7.2.167 virDomainReboot . . . . .	380
4.7.2.168 virDomainRef . . . . .	380
4.7.2.169 virDomainReset . . . . .	381
4.7.2.170 virDomainRestore . . . . .	381
4.7.2.171 virDomainRestoreFlags . . . . .	381
4.7.2.172 virDomainResume . . . . .	381
4.7.2.173 virDomainRevertToSnapshot . . . . .	381
4.7.2.174 virDomainSave . . . . .	382
4.7.2.175 virDomainSaveFlags . . . . .	382
4.7.2.176 virDomainSavelImageDefineXML . . . . .	383
4.7.2.177 virDomainSavelImageGetXMLDesc . . . . .	383
4.7.2.178 virDomainScreenshot . . . . .	383
4.7.2.179 virDomainSendKey . . . . .	383
4.7.2.180 virDomainSetAutostart . . . . .	383
4.7.2.181 virDomainSetBlkioParameters . . . . .	383
4.7.2.182 virDomainSetBlockIoTune . . . . .	383
4.7.2.183 virDomainSetInterfaceParameters . . . . .	384
4.7.2.184 virDomainSetMaxMemory . . . . .	384
4.7.2.185 virDomainSetMemory . . . . .	384
4.7.2.186 virDomainSetMemoryFlags . . . . .	384

4.7.2.187 virDomainSetMemoryParameters . . . . .	385
4.7.2.188 virDomainSetMetadata . . . . .	385
4.7.2.189 virDomainSetNumaParameters . . . . .	385
4.7.2.190 virDomainSetSchedulerParameters . . . . .	385
4.7.2.191 virDomainSetSchedulerParametersFlags . . . . .	386
4.7.2.192 virDomainSetVcpus . . . . .	386
4.7.2.193 virDomainSetVcpusFlags . . . . .	386
4.7.2.194 virDomainShutdown . . . . .	386
4.7.2.195 virDomainShutdownFlags . . . . .	387
4.7.2.196 virDomainSnapshotCreateXML . . . . .	387
4.7.2.197 virDomainSnapshotCurrent . . . . .	388
4.7.2.198 virDomainSnapshotDelete . . . . .	388
4.7.2.199 virDomainSnapshotFree . . . . .	388
4.7.2.200 virDomainSnapshotGetConnect . . . . .	388
4.7.2.201 virDomainSnapshotGetDomain . . . . .	389
4.7.2.202 virDomainSnapshotGetName . . . . .	389
4.7.2.203 virDomainSnapshotGetParent . . . . .	389
4.7.2.204 virDomainSnapshotGetXMLDesc . . . . .	389
4.7.2.205 virDomainSnapshotHasMetadata . . . . .	389
4.7.2.206 virDomainSnapshotIsCurrent . . . . .	389
4.7.2.207 virDomainSnapshotListAllChildren . . . . .	389
4.7.2.208 virDomainSnapshotListChildrenNames . . . . .	390
4.7.2.209 virDomainSnapshotListNames . . . . .	390
4.7.2.210 virDomainSnapshotLookupByName . . . . .	391
4.7.2.211 virDomainSnapshotNum . . . . .	391
4.7.2.212 virDomainSnapshotNumChildren . . . . .	391
4.7.2.213 virDomainSnapshotRef . . . . .	392
4.7.2.214 virDomainSuspend . . . . .	392
4.7.2.215 virDomainUndefine . . . . .	392
4.7.2.216 virDomainUndefineFlags . . . . .	392
4.7.2.217 virDomainUpdateDeviceFlags . . . . .	393
4.7.2.218 virDrvSupportsFeature . . . . .	393
4.7.2.219 virGetVersion . . . . .	393
4.7.2.220 virInitialize . . . . .	393
4.7.2.221 virInterfaceChangeBegin . . . . .	393
4.7.2.222 virInterfaceChangeCommit . . . . .	394
4.7.2.223 virInterfaceChangeRollback . . . . .	394
4.7.2.224 virInterfaceCreate . . . . .	394
4.7.2.225 virInterfaceDefineXML . . . . .	394
4.7.2.226 virInterfaceDestroy . . . . .	394

4.7.2.227 virInterfaceFree . . . . .	395
4.7.2.228 virInterfaceGetConnect . . . . .	395
4.7.2.229 virInterfaceGetMACString . . . . .	395
4.7.2.230 virInterfaceGetName . . . . .	395
4.7.2.231 virInterfaceGetXMLDesc . . . . .	395
4.7.2.232 virInterfaceIsActive . . . . .	396
4.7.2.233 virInterfaceLookupByMACString . . . . .	396
4.7.2.234 virInterfaceLookupByName . . . . .	396
4.7.2.235 virInterfaceRef . . . . .	396
4.7.2.236 virInterfaceUndefine . . . . .	396
4.7.2.237 virNetworkCreate . . . . .	396
4.7.2.238 virNetworkCreateXML . . . . .	396
4.7.2.239 virNetworkDefineXML . . . . .	397
4.7.2.240 virNetworkDestroy . . . . .	397
4.7.2.241 virNetworkFree . . . . .	397
4.7.2.242 virNetworkGetAutostart . . . . .	397
4.7.2.243 virNetworkGetBridgeName . . . . .	397
4.7.2.244 <b>POL New</b> virNetworkGetBridgeType . . . . .	397
4.7.2.245 virNetworkGetConnect . . . . .	398
4.7.2.246 virNetworkGetName . . . . .	398
4.7.2.247 virNetworkGetUUID . . . . .	398
4.7.2.248 virNetworkGetUUIDString . . . . .	398
4.7.2.249 virNetworkGetXMLDesc . . . . .	398
4.7.2.250 virNetworkIsActive . . . . .	398
4.7.2.251 virNetworkIsPersistent . . . . .	398
4.7.2.252 virNetworkLookupByName . . . . .	399
4.7.2.253 virNetworkLookupByUUID . . . . .	399
4.7.2.254 virNetworkLookupByUUIDString . . . . .	399
4.7.2.255 virNetworkRef . . . . .	399
4.7.2.256 virNetworkSetAutostart . . . . .	399
4.7.2.257 virNetworkUndefine . . . . .	399
4.7.2.258 virNetworkUpdate . . . . .	399
4.7.2.259 virNodeDeviceCreateXML . . . . .	400
4.7.2.260 virNodeDeviceDestroy . . . . .	400
4.7.2.261 virNodeDeviceDetach . . . . .	400
4.7.2.262 virNodeDeviceFree . . . . .	400
4.7.2.263 virNodeDeviceGetName . . . . .	400
4.7.2.264 virNodeDeviceGetParent . . . . .	400
4.7.2.265 virNodeDeviceGetXMLDesc . . . . .	400
4.7.2.266 virNodeDeviceListCaps . . . . .	401

4.7.2.267 virNodeDeviceLookupByName . . . . .	401
4.7.2.268 virNodeDeviceNumOfCaps . . . . .	401
4.7.2.269 virNodeDeviceReAttach . . . . .	401
4.7.2.270 virNodeDeviceRef . . . . .	401
4.7.2.271 virNodeDeviceReset . . . . .	401
4.7.2.272 virNodeGetCellsFreeMemory . . . . .	402
4.7.2.273 virNodeGetCPUStats . . . . .	402
4.7.2.274 virNodeGetFreeMemory . . . . .	402
4.7.2.275 virNodeGetInfo . . . . .	402
4.7.2.276 virNodeGetMemoryParameters . . . . .	403
4.7.2.277 virNodeGetMemoryStats . . . . .	403
4.7.2.278 virNodeGetSecurityModel . . . . .	403
4.7.2.279 virNodeListDevices . . . . .	403
4.7.2.280 virNodeNumOfDevices . . . . .	404
4.7.2.281 virNodeSetMemoryParameters . . . . .	404
4.7.2.282 virNodeSuspendForDuration . . . . .	404
4.7.2.283 virNWFilterDefineXML . . . . .	404
4.7.2.284 virNWFilterFree . . . . .	404
4.7.2.285 virNWFilterGetName . . . . .	404
4.7.2.286 virNWFilterGetUUID . . . . .	404
4.7.2.287 virNWFilterGetUUIDString . . . . .	405
4.7.2.288 virNWFilterGetXMLDesc . . . . .	405
4.7.2.289 virNWFilterLookupByName . . . . .	405
4.7.2.290 virNWFilterLookupByUUID . . . . .	405
4.7.2.291 virNWFilterLookupByUUIDString . . . . .	405
4.7.2.292 virNWFilterRef . . . . .	405
4.7.2.293 virNWFilterUndefine . . . . .	405
4.7.2.294 virRegisterDeviceMonitor . . . . .	406
4.7.2.295 virRegisterDriver . . . . .	406
4.7.2.296 virRegisterInterfaceDriver . . . . .	406
4.7.2.297 virRegisterNetworkDriver . . . . .	406
4.7.2.298 virRegisterNWFilterDriver . . . . .	406
4.7.2.299 virRegisterSecretDriver . . . . .	406
4.7.2.300 virRegisterStorageDriver . . . . .	406
4.7.2.301 virSecretDefineXML . . . . .	406
4.7.2.302 virSecretFree . . . . .	407
4.7.2.303 virSecretGetConnect . . . . .	407
4.7.2.304 virSecretGetUsageID . . . . .	407
4.7.2.305 virSecretGetUsageType . . . . .	407
4.7.2.306 virSecretGetUUID . . . . .	407

4.7.2.307 virSecretGetUUIDString . . . . .	407
4.7.2.308 virSecretGetValue . . . . .	408
4.7.2.309 virSecretGetXMLDesc . . . . .	408
4.7.2.310 virSecretLookupByUsage . . . . .	408
4.7.2.311 virSecretLookupByUUID . . . . .	408
4.7.2.312 virSecretLookupByUUIDString . . . . .	408
4.7.2.313 virSecretRef . . . . .	408
4.7.2.314 virSecretSetValue . . . . .	409
4.7.2.315 virSecretUndefine . . . . .	409
4.7.2.316 virStoragePoolBuild . . . . .	409
4.7.2.317 virStoragePoolCreate . . . . .	409
4.7.2.318 virStoragePoolCreateXML . . . . .	409
4.7.2.319 virStoragePoolDefineXML . . . . .	409
4.7.2.320 virStoragePoolDelete . . . . .	409
4.7.2.321 virStoragePoolDestroy . . . . .	410
4.7.2.322 virStoragePoolFree . . . . .	410
4.7.2.323 virStoragePoolGetAutostart . . . . .	410
4.7.2.324 virStoragePoolGetConnect . . . . .	410
4.7.2.325 virStoragePoolGetInfo . . . . .	410
4.7.2.326 virStoragePoolGetName . . . . .	410
4.7.2.327 virStoragePoolGetUUID . . . . .	411
4.7.2.328 virStoragePoolGetUUIDString . . . . .	411
4.7.2.329 virStoragePoolGetXMLDesc . . . . .	411
4.7.2.330 virStoragePoolsActive . . . . .	411
4.7.2.331 virStoragePoolsPersistent . . . . .	411
4.7.2.332 virStoragePoolListAllVolumes . . . . .	411
4.7.2.333 virStoragePoolListVolumes . . . . .	411
4.7.2.334 virStoragePoolLookupByName . . . . .	412
4.7.2.335 virStoragePoolLookupByUUID . . . . .	412
4.7.2.336 virStoragePoolLookupByUUIDString . . . . .	412
4.7.2.337 virStoragePoolLookupByVolume . . . . .	412
4.7.2.338 virStoragePoolNumOfVolumes . . . . .	412
4.7.2.339 virStoragePoolRef . . . . .	412
4.7.2.340 virStoragePoolRefresh . . . . .	412
4.7.2.341 virStoragePoolSetAutostart . . . . .	413
4.7.2.342 virStoragePoolUndefine . . . . .	413
4.7.2.343 virStorageVolCreateXML . . . . .	413
4.7.2.344 virStorageVolCreateXMLFrom . . . . .	413
4.7.2.345 virStorageVolDelete . . . . .	413
4.7.2.346 virStorageVolDownload . . . . .	413

4.7.2.347	virStorageVolFree	414
4.7.2.348	virStorageVolGetConnect	414
4.7.2.349	virStorageVolGetInfo	414
4.7.2.350	virStorageVolGetKey	414
4.7.2.351	virStorageVolGetName	414
4.7.2.352	virStorageVolGetPath	414
4.7.2.353	virStorageVolGetXMLDesc	414
4.7.2.354	virStorageVolLookupByKey	415
4.7.2.355	virStorageVolLookupByName	415
4.7.2.356	virStorageVolLookupByPath	415
4.7.2.357	virStorageVolRef	415
4.7.2.358	virStorageVolResize	415
4.7.2.359	virStorageVolUpload	415
4.7.2.360	virStorageVolWipe	416
4.7.2.361	virStorageVolWipePattern	416
4.7.2.362	virStreamAbort	416
4.7.2.363	virStreamEventAddCallback	416
4.7.2.364	virStreamEventRemoveCallback	416
4.7.2.365	virStreamEventUpdateCallback	416
4.7.2.366	virStreamFinish	417
4.7.2.367	virStreamFree	417
4.7.2.368	virStreamNew	417
4.7.2.369	virStreamRecv	417
4.7.2.370	virStreamRecvAll	418
4.7.2.371	virStreamRef	418
4.7.2.372	virStreamSend	418
4.7.2.373	virStreamSendAll	419
4.7.2.374	virTLSMutexDestroy	419
4.7.2.375	virTLSMutexInit	419
4.7.2.376	virTLSMutexLock	419
4.7.2.377	virTLSMutexUnlock	419
4.7.2.378	virTypedParameterValidateSet	419
4.7.3	Variable Documentation	419
4.7.3.1	initialized	419
4.7.3.2	virConnectAuthDefault	419
4.7.3.3	virConnectAuthPtrDefault	419
4.7.3.4	virConnectCredTypeDefault	419
4.7.3.5	virDeviceMonitorTab	420
4.7.3.6	virDeviceMonitorTabCount	420
4.7.3.7	virDriverTab	420

4.7.3.8	virDriverTabCount	420
4.7.3.9	virInterfaceDriverTab	420
4.7.3.10	virInterfaceDriverTabCount	420
4.7.3.11	virNetworkDriverTab	420
4.7.3.12	virNetworkDriverTabCount	420
4.7.3.13	virNWFilterDriverTab	420
4.7.3.14	virNWFilterDriverTabCount	420
4.7.3.15	virSecretDriverTab	420
4.7.3.16	virSecretDriverTabCount	420
4.7.3.17	virStorageDriverTab	420
4.7.3.18	virStorageDriverTabCount	420
4.7.3.19	virTLSThreadImpl	420
4.8	src/network/bridge_driver.c File Reference	420
4.8.1	Macro Definition Documentation	424
4.8.1.1	DNSMASQ_STATE_DIR	424
4.8.1.2	NETWORK_PID_DIR	424
4.8.1.3	NETWORK_STATE_DIR	424
4.8.1.4	PROC_NET_ROUTE	424
4.8.1.5	RADVD_STATE_DIR	424
4.8.1.6	SYSCTL_PATH	424
4.8.1.7	VIR_FROM_THIS	424
4.8.2	Function Documentation	424
4.8.2.1	networkActive	424
4.8.2.2	networkAddAddrToBridge	424
4.8.2.3	networkAddGeneralIp6tablesRules	424
4.8.2.4	networkAddGeneralIptablesRules	424
4.8.2.5	networkAddIpSpecificIptablesRules	424
4.8.2.6	networkAddIptablesRules	424
4.8.2.7	networkAddMasqueradingIptablesRules	424
4.8.2.8	networkAddRoutingIptablesRules	424
4.8.2.9	<b>POL Mod</b> networkAllocateActualDevice	424
4.8.2.10	networkAutostartConfigs	424
4.8.2.11	networkBridgeDummyNicName	424
4.8.2.12	networkBuildDhcpDaemonCommandLine	424
4.8.2.13	networkBuildDnsmasqArgv	424
4.8.2.14	networkBuildDnsmasqHostsfile	425
4.8.2.15	networkCheckRouteCollision	425
4.8.2.16	networkCloseNetwork	425
4.8.2.17	networkCreate	425
4.8.2.18	networkCreateInterfacePool	425



4.8.2.19	<a href="#">networkDefine</a>	425
4.8.2.20	<a href="#">networkDestroy</a>	425
4.8.2.21	<a href="#">networkDnsmasqLeaseFileNameDefault</a>	425
4.8.2.22	<a href="#">networkDriverLock</a>	425
4.8.2.23	<a href="#">networkDriverUnlock</a>	425
4.8.2.24	<a href="#">networkEnableIpForwarding</a>	425
4.8.2.25	<a href="#">networkFindActiveConfigs</a>	425
4.8.2.26	<a href="#">networkGetAutostart</a>	425
4.8.2.27	<a href="#">networkGetBridgeName</a>	425
4.8.2.28	<b>POL New</b> <a href="#">networkGetBridgeType</a>	425
4.8.2.29	<a href="#">networkGetNetworkAddress</a>	425
4.8.2.30	<a href="#">networkGetXMLDesc</a>	425
4.8.2.31	<a href="#">networkIsActive</a>	425
4.8.2.32	<a href="#">networkIsPersistent</a>	425
4.8.2.33	<a href="#">networkKillDaemon</a>	425
4.8.2.34	<a href="#">networkListAllNetworks</a>	425
4.8.2.35	<a href="#">networkListDefinedNetworks</a>	425
4.8.2.36	<a href="#">networkListNetworks</a>	425
4.8.2.37	<a href="#">networkLookupByName</a>	426
4.8.2.38	<a href="#">networkLookupByUUID</a>	426
4.8.2.39	<a href="#">networkNotifyActualDevice</a>	426
4.8.2.40	<a href="#">networkNumDefinedNetworks</a>	426
4.8.2.41	<a href="#">networkNumNetworks</a>	426
4.8.2.42	<a href="#">networkOpenNetwork</a>	426
4.8.2.43	<a href="#">networkRadvdConfContents</a>	426
4.8.2.44	<a href="#">networkRadvdConfigFileName</a>	426
4.8.2.45	<a href="#">networkRadvdConfWrite</a>	426
4.8.2.46	<a href="#">networkRadvdPidfileBasename</a>	426
4.8.2.47	<a href="#">networkRefreshDaemons</a>	426
4.8.2.48	<a href="#">networkRefreshDhcpDaemon</a>	426
4.8.2.49	<a href="#">networkRefreshRadvd</a>	426
4.8.2.50	<a href="#">networkRegister</a>	426
4.8.2.51	<a href="#">networkReleaseActualDevice</a>	426
4.8.2.52	<a href="#">networkReload</a>	426
4.8.2.53	<a href="#">networkReloadIptablesRules</a>	426
4.8.2.54	<a href="#">networkRemoveGeneralIptablesRules</a>	426
4.8.2.55	<a href="#">networkRemoveGeneralIptablesRules</a>	426
4.8.2.56	<a href="#">networkRemoveInactive</a>	426
4.8.2.57	<a href="#">networkRemoveIpspecificIptablesRules</a>	426
4.8.2.58	<a href="#">networkRemoveIptablesRules</a>	426

4.8.2.59	<a href="#">networkRemoveMasqueradingIptablesRules</a>	427
4.8.2.60	<a href="#">networkRemoveRoutingIptablesRules</a>	427
4.8.2.61	<a href="#">networkRestartDhcpDaemon</a>	427
4.8.2.62	<a href="#">networkSetAutostart</a>	427
4.8.2.63	<a href="#">networkSetIPv6Sysctl</a>	427
4.8.2.64	<a href="#">networkShutdown</a>	427
4.8.2.65	<a href="#">networkShutdownNetwork</a>	427
4.8.2.66	<a href="#">networkShutdownNetworkExternal</a>	427
4.8.2.67	<a href="#">networkShutdownNetworkExternal</a>	427
4.8.2.68	<a href="#">POL Mod networkShutdownNetworkVirtual</a>	427
4.8.2.69	<a href="#">networkStart</a>	427
4.8.2.70	<a href="#">networkStartDhcpDaemon</a>	427
4.8.2.71	<a href="#">networkStartNetwork</a>	427
4.8.2.72	<a href="#">networkStartNetworkExternal</a>	427
4.8.2.73	<a href="#">networkStartNetworkExternal</a>	427
4.8.2.74	<a href="#">POL Mod networkStartNetworkVirtual</a>	428
4.8.2.75	<a href="#">networkStartRadvd</a>	428
4.8.2.76	<a href="#">networkStartup</a>	428
4.8.2.77	<a href="#">networkUndefine</a>	428
4.8.2.78	<a href="#">networkUpdate</a>	428
4.8.2.79	<a href="#">POL Mod networkValidate</a>	428
4.8.3	<a href="#">Variable Documentation</a>	428
4.8.3.1	<a href="#">driverState</a>	428
4.8.3.2	<a href="#">networkDnsmasqLeaseFileName</a>	428
4.8.3.3	<a href="#">networkDriver</a>	429
4.8.3.4	<a href="#">networkStateDriver</a>	429
4.9	<a href="#">src/qemu/qemu_command.c File Reference</a>	429
4.9.1	<a href="#">Macro Definition Documentation</a>	432
4.9.1.1	<a href="#">IS_USB2_CONTROLLER</a>	432
4.9.1.2	<a href="#">QEMU_PCI_ADDRESS_LAST_FUNCTION</a>	433
4.9.1.3	<a href="#">QEMU_PCI_ADDRESS_LAST_SLOT</a>	433
4.9.1.4	<a href="#">QEMU_SERIAL_PARAM_ACCEPTED_CHARS</a>	433
4.9.1.5	<a href="#">VIR_FROM_THIS</a>	433
4.9.1.6	<a href="#">WANT_VALUE</a>	433
4.9.2	<a href="#">Function Documentation</a>	433
4.9.2.1	<a href="#">qemuAddRBDHost</a>	433
4.9.2.2	<a href="#">qemuAssignDeviceAliases</a>	433
4.9.2.3	<a href="#">qemuAssignDeviceControllerAlias</a>	433
4.9.2.4	<a href="#">qemuAssignDeviceDiskAlias</a>	433
4.9.2.5	<a href="#">qemuAssignDeviceDiskAliasCustom</a>	433

4.9.2.6	qemuAssignDeviceDiskAliasFixed . . . . .	433
4.9.2.7	qemuAssignDeviceDiskAliasLegacy . . . . .	433
4.9.2.8	qemuAssignDeviceHostdevAlias . . . . .	433
4.9.2.9	qemuAssignDeviceNetAlias . . . . .	433
4.9.2.10	qemuAssignDevicePCISlots . . . . .	433
4.9.2.11	qemuAssignDeviceRedirdevAlias . . . . .	433
4.9.2.12	qemuAssignSpaprVIOAddress . . . . .	433
4.9.2.13	qemuBuildChrArgStr . . . . .	433
4.9.2.14	qemuBuildChrChardevStr . . . . .	433
4.9.2.15	qemuBuildChrDeviceStr . . . . .	433
4.9.2.16	qemuBuildClockArgStr . . . . .	433
4.9.2.17	qemuBuildCommandLine . . . . .	434
4.9.2.18	qemuBuildControllerDevStr . . . . .	434
4.9.2.19	qemuBuildCpuArgStr . . . . .	434
4.9.2.20	qemuBuildDeviceAddressStr . . . . .	434
4.9.2.21	qemuBuildDriveDevStr . . . . .	434
4.9.2.22	qemuBuildDriveStr . . . . .	434
4.9.2.23	qemuBuildFSDevStr . . . . .	434
4.9.2.24	qemuBuildFSStr . . . . .	434
4.9.2.25	qemuBuildHostNetStr . . . . .	434
4.9.2.26	qemuBuildHubDevStr . . . . .	434
4.9.2.27	qemuBuildIoEventFdStr . . . . .	434
4.9.2.28	qemuBuildMachineArgStr . . . . .	434
4.9.2.29	qemuBuildMemballoonDevStr . . . . .	434
4.9.2.30	qemuBuildNicDevStr . . . . .	434
4.9.2.31	qemuBuildNicStr . . . . .	434
4.9.2.32	qemuBuildNumaArgStr . . . . .	434
4.9.2.33	qemuBuildPCIHostdevDevStr . . . . .	434
4.9.2.34	qemuBuildPCIHostdevPCIDevStr . . . . .	434
4.9.2.35	qemuBuildRBDString . . . . .	434
4.9.2.36	qemuBuildRedirdevDevStr . . . . .	434
4.9.2.37	qemuBuildRomStr . . . . .	434
4.9.2.38	qemuBuildSmbiosBiosStr . . . . .	434
4.9.2.39	qemuBuildSmbiosSystemStr . . . . .	435
4.9.2.40	qemuBuildSmpArgStr . . . . .	435
4.9.2.41	qemuBuildSoundCodecStr . . . . .	435
4.9.2.42	qemuBuildSoundDevStr . . . . .	435
4.9.2.43	qemuBuildUSBControllerDevStr . . . . .	435
4.9.2.44	qemuBuildUSBHostdevDevStr . . . . .	435
4.9.2.45	qemuBuildUSBHostdevUsbDevStr . . . . .	435

4.9.2.46	qemuBuildUSBInputDevStr	435
4.9.2.47	qemuBuildVideoDevStr	435
4.9.2.48	qemuBuildVirtioSerialPortDevStr	435
4.9.2.49	qemuBuildWatchdogDevStr	435
4.9.2.50	qemuCollectPCIAddress	435
4.9.2.51	qemuControllerModelUSBToCaps	435
4.9.2.52	qemuDeviceDriveHostAlias	435
4.9.2.53	qemuDomainAssignAddresses	435
4.9.2.54	qemuDomainAssignPCIAddresses	435
4.9.2.55	qemuDomainAssignS390Addresses	435
4.9.2.56	qemuDomainAssignSpaprVIOAddresses	435
4.9.2.57	qemuDomainDeviceAliasIndex	435
4.9.2.58	qemuDomainNetVLAN	435
4.9.2.59	qemuDomainPCIAddressCheckSlot	435
4.9.2.60	qemuDomainPCIAddressEnsureAddr	435
4.9.2.61	qemuDomainPCIAddressGetNextSlot	435
4.9.2.62	qemuDomainPCIAddressReleaseAddr	435
4.9.2.63	qemuDomainPCIAddressReleaseFunction	435
4.9.2.64	qemuDomainPCIAddressReleaseSlot	435
4.9.2.65	qemuDomainPCIAddressReserveAddr	436
4.9.2.66	qemuDomainPCIAddressReserveFunction	436
4.9.2.67	qemuDomainPCIAddressReserveSlot	436
4.9.2.68	qemuDomainPCIAddressSetCreate	436
4.9.2.69	qemuDomainPCIAddressSetFree	436
4.9.2.70	qemuDomainPCIAddressSetFreeEntry	436
4.9.2.71	qemuDomainPCIAddressSetNextAddr	436
4.9.2.72	qemuDomainPrimeS390VirtioDevices	436
4.9.2.73	qemuFindEnv	436
4.9.2.74	qemuFindNICForVLAN	436
4.9.2.75	qemuInitGuestCPU	436
4.9.2.76	<b>POL Mod</b> qemuNetworkInterfaceConnect	436
4.9.2.77	qemuOpenPCIConfig	436
4.9.2.78	qemuOpenVhostNet	436
4.9.2.79	qemuParseCommandLine	436
4.9.2.80	qemuParseCommandLineBootDevs	436
4.9.2.81	qemuParseCommandLineChr	436
4.9.2.82	qemuParseCommandLineCPU	436
4.9.2.83	qemuParseCommandLineDisk	437
4.9.2.84	qemuParseCommandLineNet	437
4.9.2.85	qemuParseCommandLinePCI	437

4.9.2.86	<a href="#">qemuParseCommandLinePid</a>	437
4.9.2.87	<a href="#">qemuParseCommandLineSmp</a>	437
4.9.2.88	<a href="#">qemuParseCommandLineString</a>	437
4.9.2.89	<a href="#">qemuParseCommandLineUSB</a>	437
4.9.2.90	<a href="#">qemuParseKeywords</a>	437
4.9.2.91	<a href="#">qemuParseProcFileStrings</a>	437
4.9.2.92	<a href="#">qemuParseRBDString</a>	437
4.9.2.93	<a href="#">qemuPCIAddressAsString</a>	437
4.9.2.94	<a href="#">qemuPhysIfaceConnect</a>	437
4.9.2.95	<a href="#">qemuSafeSerialParamValue</a>	437
4.9.2.96	<a href="#">qemuSetScsiControllerModel</a>	437
4.9.2.97	<a href="#">qemuSoundCodecTypeToCaps</a>	437
4.9.2.98	<a href="#">qemuSpaprVIOFindByReg</a>	437
4.9.2.99	<a href="#">qemuStringToArgvEnv</a>	437
4.9.2.100	<a href="#">qemuUsbId</a>	437
4.9.2.101	<a href="#">VIR_ENUM_IMPL</a>	437
4.10	<a href="#">src/uml/uml_conf.c File Reference</a>	438
4.10.1	<a href="#">Macro Definition Documentation</a>	438
4.10.1.1	<a href="#">umlLog</a>	438
4.10.1.2	<a href="#">VIR_FROM_THIS</a>	438
4.10.2	<a href="#">Function Documentation</a>	438
4.10.2.1	<a href="#">umlBuildCommandLine</a>	438
4.10.2.2	<a href="#">umlBuildCommandLineChr</a>	439
4.10.2.3	<a href="#">POL Mod umlBuildCommandLineNet</a>	439
4.10.2.4	<a href="#">umlCapsInit</a>	439
4.10.2.5	<a href="#">POL Mod umlConnectTapDevice</a>	439
4.10.2.6	<a href="#">umlDefaultConsoleType</a>	439
4.10.2.7	<a href="#">umlNextArg</a>	439
4.11	<a href="#">src/util/virnetdevbridge.c File Reference</a>	440
4.11.1	<a href="#">Macro Definition Documentation</a>	440
4.11.1.1	<a href="#">VIR_FROM_THIS</a>	440
4.11.2	<a href="#">Function Documentation</a>	440
4.11.2.1	<a href="#">virNetDevBridgeAddPort</a>	440
4.11.2.2	<a href="#">virNetDevBridgeCreate</a>	440
4.11.2.3	<a href="#">virNetDevBridgeDelete</a>	441
4.11.2.4	<a href="#">virNetDevBridgeGetSTP</a>	441
4.11.2.5	<a href="#">virNetDevBridgeGetSTPDelay</a>	441
4.11.2.6	<a href="#">virNetDevBridgeRemovePort</a>	441
4.11.2.7	<a href="#">virNetDevBridgeSetSTP</a>	441
4.11.2.8	<a href="#">virNetDevBridgeSetSTPDelay</a>	441

4.12	src/util/virnetdevopenvswitch.c File Reference	441
4.12.1	Macro Definition Documentation	442
4.12.1.1	VIR_FROM_THIS	442
4.12.2	Function Documentation	442
4.12.2.1	<b>POL Mod</b> virNetDevOpenvswitchAddPort	442
4.12.2.2	<b>POL New</b> virNetDevOpenvswitchBridgeCreate	442
4.12.2.3	<b>POL New</b> virNetDevOpenvswitchBridgeDelete	442
4.12.2.4	<b>POL New</b> virNetDevOpenvswitchBridgeGetSTP	442
4.12.2.5	<b>POL New</b> virNetDevOpenvswitchBridgeGetSTPDelay	442
4.12.2.6	<b>POL New</b> virNetDevOpenvswitchBridgeSetSTP	443
4.12.2.7	<b>POL New</b> virNetDevOpenvswitchBridgeSetSTPDelay	443
4.12.2.8	<b>POL New</b> virNetDevOpenvswitchFakeBridgeCreate	443
4.12.2.9	virNetDevOpenvswitchRemovePort	443
4.12.2.10	<b>POL New</b> virNetDevOpenvswitchTunnelCreate	443
4.12.2.11	<b>POL New</b> virNetDevOpenvswitchTunnelDestroy	443
4.13	src/util/virnetdevopenvswitch.h File Reference	443
4.13.1	Function Documentation	444
4.13.1.1	ATTRIBUTE_NONNULL	444
4.13.1.2	<b>POL New</b> virNetDevOpenvswitchBridgeCreate	444
4.13.1.3	<b>POL New</b> virNetDevOpenvswitchBridgeDelete	444
4.13.1.4	<b>POL New</b> virNetDevOpenvswitchBridgeGetSTP	445
4.13.1.5	<b>POL New</b> virNetDevOpenvswitchBridgeGetSTPDelay	445
4.13.1.6	<b>POL New</b> virNetDevOpenvswitchBridgeSetSTP	445
4.13.1.7	<b>POL New</b> virNetDevOpenvswitchBridgeSetSTPDelay	445
4.13.1.8	<b>POL New</b> virNetDevOpenvswitchFakeBridgeCreate	445
4.13.1.9	virNetDevOpenvswitchRemovePort	445
4.13.1.10	<b>POL New</b> virNetDevOpenvswitchTunnelCreate	445
4.13.1.11	<b>POL New</b> virNetDevOpenvswitchTunnelDestroy	445
4.13.2	Variable Documentation	445
4.13.2.1	ifname	445
4.13.2.2	macaddr	445
4.13.2.3	ovsport	445
4.13.2.4	vmuuid	445
4.14	src/util/virnetdevtap.c File Reference	445
4.14.1	Macro Definition Documentation	446
4.14.1.1	VIR_FROM_THIS	446
4.14.2	Function Documentation	446
4.14.2.1	virNetDevTapCreate	446

4.14.2.2	<b>POL Mod</b> virNetDevTapCreateInBridgePort	446
4.14.2.3	virNetDevTapDelete	447
4.15	src/util/virnetdevtap.h File Reference	447
4.15.1	Enumeration Type Documentation	447
4.15.1.1	virNetDevTapCreateFlags	447
4.15.2	Function Documentation	448
4.15.2.1	virNetDevTapCreate	448
4.15.2.2	<b>POL Mod</b> virNetDevTapCreateInBridgePort	448
4.15.2.3	virNetDevTapDelete	448
4.16	tools/virsh-network.c File Reference	448
4.16.1	Macro Definition Documentation	450
4.16.1.1	EDIT_DEFINE	450
4.16.1.2	EDIT_FREE	450
4.16.1.3	EDIT_GET_XML	450
4.16.1.4	EDIT_NOT_CHANGED	450
4.16.2	Typedef Documentation	450
4.16.2.1	vshNetworkListPtr	450
4.16.3	Function Documentation	450
4.16.3.1	cmdNetworkAutostart	450
4.16.3.2	cmdNetworkCreate	450
4.16.3.3	cmdNetworkDefine	450
4.16.3.4	cmdNetworkDestroy	450
4.16.3.5	cmdNetworkDumpXML	450
4.16.3.6	cmdNetworkEdit	450
4.16.3.7	cmdNetworkInfo	451
4.16.3.8	cmdNetworkList	451
4.16.3.9	cmdNetworkName	451
4.16.3.10	cmdNetworkStart	451
4.16.3.11	cmdNetworkUndefine	451
4.16.3.12	cmdNetworkUpdate	451
4.16.3.13	cmdNetworkUuid	451
4.16.3.14	VIR_ENUM_IMPL	451
4.16.3.15	VIR_ENUM_IMPL	451
4.16.3.16	vshCommandOptNetworkBy	451
4.16.3.17	vshNetworkGetXMLDesc	451
4.16.3.18	vshNetworkListCollect	451
4.16.3.19	vshNetworkListFree	451
4.16.3.20	vshNetworkSorter	451
4.16.4	Variable Documentation	451
4.16.4.1	info_network_autostart	451

4.16.4.2	info_network_create	451
4.16.4.3	info_network_define	451
4.16.4.4	info_network_destroy	452
4.16.4.5	info_network_dumpxml	452
4.16.4.6	info_network_edit	452
4.16.4.7	info_network_info	452
4.16.4.8	info_network_list	452
4.16.4.9	info_network_name	453
4.16.4.10	info_network_start	453
4.16.4.11	info_network_undefine	453
4.16.4.12	info_network_update	453
4.16.4.13	info_network_uuid	453
4.16.4.14	networkCmds	453
4.16.4.15	opts_network_autostart	454
4.16.4.16	opts_network_create	454
4.16.4.17	opts_network_define	454
4.16.4.18	opts_network_destroy	454
4.16.4.19	opts_network_dumpxml	455
4.16.4.20	opts_network_edit	455
4.16.4.21	opts_network_info	455
4.16.4.22	opts_network_list	455
4.16.4.23	opts_network_name	455
4.16.4.24	opts_network_start	455
4.16.4.25	opts_network_undefine	456
4.16.4.26	opts_network_update	456
4.16.4.27	opts_network_uuid	456



# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">_gemuDomainPCIAddressSet</a>	7
<a href="#">_virBlkioDeviceWeight</a>	7
<a href="#">_virConnectAuth</a>	8
<a href="#">_virConnectCredential</a>	8
<a href="#">_virDeviceMonitor</a>	9
<a href="#">_virDomainActualNetDef</a>	10
<a href="#">_virDomainBIOSDef</a>	11
<a href="#">_virDomainBlockInfo</a>	11
<a href="#">_virDomainBlockIoTuneInfo</a>	12
<a href="#">_virDomainBlockJobInfo</a>	12
<a href="#">_virDomainBlockStats</a>	13
<a href="#">_virDomainChrDef</a>	13
<a href="#">_virDomainChrSourceDef</a>	14
<a href="#">_virDomainClockDef</a>	16
<a href="#">_virDomainControllInfo</a>	16
<a href="#">_virDomainControllerDef</a>	17
<a href="#">_virDomainDef</a>	17
<a href="#">_virDomainDeviceCcidAddress</a>	23
<a href="#">_virDomainDeviceDef</a>	23
<a href="#">_virDomainDeviceDriveAddress</a>	24
<a href="#">_virDomainDeviceInfo</a>	25
<a href="#">_virDomainDeviceSpaprVioAddress</a>	26
<a href="#">_virDomainDeviceUSBAddress</a>	26
<a href="#">_virDomainDeviceUSBMaster</a>	27
<a href="#">_virDomainDeviceVirtioSerialAddress</a>	27
<a href="#">_virDomainDiskDef</a>	27
<a href="#">_virDomainDiskError</a>	30
<a href="#">_virDomainDiskHostDef</a>	31
<a href="#">_virDomainEventGraphicsAddress</a>	31
<a href="#">_virDomainEventGraphicsSubject</a>	32
<a href="#">_virDomainEventGraphicsSubjectIdentity</a>	32
<a href="#">_virDomainFSDef</a>	33
<a href="#">_virDomainGraphicsAuthDef</a>	34
<a href="#">_virDomainGraphicsDef</a>	34
<a href="#">_virDomainGraphicsListenDef</a>	36
<a href="#">_virDomainHostdevDef</a>	37
<a href="#">_virDomainHostdevOrigStates</a>	38
<a href="#">_virDomainHostdevSubsys</a>	38

<a href="#">_virDomainHubDef</a>	39
<a href="#">_virDomainInfo</a>	39
<a href="#">_virDomainInputDef</a>	40
<a href="#">_virDomainInterfaceStats</a>	40
<a href="#">_virDomainJobInfo</a>	41
<a href="#">_virDomainLeaseDef</a>	42
<a href="#">_virDomainMemballoonDef</a>	42
<a href="#">_virDomainMemoryStat</a>	43
<a href="#">_virDomainNetDef</a>	43
<a href="#">_virDomainNumatuneDef</a>	46
<a href="#">_virDomainObj</a>	46
<a href="#">_virDomainObjList</a>	47
<a href="#">_virDomainOSDef</a>	48
<a href="#">_virDomainRedirdevDef</a>	49
<a href="#">_virDomainRedirFilterDef</a>	50
<a href="#">_virDomainRedirFilterUsbDevDef</a>	50
<a href="#">_virDomainSmartcardDef</a>	50
<a href="#">_virDomainSoundCodecDef</a>	51
<a href="#">_virDomainSoundDef</a>	52
<a href="#">_virDomainStateReason</a>	52
<a href="#">_virDomainTimerCatchupDef</a>	52
<a href="#">_virDomainTimerDef</a>	53
<a href="#">_virDomainVcpuPinDef</a>	54
<a href="#">_virDomainVideoAccelDef</a>	54
<a href="#">_virDomainVideoDef</a>	54
<a href="#">_virDomainVirtioSerialOpts</a>	55
<a href="#">_virDomainWatchdogDef</a>	55
<a href="#">_virDriver</a>	56
<a href="#">_virInterfaceDriver</a>	66
<a href="#">_virNetworkDef</a>	67
<a href="#">_virNetworkDHCPHostDef</a>	69
<a href="#">_virNetworkDHCPRangeDef</a>	69
<a href="#">_virNetworkDNSDef</a>	70
<a href="#">_virNetworkDNSHostsDef</a>	70
<a href="#">_virNetworkDNSSrvRecordsDef</a>	71
<a href="#">_virNetworkDNSTxtRecordsDef</a>	72
<a href="#">_virNetworkDriver</a>	72
<a href="#">_virNetworkForwardIfDef</a>	74
<a href="#">_virNetworkForwardPfDef</a>	74
<a href="#">_virNetworkIpDef</a>	75
<a href="#">_virNetworkObj</a>	76
<a href="#">_virNetworkObjList</a>	76
<a href="#">_virNodeCPUStats</a>	77
<a href="#">_virNodeInfo</a>	77
<a href="#">_virNodeMemoryStats</a>	78
<a href="#">_virNWFilterDriver</a>	78
<a href="#">_virPortGroupDef</a>	79
<a href="#">_virSecretDriver</a>	80
<a href="#">_virSecurityDeviceLabelDef</a>	81
<a href="#">_virSecurityLabel</a>	81
<a href="#">_virSecurityLabelDef</a>	82
<a href="#">_virSecurityModel</a>	83
<a href="#">_virStorageDriver</a>	83
<a href="#">_virStoragePoolInfo</a>	86
<a href="#">_virStorageVollInfo</a>	86
<a href="#">_virStreamDriver</a>	87
<a href="#">_virTunnelDef</a>	88
<a href="#">_virTypedParameter</a>	88

<a href="#">_virVcpuInfo</a>	89
<a href="#">network_driver</a>	89
<a href="#">virDomainIDData</a>	90
<a href="#">virDomainListData</a>	90
<a href="#">virDomainNameData</a>	91
<a href="#">vshNetworkList</a>	91



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

include/libvirt/libvirt.h . . . . .	93
src/driver.h . . . . .	307
src/libvirt.c . . . . .	330
src/conf/domain_conf.c . . . . .	227
src/conf/domain_conf.h . . . . .	251
src/conf/network_conf.c . . . . .	293
src/conf/network_conf.h . . . . .	301
src/network/bridge_driver.c . . . . .	420
src/qemu/qemu_command.c . . . . .	429
src/uml/uml_conf.c . . . . .	438
src/util/virnetdevbridge.c . . . . .	440
src/util/virnetdevopenvswitch.c . . . . .	441
src/util/virnetdevopenvswitch.h . . . . .	443
src/util/virnetdevtap.c . . . . .	445
src/util/virnetdevtap.h . . . . .	447
tools/virsh-network.c . . . . .	448



## Chapter 3

# Data Structure Documentation

### 3.1 `_qemuDomainPCIAAddressSet` Struct Reference

#### Data Fields

- `virHashTablePtr` [used](#)
- `int` [nextslot](#)

#### 3.1.1 Field Documentation

##### 3.1.1.1 `int nextslot`

##### 3.1.1.2 `virHashTablePtr used`

The documentation for this struct was generated from the following file:

- `src/qemu/qemu_command.c`

### 3.2 `_virBlkioDeviceWeight` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- `char *` [path](#)
- `unsigned int` [weight](#)

#### 3.2.1 Field Documentation

##### 3.2.1.1 `char* path`

##### 3.2.1.2 `unsigned int weight`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

### 3.3 `_virConnectAuth` Struct Reference

```
#include <libvirt.h>
```

#### Data Fields

- int \* [credtype](#)
- unsigned int [ncredtype](#)
- [virConnectAuthCallbackPtr](#) cb
- void \* [cbdata](#)

#### 3.3.1 Field Documentation

3.3.1.1 `virConnectAuthCallbackPtr` cb

3.3.1.2 `void*` cbdata

3.3.1.3 `int*` credtype

3.3.1.4 `unsigned int` ncredtype

The documentation for this struct was generated from the following file:

- `include/libvirt/libvirt.h`

### 3.4 `_virConnectCredential` Struct Reference

```
#include <libvirt.h>
```

#### Data Fields

- int [type](#)
- const char \* [prompt](#)
- const char \* [challenge](#)
- const char \* [defresult](#)
- char \* [result](#)
- unsigned int [resultlen](#)

#### 3.4.1 Field Documentation

3.4.1.1 `const char*` challenge

3.4.1.2 `const char*` defresult

3.4.1.3 `const char*` prompt

3.4.1.4 `char*` result

3.4.1.5 `unsigned int` resultlen



#### 3.4.1.6 `int` type

The documentation for this struct was generated from the following file:

- `include/libvirt/libvirt.h`

## 3.5 `_virDeviceMonitor` Struct Reference

```
#include <driver.h>
```

### Data Fields

- `const char * name`
- `virDrvOpen open`
- `virDrvClose close`
- `virDevMonNumOfDevices numOfDevices`
- `virDevMonListDevices listDevices`
- `virDevMonListAllNodeDevices listAllNodeDevices`
- `virDevMonDeviceLookupByName deviceLookupByName`
- `virDevMonDeviceGetXMLDesc deviceGetXMLDesc`
- `virDevMonDeviceGetParent deviceGetParent`
- `virDevMonDeviceNumOfCaps deviceNumOfCaps`
- `virDevMonDeviceListCaps deviceListCaps`
- `virDrvNodeDeviceCreateXML deviceCreateXML`
- `virDrvNodeDeviceDestroy deviceDestroy`

### 3.5.1 Detailed Description

`_virDeviceMonitor`:

Structure associated with monitoring the devices on a virtualized node.

### 3.5.2 Field Documentation

3.5.2.1 `virDrvClose close`

3.5.2.2 `virDrvNodeDeviceCreateXML deviceCreateXML`

3.5.2.3 `virDrvNodeDeviceDestroy deviceDestroy`

3.5.2.4 `virDevMonDeviceGetParent deviceGetParent`

3.5.2.5 `virDevMonDeviceGetXMLDesc deviceGetXMLDesc`

3.5.2.6 `virDevMonDeviceListCaps deviceListCaps`

3.5.2.7 `virDevMonDeviceLookupByName deviceLookupByName`

3.5.2.8 `virDevMonDeviceNumOfCaps deviceNumOfCaps`

3.5.2.9 `virDevMonListAllNodeDevices listAllNodeDevices`

### 3.5.2.10 virDevMonListDevices listDevices

### 3.5.2.11 const char\* name

### 3.5.2.12 virDevMonNumOfDevices numOfDevices

### 3.5.2.13 virDrvOpen open

The documentation for this struct was generated from the following file:

- src/[driver.h](#)

## 3.6 \_virDomainActualNetDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [type](#)
  - union {
    - struct {
      - char \* [brname](#)
      - char \* [brtype](#) POL New
    - [bridge](#)
    - struct {
      - char \* [linkdev](#)
      - int [mode](#)
    - [direct](#)
    - struct {
      - [virDomainHostdevDef](#) [def](#)
      - [hostdev](#)
  - [data](#)
- virNetDevVPortProfilePtr [virtPortProfile](#)
- virNetDevBandwidthPtr [bandwidth](#)
- virNetDevVlan [vlan](#)

### 3.6.1 Field Documentation

#### 3.6.1.1 virNetDevBandwidthPtr bandwidth

#### 3.6.1.2 struct { ... } bridge

#### 3.6.1.3 char\* brname

#### 3.6.1.4 char\* brtype

#### 3.6.1.5 union { ... } data

#### 3.6.1.6 virDomainHostdevDef def

#### 3.6.1.7 struct { ... } direct

3.6.1.8 struct { ... } hostdev

3.6.1.9 char\* linkdev

3.6.1.10 int mode

3.6.1.11 int type

3.6.1.12 virNetDevVPortProfilePtr virtPortProfile

3.6.1.13 virNetDevVlan vlan

The documentation for this struct was generated from the following file:

- src/conf/[domain\\_conf.h](#)

## 3.7 \_virDomainBIOSDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [useserial](#)
- bool [rt\\_set](#)
- int [rt\\_delay](#)

### 3.7.1 Field Documentation

3.7.1.1 int rt\_delay

3.7.1.2 bool rt\_set

3.7.1.3 int useserial

The documentation for this struct was generated from the following file:

- src/conf/[domain\\_conf.h](#)

## 3.8 \_virDomainBlockInfo Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- unsigned long long [capacity](#)
- unsigned long long [allocation](#)
- unsigned long long [physical](#)

### 3.8.1 Field Documentation

3.8.1.1 unsigned long long allocation

3.8.1.2 unsigned long long capacity

3.8.1.3 unsigned long long physical

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.9 `_virDomainBlockIoTuneInfo` Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- unsigned long long [total\\_bytes\\_sec](#)
- unsigned long long [read\\_bytes\\_sec](#)
- unsigned long long [write\\_bytes\\_sec](#)
- unsigned long long [total\\_iops\\_sec](#)
- unsigned long long [read\\_iops\\_sec](#)
- unsigned long long [write\\_iops\\_sec](#)

### 3.9.1 Field Documentation

3.9.1.1 unsigned long long read\_bytes\_sec

3.9.1.2 unsigned long long read\_iops\_sec

3.9.1.3 unsigned long long total\_bytes\_sec

3.9.1.4 unsigned long long total\_iops\_sec

3.9.1.5 unsigned long long write\_bytes\_sec

3.9.1.6 unsigned long long write\_iops\_sec

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.10 `_virDomainBlockJobInfo` Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- [virDomainBlockJobType](#) type
- unsigned long [bandwidth](#)

- [virDomainBlockJobCursor cur](#)
- [virDomainBlockJobCursor end](#)

### 3.10.1 Field Documentation

3.10.1.1 unsigned long bandwidth

3.10.1.2 virDomainBlockJobCursor cur

3.10.1.3 virDomainBlockJobCursor end

3.10.1.4 virDomainBlockJobType type

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.11 \_virDomainBlockStats Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- long long [rd\\_req](#)
- long long [rd\\_bytes](#)
- long long [wr\\_req](#)
- long long [wr\\_bytes](#)
- long long [errs](#)

### 3.11.1 Field Documentation

3.11.1.1 long long errs

3.11.1.2 long long rd\_bytes

3.11.1.3 long long rd\_req

3.11.1.4 long long wr\_bytes

3.11.1.5 long long wr\_req

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.12 \_virDomainChrDef Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- int [deviceType](#)
- int [targetType](#)
- union {
  - int [port](#)
  - virSocketAddrPtr [addr](#)
  - char \* [name](#)
 } [target](#)
- [virDomainChrSourceDef](#) [source](#)
- [virDomainDeviceInfo](#) [info](#)
- size\_t [nseclabels](#)
- [virSecurityDeviceLabelDefPtr](#) \* [seclabels](#)

### 3.12.1 Field Documentation

3.12.1.1 [virSocketAddrPtr](#) [addr](#)

3.12.1.2 [int](#) [deviceType](#)

3.12.1.3 [virDomainDeviceInfo](#) [info](#)

3.12.1.4 [char\\*](#) [name](#)

3.12.1.5 [size\\_t](#) [nseclabels](#)

3.12.1.6 [int](#) [port](#)

3.12.1.7 [virSecurityDeviceLabelDefPtr\\*](#) [seclabels](#)

3.12.1.8 [virDomainChrSourceDef](#) [source](#)

3.12.1.9 [union { ... }](#) [target](#)

3.12.1.10 [int](#) [targetType](#)

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.13 [\\_virDomainChrSourceDef](#) Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- int [type](#)
- union {
  - struct {
    - char \* [path](#)
 } [file](#)
  - struct {
    - char \* [host](#)

```
    char * service
    bool listen
    int protocol
} tcp
struct {
    char * bindHost
    char * bindService
    char * connectHost
    char * connectService
} udp
struct {
    char * path
    bool listen
} nix
int spicevmc
} data
```

### 3.13.1 Field Documentation

3.13.1.1 `char* bindHost`

3.13.1.2 `char* bindService`

3.13.1.3 `char* connectHost`

3.13.1.4 `char* connectService`

3.13.1.5 `union { ... } data`

3.13.1.6 `struct { ... } file`

3.13.1.7 `char* host`

3.13.1.8 `bool listen`

3.13.1.9 `struct { ... } nix`

3.13.1.10 `char* path`

3.13.1.11 `int protocol`

3.13.1.12 `char* service`

3.13.1.13 `int spicevmc`

3.13.1.14 `struct { ... } tcp`

3.13.1.15 `int type`

3.13.1.16 `struct { ... } udp`

The documentation for this struct was generated from the following file:

- `src/conf/domain\_conf.h`

### 3.14 `_virDomainClockDef` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- int [offset](#)
- union {
  - int [utc\\_reset](#)
  - struct {
    - long long [adjustment](#)
    - int [basis](#)
  - } [variable](#)
  - char \* [timezone](#)
- } [data](#)
- int [ntimers](#)
- [virDomainTimerDefPtr](#) \* [timers](#)

#### 3.14.1 Field Documentation

3.14.1.1 `long long adjustment`

3.14.1.2 `int basis`

3.14.1.3 `union { ... } data`

3.14.1.4 `int ntimers`

3.14.1.5 `int offset`

3.14.1.6 `virDomainTimerDefPtr* timers`

3.14.1.7 `char* timezone`

3.14.1.8 `int utc_reset`

3.14.1.9 `struct { ... } variable`

The documentation for this struct was generated from the following file:

- `src/conf/domain\_conf.h`

### 3.15 `_virDomainControllInfo` Struct Reference

```
#include <libvirt.h>
```

#### Data Fields

- unsigned int [state](#)
- unsigned int [details](#)
- unsigned long long [stateTime](#)



### 3.15.1 Field Documentation

#### 3.15.1.1 unsigned int details

#### 3.15.1.2 unsigned int state

#### 3.15.1.3 unsigned long long stateTime

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.16 \_virDomainControllerDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [type](#)
- int [idx](#)
- int [model](#)
- union {
  - [virDomainVirtioSerialOpts](#) [vioserial](#)
  - [opts](#)
- [virDomainDeviceInfo](#) [info](#)

### 3.16.1 Field Documentation

#### 3.16.1.1 int idx

#### 3.16.1.2 virDomainDeviceInfo info

#### 3.16.1.3 int model

#### 3.16.1.4 union { ... } opts

#### 3.16.1.5 int type

#### 3.16.1.6 virDomainVirtioSerialOpts vioserial

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.17 \_virDomainDef Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- int [virtType](#)
- int [id](#)
- unsigned char [uuid](#) [[VIR\\_UUID\\_BUFLLEN](#)]
- char \* [name](#)
- char \* [title](#)
- char \* [description](#)
- struct {
  - unsigned int [weight](#)
  - size\_t [ndevices](#)
  - [virBlkioDeviceWeightPtr](#) [devices](#)
- } [blkio](#)
- struct {
  - unsigned long long [max\\_balloon](#)
  - unsigned long long [cur\\_balloon](#)
  - bool [hugepage\\_backed](#)
  - int [dump\\_core](#)
  - unsigned long long [hard\\_limit](#)
  - unsigned long long [soft\\_limit](#)
  - unsigned long long [min\\_guarantee](#)
  - unsigned long long [swap\\_hard\\_limit](#)
- } [mem](#)
- unsigned short [vcpus](#)
- unsigned short [maxvcpus](#)
- int [placement\\_mode](#)
- virBitmapPtr [cpumask](#)
- struct {
  - unsigned long [shares](#)
  - unsigned long long [period](#)
  - long long [quota](#)
  - unsigned long long [emulator\\_period](#)
  - long long [emulator\\_quota](#)
  - int [nvcupin](#)
  - [virDomainVcpuPinDefPtr](#) \* [vcupin](#)
  - [virDomainVcpuPinDefPtr](#) [emulatorpin](#)
- } [cputune](#)
- [virDomainNumatuneDef](#) [numatune](#)
- int [onReboot](#)
- int [onPoweroff](#)
- int [onCrash](#)
- struct {
  - int [s3](#)
  - int [s4](#)
- } [pm](#)
- [virDomainOSDef](#) [os](#)
- char \* [emulator](#)
- int [features](#)
- int [apic\\_eoi](#)
- [virDomainClockDef](#) [clock](#)
- int [ngraphics](#)
- [virDomainGraphicsDefPtr](#) \* [graphics](#)
- int [ndisks](#)

- `virDomainDiskDefPtr` \* `disks`
- `int` `ncontrollers`
- `virDomainControllerDefPtr` \* `controllers`
- `int` `nfss`
- `virDomainFSDefPtr` \* `fss`
- `int` `nnets`
- `virDomainNetDefPtr` \* `nets`
- `int` `ninputs`
- `virDomainInputDefPtr` \* `inputs`
- `int` `nsounds`
- `virDomainSoundDefPtr` \* `sounds`
- `int` `nvideos`
- `virDomainVideoDefPtr` \* `videos`
- `int` `nhostdevs`
- `virDomainHostdevDefPtr` \* `hostdevs`
- `int` `nredirdevs`
- `virDomainRedirdevDefPtr` \* `redirdevs`
- `int` `nsmartcards`
- `virDomainSmartcardDefPtr` \* `smartcards`
- `int` `nserials`
- `virDomainChrDefPtr` \* `serials`
- `int` `nparallels`
- `virDomainChrDefPtr` \* `parallels`
- `int` `nchannels`
- `virDomainChrDefPtr` \* `channels`
- `int` `nconsoles`
- `virDomainChrDefPtr` \* `consoles`
- `size_t` `nleases`
- `virDomainLeaseDefPtr` \* `leases`
- `int` `nhubs`
- `virDomainHubDefPtr` \* `hubs`
- `size_t` `nseclabels`
- `virSecurityLabelDefPtr` \* `seclabels`
- `virDomainWatchdogDefPtr` `watchdog`
- `virDomainMemballoonDefPtr` `memballoon`
- `virCPUDefPtr` `cpu`
- `virSysinfoDefPtr` `sysinfo`
- `virDomainRedirFilterDefPtr` `redirfilter`
- `void` \* `namespaceData`
- `virDomainXMLNamespace` `ns`
- `xmlNodePtr` `metadata`

### 3.17.1 Field Documentation

3.17.1.1 `int` `apic_eoi`

3.17.1.2 `struct { ... }` `blkio`

3.17.1.3 `virDomainChrDefPtr`\* `channels`

3.17.1.4 `virDomainClockDef` `clock`

3.17.1.5 `virDomainChrDefPtr`\* `consoles`

- 3.17.1.6 **virDomainControllerDefPtr\*** controllers
- 3.17.1.7 **virCPUDefPtr** cpu
- 3.17.1.8 **virBitmapPtr** cpumask
- 3.17.1.9 **struct { ... }** cputune
- 3.17.1.10 **unsigned long long** cur\_balloon
- 3.17.1.11 **char\*** description
- 3.17.1.12 **virBlkioDeviceWeightPtr** devices
- 3.17.1.13 **virDomainDiskDefPtr\*** disks
- 3.17.1.14 **int** dump\_core
- 3.17.1.15 **char\*** emulator
- 3.17.1.16 **unsigned long long** emulator\_period
- 3.17.1.17 **long long** emulator\_quota
- 3.17.1.18 **virDomainVcpuPinDefPtr** emulatorpin
- 3.17.1.19 **int** features
- 3.17.1.20 **virDomainFSDefPtr\*** fss
- 3.17.1.21 **virDomainGraphicsDefPtr\*** graphics
- 3.17.1.22 **unsigned long long** hard\_limit
- 3.17.1.23 **virDomainHostdevDefPtr\*** hostdevs
- 3.17.1.24 **virDomainHubDefPtr\*** hubs
- 3.17.1.25 **bool** hugepage\_backed
- 3.17.1.26 **int** id
- 3.17.1.27 **virDomainInputDefPtr\*** inputs
- 3.17.1.28 **virDomainLeaseDefPtr\*** leases
- 3.17.1.29 **unsigned long long** max\_balloon
- 3.17.1.30 **unsigned short** maxvcpus
- 3.17.1.31 **struct { ... }** mem
- 3.17.1.32 **virDomainMemballoonDefPtr** memballoon
- 3.17.1.33 **xmlNodePtr** metadata

- 3.17.1.34 unsigned long long min\_guarantee
- 3.17.1.35 char\* name
- 3.17.1.36 void\* namespaceData
- 3.17.1.37 int nchannels
- 3.17.1.38 int nconsoles
- 3.17.1.39 int ncontrollers
- 3.17.1.40 size\_t ndevices
- 3.17.1.41 int ndisks
- 3.17.1.42 virDomainNetDefPtr\* nets
- 3.17.1.43 int nfss
- 3.17.1.44 int ngraphics
- 3.17.1.45 int nhostdevs
- 3.17.1.46 int nhubs
- 3.17.1.47 int ninputs
- 3.17.1.48 size\_t nleases
- 3.17.1.49 int nnets
- 3.17.1.50 int nparallels
- 3.17.1.51 int nredirdevs
- 3.17.1.52 virDomainXMLNamespace ns
- 3.17.1.53 size\_t nseclabels
- 3.17.1.54 int nserials
- 3.17.1.55 int nsmartcards
- 3.17.1.56 int nsounds
- 3.17.1.57 virDomainNumatuneDef numatune
- 3.17.1.58 int nvcpupin
- 3.17.1.59 int nvideos
- 3.17.1.60 int onCrash
- 3.17.1.61 int onPoweroff

- 3.17.1.62 `int onReboot`
- 3.17.1.63 `virDomainOSDef os`
- 3.17.1.64 `virDomainChrDefPtr* parallels`
- 3.17.1.65 `unsigned long long period`
- 3.17.1.66 `int placement_mode`
- 3.17.1.67 `struct { ... } pm`
- 3.17.1.68 `long long quota`
- 3.17.1.69 `virDomainRedirdevDefPtr* redirdevs`
- 3.17.1.70 `virDomainRedirFilterDefPtr redirfilter`
- 3.17.1.71 `int s3`
- 3.17.1.72 `int s4`
- 3.17.1.73 `virSecurityLabelDefPtr* seclabels`
- 3.17.1.74 `virDomainChrDefPtr* serials`
- 3.17.1.75 `unsigned long shares`
- 3.17.1.76 `virDomainSmartcardDefPtr* smartcards`
- 3.17.1.77 `unsigned long long soft_limit`
- 3.17.1.78 `virDomainSoundDefPtr* sounds`
- 3.17.1.79 `unsigned long long swap_hard_limit`
- 3.17.1.80 `virSysinfoDefPtr sysinfo`
- 3.17.1.81 `char* title`
- 3.17.1.82 `unsigned char uuid[VIR_UUID_BUFLen]`
- 3.17.1.83 `virDomainVcpuPinDefPtr* vcpupin`
- 3.17.1.84 `unsigned short vcpus`
- 3.17.1.85 `virDomainVideoDefPtr* videos`
- 3.17.1.86 `int virtType`
- 3.17.1.87 `virDomainWatchdogDefPtr watchdog`
- 3.17.1.88 `unsigned int weight`

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.18 \_virDomainDeviceCcidAddress Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- unsigned int [controller](#)
- unsigned int [slot](#)

### 3.18.1 Field Documentation

3.18.1.1 unsigned int controller

3.18.1.2 unsigned int slot

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.19 \_virDomainDeviceDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [type](#)
- union {
  - [virDomainDiskDefPtr](#) disk
  - [virDomainControllerDefPtr](#) controller
  - [virDomainLeaseDefPtr](#) lease
  - [virDomainFSDefPtr](#) fs
  - [virDomainNetDefPtr](#) net
  - [virDomainInputDefPtr](#) input
  - [virDomainSoundDefPtr](#) sound
  - [virDomainVideoDefPtr](#) video
  - [virDomainHostdevDefPtr](#) hostdev
  - [virDomainWatchdogDefPtr](#) watchdog
  - [virDomainGraphicsDefPtr](#) graphics
  - [virDomainHubDefPtr](#) hub
  - [virDomainRedirdevDefPtr](#) redirdev
  - [virDomainSmartcardDefPtr](#) smartcard
  - [virDomainChrDefPtr](#) chr
  - [virDomainMemballoonDefPtr](#) memballoon
- } data

### 3.19.1 Field Documentation

3.19.1.1 [virDomainChrDefPtr](#) chr

3.19.1.2 [virDomainControllerDefPtr](#) controller

- 3.19.1.3 `union { ... } data`
- 3.19.1.4 `virDomainDiskDefPtr disk`
- 3.19.1.5 `virDomainFSDefPtr fs`
- 3.19.1.6 `virDomainGraphicsDefPtr graphics`
- 3.19.1.7 `virDomainHostdevDefPtr hostdev`
- 3.19.1.8 `virDomainHubDefPtr hub`
- 3.19.1.9 `virDomainInputDefPtr input`
- 3.19.1.10 `virDomainLeaseDefPtr lease`
- 3.19.1.11 `virDomainMemballoonDefPtr memballoon`
- 3.19.1.12 `virDomainNetDefPtr net`
- 3.19.1.13 `virDomainRedirdevDefPtr redirdev`
- 3.19.1.14 `virDomainSmartcardDefPtr smartcard`
- 3.19.1.15 `virDomainSoundDefPtr sound`
- 3.19.1.16 `int type`
- 3.19.1.17 `virDomainVideoDefPtr video`
- 3.19.1.18 `virDomainWatchdogDefPtr watchdog`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

## 3.20 `_virDomainDeviceDriveAddress` Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- unsigned int `controller`
- unsigned int `bus`
- unsigned int `target`
- unsigned int `unit`

### 3.20.1 Field Documentation

3.20.1.1 unsigned int `bus`

3.20.1.2 unsigned int `controller`



3.20.1.3 unsigned int target

3.20.1.4 unsigned int unit

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.21 \_virDomainDeviceInfo Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- char \* [alias](#)
- int [type](#)
- union {
  - [virDevicePCIAddress](#) [pci](#)
  - [virDomainDeviceDriveAddress](#) [drive](#)
  - [virDomainDeviceVirtioSerialAddress](#) [vioserial](#)
  - [virDomainDeviceCcidAddress](#) [ccid](#)
  - [virDomainDeviceUSBAddress](#) [usb](#)
  - [virDomainDeviceSpaprVioAddress](#) [spaprvio](#)
 } [addr](#)
- int [mastertype](#)
- union {
  - [virDomainDeviceUSBMaster](#) [usb](#)
 } [master](#)
- int [rombar](#)
- char \* [romfile](#)
- int [bootIndex](#)

### 3.21.1 Field Documentation

3.21.1.1 union { ... } [addr](#)

3.21.1.2 char\* [alias](#)

3.21.1.3 int [bootIndex](#)

3.21.1.4 [virDomainDeviceCcidAddress](#) [ccid](#)

3.21.1.5 [virDomainDeviceDriveAddress](#) [drive](#)

3.21.1.6 union { ... } [master](#)

3.21.1.7 int [mastertype](#)

3.21.1.8 [virDevicePCIAddress](#) [pci](#)

3.21.1.9 int [rombar](#)

3.21.1.10 `char*` `romfile`

3.21.1.11 `virDomainDeviceSpaprVioAddress` `spaprvio`

3.21.1.12 `int` `type`

3.21.1.13 `virDomainDeviceUSBAddress` `usb`

3.21.1.14 `virDomainDeviceUSBMaster` `usb`

3.21.1.15 `virDomainDeviceVirtioSerialAddress` `vioserial`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

## 3.22 `_virDomainDeviceSpaprVioAddress` Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- unsigned long long `reg`
- bool `has_reg`

### 3.22.1 Field Documentation

3.22.1.1 `bool` `has_reg`

3.22.1.2 `unsigned long long` `reg`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

## 3.23 `_virDomainDeviceUSBAddress` Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- unsigned int `bus`
- `char *` `port`

### 3.23.1 Field Documentation

3.23.1.1 `unsigned int` `bus`

3.23.1.2 `char*` `port`

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.24 \_virDomainDeviceUSBMaster Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- unsigned int [startport](#)

### 3.24.1 Field Documentation

#### 3.24.1.1 unsigned int startport

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.25 \_virDomainDeviceVirtioSerialAddress Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- unsigned int [controller](#)
- unsigned int [bus](#)
- unsigned int [port](#)

### 3.25.1 Field Documentation

#### 3.25.1.1 unsigned int bus

#### 3.25.1.2 unsigned int controller

#### 3.25.1.3 unsigned int port

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.26 \_virDomainDiskDef Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- int [type](#)
- int [device](#)
- int [bus](#)
- char \* [src](#)
- char \* [dst](#)
- int [tray\\_status](#)
- int [protocol](#)
- int [nhosts](#)
- [virDomainDiskHostDefPtr](#) [hosts](#)
- struct {
  - char \* [username](#)
  - int [secretType](#)
  - union {
    - unsigned char [uuid](#) [[VIR\\_UUID\\_BUFLLEN](#)]
    - char \* [usage](#)
  - } [secret](#)
- } [auth](#)
- char \* [driverName](#)
- char \* [driverType](#)
- char \* [mirror](#)
- char \* [mirrorFormat](#)
- bool [mirroring](#)
- struct {
  - unsigned int [cylinders](#)
  - unsigned int [heads](#)
  - unsigned int [sectors](#)
  - int [trans](#)
- } [geometry](#)
- struct {
  - unsigned int [logical\\_block\\_size](#)
  - unsigned int [physical\\_block\\_size](#)
- } [blockio](#)
- [virDomainBlockIoTuneInfo](#) [blkdeviotune](#)
- char \* [serial](#)
- char \* [wwn](#)
- int [cachemode](#)
- int [error\\_policy](#)
- int [rerror\\_policy](#)
- int [iomode](#)
- int [ioeventfd](#)
- int [event\\_idx](#)
- int [copy\\_on\\_read](#)
- int [snapshot](#)
- int [startupPolicy](#)
- unsigned int [readonly](#): 1
- unsigned int [shared](#): 1
- unsigned int [transient](#): 1
- [virDomainDeviceInfo](#) [info](#)
- [virStorageEncryptionPtr](#) [encryption](#)
- bool [rawio\\_specified](#)
- int [rawio](#)
- size\_t [nseclabels](#)
- [virSecurityDeviceLabelDefPtr](#) \* [seclabels](#)

### 3.26.1 Field Documentation

3.26.1.1 struct { ... } auth

3.26.1.2 virDomainBlockIoTuneInfo blkdeviotune

3.26.1.3 struct { ... } blockio

3.26.1.4 int bus

3.26.1.5 int cachemode

3.26.1.6 int copy\_on\_read

3.26.1.7 unsigned int cylinders

3.26.1.8 int device

3.26.1.9 char\* driverName

3.26.1.10 char\* driverType

3.26.1.11 char\* dst

3.26.1.12 virStorageEncryptionPtr encryption

3.26.1.13 int error\_policy

3.26.1.14 int event\_idx

3.26.1.15 struct { ... } geometry

3.26.1.16 unsigned int heads

3.26.1.17 virDomainDiskHostDefPtr hosts

3.26.1.18 virDomainDeviceInfo info

3.26.1.19 int ioeventfd

3.26.1.20 int iomode

3.26.1.21 unsigned int logical\_block\_size

3.26.1.22 char\* mirror

3.26.1.23 char\* mirrorFormat

3.26.1.24 bool mirroring

3.26.1.25 int nhosts

3.26.1.26 size\_t nseclabels

3.26.1.27 unsigned int physical\_block\_size

- 3.26.1.28 int protocol
- 3.26.1.29 int rawio
- 3.26.1.30 bool rawio\_specified
- 3.26.1.31 unsigned int readonly
- 3.26.1.32 int error\_policy
- 3.26.1.33 virSecurityDeviceLabelDefPtr\* seclabels
- 3.26.1.34 union { ... } secret
- 3.26.1.35 int secretType
- 3.26.1.36 unsigned int sectors
- 3.26.1.37 char\* serial
- 3.26.1.38 unsigned int shared
- 3.26.1.39 int snapshot
- 3.26.1.40 char\* src
- 3.26.1.41 int startupPolicy
- 3.26.1.42 int trans
- 3.26.1.43 unsigned int transient
- 3.26.1.44 int tray\_status
- 3.26.1.45 int type
- 3.26.1.46 char\* usage
- 3.26.1.47 char\* username
- 3.26.1.48 unsigned char uuid[VIR\_UUID\_BUFLen]
- 3.26.1.49 char\* wwn

The documentation for this struct was generated from the following file:

- src/conf/[domain\\_conf.h](#)

## 3.27 \_virDomainDiskError Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- char \* [disk](#)

- int [error](#)

### 3.27.1 Field Documentation

#### 3.27.1.1 char\* disk

#### 3.27.1.2 int error

The documentation for this struct was generated from the following file:

- include/libvirt/[libvirt.h](#)

## 3.28 \_virDomainDiskHostDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- char \* [name](#)
- char \* [port](#)

### 3.28.1 Field Documentation

#### 3.28.1.1 char\* name

#### 3.28.1.2 char\* port

The documentation for this struct was generated from the following file:

- src/conf/[domain\\_conf.h](#)

## 3.29 \_virDomainEventGraphicsAddress Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- int [family](#)
- const char \* [node](#)
- const char \* [service](#)

### 3.29.1 Detailed Description

virDomainEventGraphicsAddress:

The data structure containing connection address details

### 3.29.2 Field Documentation

#### 3.29.2.1 int family

#### 3.29.2.2 const char\* node

#### 3.29.2.3 const char\* service

The documentation for this struct was generated from the following file:

- include/libvirt/libvirt.h

## 3.30 \_virDomainEventGraphicsSubject Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- int [nidentity](#)
- [virDomainEventGraphicsSubjectIdentityPtr identities](#)

### 3.30.1 Detailed Description

virDomainEventGraphicsSubject:

The data structure representing an authenticated subject

A subject will have zero or more identities. The types of identity differ according to the authentication scheme

### 3.30.2 Field Documentation

#### 3.30.2.1 virDomainEventGraphicsSubjectIdentityPtr identities

#### 3.30.2.2 int nidentity

The documentation for this struct was generated from the following file:

- include/libvirt/libvirt.h

## 3.31 \_virDomainEventGraphicsSubjectIdentity Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- const char \* [type](#)
- const char \* [name](#)



### 3.31.1 Detailed Description

virDomainEventGraphicsSubjectIdentity:

The data structure representing a single identity

The types of identity differ according to the authentication scheme, some examples are 'x509dname' and 'sasl-Username'.

### 3.31.2 Field Documentation

3.31.2.1 `const char* name`

3.31.2.2 `const char* type`

The documentation for this struct was generated from the following file:

- `include/libvirt/libvirt.h`

## 3.32 \_virDomainFSDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- `int type`
- `int fsdriver`
- `int accessmode`
- `int wrpolicy`
- `unsigned long long usage`
- `char * src`
- `char * dst`
- `unsigned int readonly: 1`
- `virDomainDeviceInfo info`
- `unsigned long long space_hard_limit`
- `unsigned long long space_soft_limit`

### 3.32.1 Field Documentation

3.32.1.1 `int accessmode`

3.32.1.2 `char* dst`

3.32.1.3 `int fsdriver`

3.32.1.4 `virDomainDeviceInfo info`

3.32.1.5 `unsigned int readonly`

3.32.1.6 `unsigned long long space_hard_limit`

3.32.1.7 `unsigned long long space_soft_limit`

3.32.1.8 char\* src

3.32.1.9 int type

3.32.1.10 unsigned long long usage

3.32.1.11 int wrpolicy

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

### 3.33 \_virDomainGraphicsAuthDef Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- char \* [passwd](#)
- unsigned int [expires](#): 1
- time\_t [validTo](#)
- int [connected](#)

#### 3.33.1 Field Documentation

3.33.1.1 int connected

3.33.1.2 unsigned int expires

3.33.1.3 char\* passwd

3.33.1.4 time\_t validTo

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

### 3.34 \_virDomainGraphicsDef Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- int [type](#)
- union {
  - struct {
    - int [port](#)
    - unsigned int [autoport](#):1
    - char \* [keymap](#)
    - char \* [socket](#)
    - [virDomainGraphicsAuthDef](#) auth

```

    } vnc
    struct {
        char * display
        char * xauth
        int fullscreen
    } sdl
    struct {
        int port
        unsigned int autoport:1
        unsigned int replaceUser:1
        unsigned int multiUser:1
    } rdp
    struct {
        char * display
        unsigned int fullscreen:1
    } desktop
    struct {
        int port
        int tlsPort
        int mousemode
        char * keymap
        virDomainGraphicsAuthDef auth
        unsigned int autoport:1
        int channels [VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_LAST]
        int defaultMode
        int image
        int jpeg
        int zlib
        int playback
        int streaming
        int copypaste
    } spice
} data

```

- size\_t nListens
- virDomainGraphicsListenDefPtr listens

### 3.34.1 Field Documentation

#### 3.34.1.1 virDomainGraphicsAuthDef auth

#### 3.34.1.2 unsigned int autoport

#### 3.34.1.3 int channels[VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_LAST]

#### 3.34.1.4 int copypaste

#### 3.34.1.5 union { ... } data

#### 3.34.1.6 int defaultMode

#### 3.34.1.7 struct { ... } desktop

#### 3.34.1.8 char\* display

#### 3.34.1.9 int fullscreen

- 3.34.1.10 unsigned int fullscreen
- 3.34.1.11 int image
- 3.34.1.12 int jpeg
- 3.34.1.13 char\* keymap
- 3.34.1.14 virDomainGraphicsListenDefPtr listens
- 3.34.1.15 int mousemode
- 3.34.1.16 unsigned int multiUser
- 3.34.1.17 size\_t nListens
- 3.34.1.18 int playback
- 3.34.1.19 int port
- 3.34.1.20 struct { ... } rdp
- 3.34.1.21 unsigned int replaceUser
- 3.34.1.22 struct { ... } sdl
- 3.34.1.23 char\* socket
- 3.34.1.24 struct { ... } spice
- 3.34.1.25 int streaming
- 3.34.1.26 int tlsPort
- 3.34.1.27 int type
- 3.34.1.28 struct { ... } vnc
- 3.34.1.29 char\* xauth
- 3.34.1.30 int zlib

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.35 \_virDomainGraphicsListenDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [type](#)
- char \* [address](#)
- char \* [network](#)

### 3.35.1 Field Documentation

3.35.1.1 char\* address

3.35.1.2 char\* network

3.35.1.3 int type

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.36 \_virDomainHostdevDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- [virDomainDeviceDef](#) parent
- int [mode](#)
- unsigned int [managed](#): 1
- union {
  - [virDomainHostdevSubsys](#) subsys
  - struct {
    - int [dummy](#)
  - [caps](#)
  - [source](#)
- [virDomainHostdevOrigStates](#) origstates
- [virDomainDeviceInfoPtr](#) info

### 3.36.1 Field Documentation

3.36.1.1 struct { ... } caps

3.36.1.2 int dummy

3.36.1.3 [virDomainDeviceInfoPtr](#) info

3.36.1.4 unsigned int managed

3.36.1.5 int mode

3.36.1.6 [virDomainHostdevOrigStates](#) origstates

3.36.1.7 [virDomainDeviceDef](#) parent

3.36.1.8 union { ... } source

3.36.1.9 [virDomainHostdevSubsys](#) subsys

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

### 3.37 `_virDomainHostdevOrigStates` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- union {
  - struct {
    - unsigned int `unbind_from_stub`: 1
    - unsigned int `remove_slot`: 1
    - unsigned int `reprobe`:1
  - } `pci`
- } `states`

#### 3.37.1 Field Documentation

3.37.1.1 struct { ... } `pci`

3.37.1.2 unsigned int `remove_slot`

3.37.1.3 unsigned int `reprobe`

3.37.1.4 union { ... } `states`

3.37.1.5 unsigned int `unbind_from_stub`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

### 3.38 `_virDomainHostdevSubsys` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- int `type`
- union {
  - struct {
    - unsigned `bus`
    - unsigned `device`
    - unsigned `vendor`
    - unsigned `product`
  - } `usb`
  - virDevicePCIAddress `pci`
- } `u`

#### 3.38.1 Field Documentation

3.38.1.1 unsigned `bus`

3.38.1.2 unsigned device

3.38.1.3 virDevicePCIAddress pci

3.38.1.4 unsigned product

3.38.1.5 int type

3.38.1.6 union { ... } u

3.38.1.7 struct { ... } usb

3.38.1.8 unsigned vendor

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.39 \_virDomainHubDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [type](#)
- [virDomainDeviceInfo](#) info

### 3.39.1 Field Documentation

3.39.1.1 [virDomainDeviceInfo](#) info

3.39.1.2 int type

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.40 \_virDomainInfo Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- unsigned char [state](#)
- unsigned long [maxMem](#)
- unsigned long [memory](#)
- unsigned short [nrVirtCpu](#)
- unsigned long long [cpuTime](#)

### 3.40.1 Field Documentation

3.40.1.1 unsigned long long cpuTime

3.40.1.2 unsigned long maxMem

3.40.1.3 unsigned long memory

3.40.1.4 unsigned short nrVirtCpu

3.40.1.5 unsigned char state

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.41 \_virDomainInputDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [type](#)
- int [bus](#)
- [virDomainDeviceInfo](#) info

### 3.41.1 Field Documentation

3.41.1.1 int bus

3.41.1.2 [virDomainDeviceInfo](#) info

3.41.1.3 int type

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.42 \_virDomainInterfaceStats Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- long long [rx\\_bytes](#)
- long long [rx\\_packets](#)
- long long [rx\\_errs](#)
- long long [rx\\_drop](#)
- long long [tx\\_bytes](#)
- long long [tx\\_packets](#)
- long long [tx\\_errs](#)
- long long [tx\\_drop](#)



### 3.42.1 Field Documentation

3.42.1.1 long long rx\_bytes

3.42.1.2 long long rx\_drop

3.42.1.3 long long rx\_errs

3.42.1.4 long long rx\_packets

3.42.1.5 long long tx\_bytes

3.42.1.6 long long tx\_drop

3.42.1.7 long long tx\_errs

3.42.1.8 long long tx\_packets

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.43 \_virDomainJobInfo Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- int [type](#)
- unsigned long long [timeElapsed](#)
- unsigned long long [timeRemaining](#)
- unsigned long long [dataTotal](#)
- unsigned long long [dataProcessed](#)
- unsigned long long [dataRemaining](#)
- unsigned long long [memTotal](#)
- unsigned long long [memProcessed](#)
- unsigned long long [memRemaining](#)
- unsigned long long [fileTotal](#)
- unsigned long long [fileProcessed](#)
- unsigned long long [fileRemaining](#)

### 3.43.1 Field Documentation

3.43.1.1 unsigned long long dataProcessed

3.43.1.2 unsigned long long dataRemaining

3.43.1.3 unsigned long long dataTotal

3.43.1.4 unsigned long long fileProcessed

3.43.1.5 unsigned long long fileRemaining

3.43.1.6 unsigned long long fileTotal

3.43.1.7 unsigned long long memProcessed

3.43.1.8 unsigned long long memRemaining

3.43.1.9 unsigned long long memTotal

3.43.1.10 unsigned long long timeElapsed

3.43.1.11 unsigned long long timeRemaining

3.43.1.12 int type

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.44 `_virDomainLeaseDef` Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- char \* [lockspace](#)
- char \* [key](#)
- char \* [path](#)
- unsigned long long [offset](#)

### 3.44.1 Field Documentation

3.44.1.1 char\* key

3.44.1.2 char\* lockspace

3.44.1.3 unsigned long long offset

3.44.1.4 char\* path

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.45 `_virDomainMemballoonDef` Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [model](#)
- [virDomainDeviceInfo](#) info

### 3.45.1 Field Documentation

#### 3.45.1.1 virDomainDeviceInfo info

#### 3.45.1.2 int model

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.46 \_virDomainMemoryStat Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- int [tag](#)
- unsigned long long [val](#)

### 3.46.1 Field Documentation

#### 3.46.1.1 int tag

#### 3.46.1.2 unsigned long long val

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.47 \_virDomainNetDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- enum [virDomainNetType](#) type
- virMacAddr [mac](#)
- char \* [model](#)
- union {
  - struct {
    - enum [virDomainNetBackendType](#) name
    - enum [virDomainNetVirtioTxModeType](#) txmode
    - enum [virDomainIoEventFd](#) ioeventfd
    - enum [virDomainVirtioEventIdx](#) event\_idx
  - [virtio](#)
  - [driver](#)
- union {
  - struct {
    - char \* [dev](#)
    - char \* [ipaddr](#)

```

    } ethernet
    struct {
        char * address
        int port
    } socket
    struct {
        char * name
        char * portgroup
        virDomainActualNetDefPtr actual
    } network
    struct {
        char * brname
        char * brtype POL New
        char * ipaddr
    } bridge
    struct {
        char * name
    } internal
    struct {
        char * linkdev
        int mode
    } direct
    struct {
        virDomainHostdevDef def
    } hostdev
} data

```

- virNetDevVPortProfilePtr virtPortProfile
- struct {
 bool sndbuf\_specified
 unsigned long sndbuf
 } tune
- char \* script
- char \* ifname
- virDomainDeviceInfo info
- char \* filter
- virNWFilterHashTablePtr filterparams
- virNetDevBandwidthPtr bandwidth
- virNetDevVlan vlan
- int linkstate

### 3.47.1 Field Documentation

#### 3.47.1.1 virDomainActualNetDefPtr actual

#### 3.47.1.2 char\* address

#### 3.47.1.3 virNetDevBandwidthPtr bandwidth

#### 3.47.1.4 struct { ... } bridge

#### 3.47.1.5 char\* brname

#### 3.47.1.6 char\* brtype

- 3.47.1.7 union { ... } data
- 3.47.1.8 virDomainHostdevDef def
- 3.47.1.9 char\* dev
- 3.47.1.10 struct { ... } direct
- 3.47.1.11 union { ... } driver
- 3.47.1.12 struct { ... } ethernet
- 3.47.1.13 enum virDomainVirtioEventIdx event\_idx
- 3.47.1.14 char\* filter
- 3.47.1.15 virNWFilterHashTablePtr filterparams
- 3.47.1.16 struct { ... } hostdev
- 3.47.1.17 char\* ifname
- 3.47.1.18 virDomainDeviceInfo info
- 3.47.1.19 struct { ... } internal
- 3.47.1.20 enum virDomainIoEventFd ioeventfd
- 3.47.1.21 char\* ipaddr
- 3.47.1.22 char\* linkdev
- 3.47.1.23 int linkstate
- 3.47.1.24 virMacAddr mac
- 3.47.1.25 int mode
- 3.47.1.26 char\* model
- 3.47.1.27 enum virDomainNetBackendType name
- 3.47.1.28 char\* name
- 3.47.1.29 struct { ... } network
- 3.47.1.30 int port
- 3.47.1.31 char\* portgroup
- 3.47.1.32 char\* script
- 3.47.1.33 unsigned long sndbuf
- 3.47.1.34 bool sndbuf\_specified

3.47.1.35 struct { ... } socket

3.47.1.36 struct { ... } tune

3.47.1.37 enum virDomainNetVirtioTxModeType txmode

3.47.1.38 enum virDomainNetType type

3.47.1.39 struct { ... } virtio

3.47.1.40 virNetDevVPortProfilePtr virtPortProfile

3.47.1.41 virNetDevVlan vlan

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.48 \_virDomainNumatuneDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- struct {  
    virBitmapPtr [nodemask](#)  
    int [mode](#)  
    int [placement\\_mode](#)  
} [memory](#)

### 3.48.1 Field Documentation

3.48.1.1 struct { ... } memory

3.48.1.2 int mode

3.48.1.3 virBitmapPtr nodemask

3.48.1.4 int placement\_mode

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.49 \_virDomainObj Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- virObject [object](#)

- virMutex [lock](#)
- pid\_t [pid](#)
- virDomainStateReason [state](#)
- unsigned int [autostart](#): 1
- unsigned int [persistent](#): 1
- unsigned int [updated](#): 1
- virDomainDefPtr [def](#)
- virDomainDefPtr [newDef](#)
- virDomainSnapshotObjListPtr [snapshots](#)
- virDomainSnapshotObjPtr [current\\_snapshot](#)
- bool [hasManagedSave](#)
- void \* [privateData](#)
- void(\* [privateDataFreeFunc](#))(void \*)
- int [taint](#)

### 3.49.1 Field Documentation

3.49.1.1 unsigned int autostart

3.49.1.2 virDomainSnapshotObjPtr current\_snapshot

3.49.1.3 virDomainDefPtr def

3.49.1.4 bool hasManagedSave

3.49.1.5 virMutex lock

3.49.1.6 virDomainDefPtr newDef

3.49.1.7 virObject object

3.49.1.8 unsigned int persistent

3.49.1.9 pid\_t pid

3.49.1.10 void\* privateData

3.49.1.11 void(\* privateDataFreeFunc)(void \*)

3.49.1.12 virDomainSnapshotObjListPtr snapshots

3.49.1.13 virDomainStateReason state

3.49.1.14 int taint

3.49.1.15 unsigned int updated

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.50 \_virDomainObjList Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- virHashTable \* [objs](#)

### 3.50.1 Field Documentation

#### 3.50.1.1 virHashTable\* objs

The documentation for this struct was generated from the following file:

- src/conf/[domain\\_conf.h](#)

## 3.51 \_virDomainOSDef Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- char \* [type](#)
- char \* [arch](#)
- char \* [machine](#)
- int [nBootDevs](#)
- int [bootDevs](#) [[VIR\\_DOMAIN\\_BOOT\\_LAST](#)]
- int [bootmenu](#)
- char \* [init](#)
- char \*\* [initargv](#)
- char \* [kernel](#)
- char \* [initrd](#)
- char \* [cmdline](#)
- char \* [root](#)
- char \* [loader](#)
- char \* [bootloader](#)
- char \* [bootloaderArgs](#)
- int [smbios\\_mode](#)
- [virDomainBIOSDef](#) [bios](#)

### 3.51.1 Field Documentation

#### 3.51.1.1 char\* arch

#### 3.51.1.2 virDomainBIOSDef bios

#### 3.51.1.3 int bootDevs[VIR\_DOMAIN\_BOOT\_LAST]

#### 3.51.1.4 char\* bootloader

#### 3.51.1.5 char\* bootloaderArgs

#### 3.51.1.6 int bootmenu

#### 3.51.1.7 char\* cmdline



3.51.1.8 char\* init

3.51.1.9 char\*\* initargv

3.51.1.10 char\* initrd

3.51.1.11 char\* kernel

3.51.1.12 char\* loader

3.51.1.13 char\* machine

3.51.1.14 int nBootDevs

3.51.1.15 char\* root

3.51.1.16 int smbios\_mode

3.51.1.17 char\* type

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.52 \_virDomainRedirdevDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- int [bus](#)
- union {  
    [virDomainChrSourceDef](#) chr  
} [source](#)
- [virDomainDeviceInfo](#) info

### 3.52.1 Field Documentation

3.52.1.1 int bus

3.52.1.2 [virDomainChrSourceDef](#) chr

3.52.1.3 [virDomainDeviceInfo](#) info

3.52.1.4 union { ... } source

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

### 3.53 `_virDomainRedirFilterDef` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- `size_t nusbdevs`
- `virDomainRedirFilterUsbDevDefPtr * usbdevs`

#### 3.53.1 Field Documentation

3.53.1.1 `size_t nusbdevs`

3.53.1.2 `virDomainRedirFilterUsbDevDefPtr* usbdevs`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

### 3.54 `_virDomainRedirFilterUsbDevDef` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- `int usbClass`
- `int vendor`
- `int product`
- `int version`
- `unsigned int allow:1`

#### 3.54.1 Field Documentation

3.54.1.1 `unsigned int allow`

3.54.1.2 `int product`

3.54.1.3 `int usbClass`

3.54.1.4 `int vendor`

3.54.1.5 `int version`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

### 3.55 `_virDomainSmartcardDef` Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- int [type](#)
- union {
  - struct {
    - char \* [file](#) [[VIR\\_DOMAIN\\_SMARTCARD\\_NUM\\_CERTIFICATES](#)]
    - char \* [database](#)
  - [cert](#)
  - [virDomainChrSourceDef](#) [passthru](#)
- } [data](#)
- [virDomainDeviceInfo](#) [info](#)

### 3.55.1 Field Documentation

3.55.1.1 struct { ... } [cert](#)

3.55.1.2 union { ... } [data](#)

3.55.1.3 char\* [database](#)

3.55.1.4 char\* [file](#)[[VIR\\_DOMAIN\\_SMARTCARD\\_NUM\\_CERTIFICATES](#)]

3.55.1.5 [virDomainDeviceInfo](#) [info](#)

3.55.1.6 [virDomainChrSourceDef](#) [passthru](#)

3.55.1.7 int [type](#)

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.56 \_virDomainSoundCodecDef Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- int [type](#)
- int [cad](#)

### 3.56.1 Field Documentation

3.56.1.1 int [cad](#)

3.56.1.2 int [type](#)

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

### 3.57 `_virDomainSoundDef` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- int [model](#)
- [virDomainDeviceInfo](#) info
- int [ncodecs](#)
- [virDomainSoundCodecDefPtr](#) \* [codecs](#)

#### 3.57.1 Field Documentation

3.57.1.1 `virDomainSoundCodecDefPtr`\* [codecs](#)

3.57.1.2 `virDomainDeviceInfo` [info](#)

3.57.1.3 int [model](#)

3.57.1.4 int [ncodecs](#)

The documentation for this struct was generated from the following file:

- `src/conf/domain\_conf.h`

### 3.58 `_virDomainStateReason` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- int [state](#)
- int [reason](#)

#### 3.58.1 Field Documentation

3.58.1.1 int [reason](#)

3.58.1.2 int [state](#)

The documentation for this struct was generated from the following file:

- `src/conf/domain\_conf.h`

### 3.59 `_virDomainTimerCatchupDef` Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- unsigned long [threshold](#)
- unsigned long [slew](#)
- unsigned long [limit](#)

### 3.59.1 Field Documentation

3.59.1.1 unsigned long limit

3.59.1.2 unsigned long slew

3.59.1.3 unsigned long threshold

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.60 \_virDomainTimerDef Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- int [name](#)
- int [present](#)
- int [tickpolicy](#)
- [virDomainTimerCatchupDef](#) catchup
- int [track](#)
- unsigned long [frequency](#)
- int [mode](#)

### 3.60.1 Field Documentation

3.60.1.1 [virDomainTimerCatchupDef](#) catchup

3.60.1.2 unsigned long frequency

3.60.1.3 int mode

3.60.1.4 int name

3.60.1.5 int present

3.60.1.6 int tickpolicy

3.60.1.7 int track

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

### 3.61 `_virDomainVcpuPinDef` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- int `vcpuid`
- virBitmapPtr `cpumask`

#### 3.61.1 Field Documentation

3.61.1.1 `virBitmapPtr cpumask`

3.61.1.2 `int vcpuid`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

### 3.62 `_virDomainVideoAccelDef` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- unsigned int `support3d`:1
- unsigned int `support2d`:1

#### 3.62.1 Field Documentation

3.62.1.1 `unsigned int support2d`

3.62.1.2 `unsigned int support3d`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

### 3.63 `_virDomainVideoDef` Struct Reference

```
#include <domain_conf.h>
```

#### Data Fields

- int `type`
- unsigned int `vram`
- unsigned int `heads`
- `virDomainVideoAccelDefPtr accel`
- `virDomainDeviceInfo info`

### 3.63.1 Field Documentation

3.63.1.1 `virDomainVideoAccelDefPtr accel`

3.63.1.2 `unsigned int heads`

3.63.1.3 `virDomainDeviceInfo info`

3.63.1.4 `int type`

3.63.1.5 `unsigned int vram`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

## 3.64 \_virDomainVirtioSerialOpts Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- `int ports`
- `int vectors`

### 3.64.1 Field Documentation

3.64.1.1 `int ports`

3.64.1.2 `int vectors`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.h`

## 3.65 \_virDomainWatchdogDef Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- `int model`
- `int action`
- `virDomainDeviceInfo info`

### 3.65.1 Field Documentation

3.65.1.1 `int action`

3.65.1.2 `virDomainDeviceInfo info`

### 3.65.1.3 int model

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.66 \_virDriver Struct Reference

```
#include <driver.h>
```

### Data Fields

- int [no](#)
- const char \* [name](#)
- [virDrvOpen](#) open
- [virDrvClose](#) close
- [virDrvDrvSupportsFeature](#) supports\_feature
- [virDrvGetType](#) type
- [virDrvGetVersion](#) version
- [virDrvGetLibVersion](#) libvirtVersion
- [virDrvGetHostname](#) getHostname
- [virDrvGetSysinfo](#) getSysinfo
- [virDrvGetMaxVcpus](#) getMaxVcpus
- [virDrvNodeGetInfo](#) nodeGetInfo
- [virDrvGetCapabilities](#) getCapabilities
- [virDrvListDomains](#) listDomains
- [virDrvNumOfDomains](#) numOfDomains
- [virDrvListAllDomains](#) listAllDomains
- [virDrvDomainCreateXML](#) domainCreateXML
- [virDrvDomainLookupByID](#) domainLookupByID
- [virDrvDomainLookupByUUID](#) domainLookupByUUID
- [virDrvDomainLookupByName](#) domainLookupByName
- [virDrvDomainSuspend](#) domainSuspend
- [virDrvDomainResume](#) domainResume
- [virDrvDomainPMSuspendForDuration](#) domainPMSuspendForDuration
- [virDrvDomainPMWakeup](#) domainPMWakeup
- [virDrvDomainShutdown](#) domainShutdown
- [virDrvDomainShutdownFlags](#) domainShutdownFlags
- [virDrvDomainReboot](#) domainReboot
- [virDrvDomainReset](#) domainReset
- [virDrvDomainDestroy](#) domainDestroy
- [virDrvDomainDestroyFlags](#) domainDestroyFlags
- [virDrvDomainGetOSType](#) domainGetOSType
- [virDrvDomainGetHostname](#) domainGetHostname
- [virDrvDomainGetMaxMemory](#) domainGetMaxMemory
- [virDrvDomainSetMaxMemory](#) domainSetMaxMemory
- [virDrvDomainSetMemory](#) domainSetMemory
- [virDrvDomainSetMemoryFlags](#) domainSetMemoryFlags
- [virDrvDomainSetMemoryParameters](#) domainSetMemoryParameters
- [virDrvDomainGetMemoryParameters](#) domainGetMemoryParameters
- [virDrvDomainSetNumaParameters](#) domainSetNumaParameters
- [virDrvDomainGetNumaParameters](#) domainGetNumaParameters
- [virDrvDomainSetBlkioParameters](#) domainSetBlkioParameters



- [virDrvDomainGetBlkioParameters](#) [domainGetBlkioParameters](#)
- [virDrvDomainGetInfo](#) [domainGetInfo](#)
- [virDrvDomainGetState](#) [domainGetState](#)
- [virDrvDomainGetControllInfo](#) [domainGetControllInfo](#)
- [virDrvDomainSave](#) [domainSave](#)
- [virDrvDomainSaveFlags](#) [domainSaveFlags](#)
- [virDrvDomainRestore](#) [domainRestore](#)
- [virDrvDomainRestoreFlags](#) [domainRestoreFlags](#)
- [virDrvDomainSavImageGetXMLDesc](#) [domainSavImageGetXMLDesc](#)
- [virDrvDomainSavImageDefineXML](#) [domainSavImageDefineXML](#)
- [virDrvDomainCoreDump](#) [domainCoreDump](#)
- [virDrvDomainScreenshot](#) [domainScreenshot](#)
- [virDrvDomainSetVcpus](#) [domainSetVcpus](#)
- [virDrvDomainSetVcpusFlags](#) [domainSetVcpusFlags](#)
- [virDrvDomainGetVcpusFlags](#) [domainGetVcpusFlags](#)
- [virDrvDomainPinVcpu](#) [domainPinVcpu](#)
- [virDrvDomainPinVcpuFlags](#) [domainPinVcpuFlags](#)
- [virDrvDomainGetVcpuPinInfo](#) [domainGetVcpuPinInfo](#)
- [virDrvDomainPinEmulator](#) [domainPinEmulator](#)
- [virDrvDomainGetEmulatorPinInfo](#) [domainGetEmulatorPinInfo](#)
- [virDrvDomainGetVcpus](#) [domainGetVcpus](#)
- [virDrvDomainGetMaxVcpus](#) [domainGetMaxVcpus](#)
- [virDrvDomainGetSecurityLabel](#) [domainGetSecurityLabel](#)
- [virDrvDomainGetSecurityLabelList](#) [domainGetSecurityLabelList](#)
- [virDrvNodeGetSecurityModel](#) [nodeGetSecurityModel](#)
- [virDrvDomainGetXMLDesc](#) [domainGetXMLDesc](#)
- [virDrvConnectDomainXMLFromNative](#) [domainXMLFromNative](#)
- [virDrvConnectDomainXMLToNative](#) [domainXMLToNative](#)
- [virDrvListDefinedDomains](#) [listDefinedDomains](#)
- [virDrvNumOfDefinedDomains](#) [numOfDefinedDomains](#)
- [virDrvDomainCreate](#) [domainCreate](#)
- [virDrvDomainCreateWithFlags](#) [domainCreateWithFlags](#)
- [virDrvDomainDefineXML](#) [domainDefineXML](#)
- [virDrvDomainUndefine](#) [domainUndefine](#)
- [virDrvDomainUndefineFlags](#) [domainUndefineFlags](#)
- [virDrvDomainAttachDevice](#) [domainAttachDevice](#)
- [virDrvDomainAttachDeviceFlags](#) [domainAttachDeviceFlags](#)
- [virDrvDomainDetachDevice](#) [domainDetachDevice](#)
- [virDrvDomainDetachDeviceFlags](#) [domainDetachDeviceFlags](#)
- [virDrvDomainUpdateDeviceFlags](#) [domainUpdateDeviceFlags](#)
- [virDrvDomainGetAutostart](#) [domainGetAutostart](#)
- [virDrvDomainSetAutostart](#) [domainSetAutostart](#)
- [virDrvDomainGetSchedulerType](#) [domainGetSchedulerType](#)
- [virDrvDomainGetSchedulerParameters](#) [domainGetSchedulerParameters](#)
- [virDrvDomainGetSchedulerParametersFlags](#) [domainGetSchedulerParametersFlags](#)
- [virDrvDomainSetSchedulerParameters](#) [domainSetSchedulerParameters](#)
- [virDrvDomainSetSchedulerParametersFlags](#) [domainSetSchedulerParametersFlags](#)
- [virDrvDomainMigratePrepare](#) [domainMigratePrepare](#)
- [virDrvDomainMigratePerform](#) [domainMigratePerform](#)
- [virDrvDomainMigrateFinish](#) [domainMigrateFinish](#)
- [virDrvDomainBlockResize](#) [domainBlockResize](#)
- [virDrvDomainBlockStats](#) [domainBlockStats](#)
- [virDrvDomainBlockStatsFlags](#) [domainBlockStatsFlags](#)
- [virDrvDomainInterfaceStats](#) [domainInterfaceStats](#)
- [virDrvDomainSetInterfaceParameters](#) [domainSetInterfaceParameters](#)

- [virDrvDomainGetInterfaceParameters](#) `domainGetInterfaceParameters`
- [virDrvDomainMemoryStats](#) `domainMemoryStats`
- [virDrvDomainBlockPeek](#) `domainBlockPeek`
- [virDrvDomainMemoryPeek](#) `domainMemoryPeek`
- [virDrvDomainGetBlockInfo](#) `domainGetBlockInfo`
- [virDrvNodeGetCPUStats](#) `nodeGetCPUStats`
- [virDrvNodeGetMemoryStats](#) `nodeGetMemoryStats`
- [virDrvNodeGetCellsFreeMemory](#) `nodeGetCellsFreeMemory`
- [virDrvNodeGetFreeMemory](#) `nodeGetFreeMemory`
- [virDrvDomainEventRegister](#) `domainEventRegister`
- [virDrvDomainEventDeregister](#) `domainEventDeregister`
- [virDrvDomainMigratePrepare2](#) `domainMigratePrepare2`
- [virDrvDomainMigrateFinish2](#) `domainMigrateFinish2`
- [virDrvNodeDeviceDettach](#) `nodeDeviceDettach`
- [virDrvNodeDeviceReAttach](#) `nodeDeviceReAttach`
- [virDrvNodeDeviceReset](#) `nodeDeviceReset`
- [virDrvDomainMigratePrepareTunnel](#) `domainMigratePrepareTunnel`
- [virDrvConnectIsEncrypted](#) `isEncrypted`
- [virDrvConnectIsSecure](#) `isSecure`
- [virDrvDomainsIsActive](#) `domainsIsActive`
- [virDrvDomainsIsPersistent](#) `domainsIsPersistent`
- [virDrvDomainsIsUpdated](#) `domainsIsUpdated`
- [virDrvCompareCPU](#) `cpuCompare`
- [virDrvBaselineCPU](#) `cpuBaseline`
- [virDrvDomainGetJobInfo](#) `domainGetJobInfo`
- [virDrvDomainAbortJob](#) `domainAbortJob`
- [virDrvDomainMigrateSetMaxDowntime](#) `domainMigrateSetMaxDowntime`
- [virDrvDomainMigrateGetMaxSpeed](#) `domainMigrateGetMaxSpeed`
- [virDrvDomainMigrateSetMaxSpeed](#) `domainMigrateSetMaxSpeed`
- [virDrvDomainEventRegisterAny](#) `domainEventRegisterAny`
- [virDrvDomainEventDeregisterAny](#) `domainEventDeregisterAny`
- [virDrvDomainManagedSave](#) `domainManagedSave`
- [virDrvDomainHasManagedSaveImage](#) `domainHasManagedSaveImage`
- [virDrvDomainManagedSaveRemove](#) `domainManagedSaveRemove`
- [virDrvDomainSnapshotCreateXML](#) `domainSnapshotCreateXML`
- [virDrvDomainSnapshotGetXMLDesc](#) `domainSnapshotGetXMLDesc`
- [virDrvDomainSnapshotNum](#) `domainSnapshotNum`
- [virDrvDomainSnapshotListNames](#) `domainSnapshotListNames`
- [virDrvDomainListAllSnapshots](#) `domainListAllSnapshots`
- [virDrvDomainSnapshotNumChildren](#) `domainSnapshotNumChildren`
- [virDrvDomainSnapshotListChildrenNames](#) `domainSnapshotListChildrenNames`
- [virDrvDomainSnapshotListAllChildren](#) `domainSnapshotListAllChildren`
- [virDrvDomainSnapshotLookupByName](#) `domainSnapshotLookupByName`
- [virDrvDomainHasCurrentSnapshot](#) `domainHasCurrentSnapshot`
- [virDrvDomainSnapshotGetParent](#) `domainSnapshotGetParent`
- [virDrvDomainSnapshotCurrent](#) `domainSnapshotCurrent`
- [virDrvDomainSnapshotIsCurrent](#) `domainSnapshotIsCurrent`
- [virDrvDomainSnapshotHasMetadata](#) `domainSnapshotHasMetadata`
- [virDrvDomainRevertToSnapshot](#) `domainRevertToSnapshot`
- [virDrvDomainSnapshotDelete](#) `domainSnapshotDelete`
- [virDrvDomainQemuMonitorCommand](#) `qemuDomainMonitorCommand`
- [virDrvDomainQemuAttach](#) `qemuDomainAttach`
- [virDrvDomainQemuAgentCommand](#) `qemuDomainArbitraryAgentCommand`
- [virDrvDomainOpenConsole](#) `domainOpenConsole`
- [virDrvDomainOpenGraphics](#) `domainOpenGraphics`

- [virDrvDomainInjectNMI](#) domainInjectNMI
- [virDrvDomainMigrateBegin3](#) domainMigrateBegin3
- [virDrvDomainMigratePrepare3](#) domainMigratePrepare3
- [virDrvDomainMigratePrepareTunnel3](#) domainMigratePrepareTunnel3
- [virDrvDomainMigratePerform3](#) domainMigratePerform3
- [virDrvDomainMigrateFinish3](#) domainMigrateFinish3
- [virDrvDomainMigrateConfirm3](#) domainMigrateConfirm3
- [virDrvDomainSendKey](#) domainSendKey
- [virDrvDomainBlockJobAbort](#) domainBlockJobAbort
- [virDrvDomainGetBlockJobInfo](#) domainGetBlockJobInfo
- [virDrvDomainBlockJobSetSpeed](#) domainBlockJobSetSpeed
- [virDrvDomainBlockPull](#) domainBlockPull
- [virDrvDomainBlockRebase](#) domainBlockRebase
- [virDrvDomainBlockCommit](#) domainBlockCommit
- [virDrvSetKeepAlive](#) setKeepAlive
- [virDrvConnectIsAlive](#) isAlive
- [virDrvNodeSuspendForDuration](#) nodeSuspendForDuration
- [virDrvDomainSetBlockIoTune](#) domainSetBlockIoTune
- [virDrvDomainGetBlockIoTune](#) domainGetBlockIoTune
- [virDrvDomainGetCPUStats](#) domainGetCPUStats
- [virDrvDomainGetDiskErrors](#) domainGetDiskErrors
- [virDrvDomainSetMetadata](#) domainSetMetadata
- [virDrvDomainGetMetadata](#) domainGetMetadata
- [virDrvNodeGetMemoryParameters](#) nodeGetMemoryParameters
- [virDrvNodeSetMemoryParameters](#) nodeSetMemoryParameters

### 3.66.1 Detailed Description

[\\_virDriver](#):

Structure associated to a virtualization driver, defining the various entry points for it.

All drivers must support the following fields/methods:

- no
- name
- open
- close

### 3.66.2 Field Documentation

3.66.2.1 **virDrvClose** close

3.66.2.2 **virDrvBaselineCPU** cpuBaseline

3.66.2.3 **virDrvCompareCPU** cpuCompare

3.66.2.4 **virDrvDomainAbortJob** domainAbortJob

3.66.2.5 **virDrvDomainAttachDevice** domainAttachDevice

3.66.2.6 **virDrvDomainAttachDeviceFlags** domainAttachDeviceFlags

- 3.66.2.7 **virDrvDomainBlockCommit** domainBlockCommit
- 3.66.2.8 **virDrvDomainBlockJobAbort** domainBlockJobAbort
- 3.66.2.9 **virDrvDomainBlockJobSetSpeed** domainBlockJobSetSpeed
- 3.66.2.10 **virDrvDomainBlockPeek** domainBlockPeek
- 3.66.2.11 **virDrvDomainBlockPull** domainBlockPull
- 3.66.2.12 **virDrvDomainBlockRebase** domainBlockRebase
- 3.66.2.13 **virDrvDomainBlockResize** domainBlockResize
- 3.66.2.14 **virDrvDomainBlockStats** domainBlockStats
- 3.66.2.15 **virDrvDomainBlockStatsFlags** domainBlockStatsFlags
- 3.66.2.16 **virDrvDomainCoreDump** domainCoreDump
- 3.66.2.17 **virDrvDomainCreate** domainCreate
- 3.66.2.18 **virDrvDomainCreateWithFlags** domainCreateWithFlags
- 3.66.2.19 **virDrvDomainCreateXML** domainCreateXML
- 3.66.2.20 **virDrvDomainDefineXML** domainDefineXML
- 3.66.2.21 **virDrvDomainDestroy** domainDestroy
- 3.66.2.22 **virDrvDomainDestroyFlags** domainDestroyFlags
- 3.66.2.23 **virDrvDomainDetachDevice** domainDetachDevice
- 3.66.2.24 **virDrvDomainDetachDeviceFlags** domainDetachDeviceFlags
- 3.66.2.25 **virDrvDomainEventDeregister** domainEventDeregister
- 3.66.2.26 **virDrvDomainEventDeregisterAny** domainEventDeregisterAny
- 3.66.2.27 **virDrvDomainEventRegister** domainEventRegister
- 3.66.2.28 **virDrvDomainEventRegisterAny** domainEventRegisterAny
- 3.66.2.29 **virDrvDomainGetAutostart** domainGetAutostart
- 3.66.2.30 **virDrvDomainGetBlkioParameters** domainGetBlkioParameters
- 3.66.2.31 **virDrvDomainGetBlockInfo** domainGetBlockInfo
- 3.66.2.32 **virDrvDomainGetBlockIoTune** domainGetBlockIoTune
- 3.66.2.33 **virDrvDomainGetBlockJobInfo** domainGetBlockJobInfo
- 3.66.2.34 **virDrvDomainGetControllInfo** domainGetControllInfo

- 3.66.2.35 **virDrvDomainGetCPUStats** domainGetCPUStats
- 3.66.2.36 **virDrvDomainGetDiskErrors** domainGetDiskErrors
- 3.66.2.37 **virDrvDomainGetEmulatorPinInfo** domainGetEmulatorPinInfo
- 3.66.2.38 **virDrvDomainGetHostname** domainGetHostname
- 3.66.2.39 **virDrvDomainGetInfo** domainGetInfo
- 3.66.2.40 **virDrvDomainGetInterfaceParameters** domainGetInterfaceParameters
- 3.66.2.41 **virDrvDomainGetJobInfo** domainGetJobInfo
- 3.66.2.42 **virDrvDomainGetMaxMemory** domainGetMaxMemory
- 3.66.2.43 **virDrvDomainGetMaxVcpus** domainGetMaxVcpus
- 3.66.2.44 **virDrvDomainGetMemoryParameters** domainGetMemoryParameters
- 3.66.2.45 **virDrvDomainGetMetadata** domainGetMetadata
- 3.66.2.46 **virDrvDomainGetNumaParameters** domainGetNumaParameters
- 3.66.2.47 **virDrvDomainGetOSType** domainGetOSType
- 3.66.2.48 **virDrvDomainGetSchedulerParameters** domainGetSchedulerParameters
- 3.66.2.49 **virDrvDomainGetSchedulerParametersFlags** domainGetSchedulerParametersFlags
- 3.66.2.50 **virDrvDomainGetSchedulerType** domainGetSchedulerType
- 3.66.2.51 **virDrvDomainGetSecurityLabel** domainGetSecurityLabel
- 3.66.2.52 **virDrvDomainGetSecurityLabelList** domainGetSecurityLabelList
- 3.66.2.53 **virDrvDomainGetState** domainGetState
- 3.66.2.54 **virDrvDomainGetVcpuPinInfo** domainGetVcpuPinInfo
- 3.66.2.55 **virDrvDomainGetVcpus** domainGetVcpus
- 3.66.2.56 **virDrvDomainGetVcpusFlags** domainGetVcpusFlags
- 3.66.2.57 **virDrvDomainGetXMLDesc** domainGetXMLDesc
- 3.66.2.58 **virDrvDomainHasCurrentSnapshot** domainHasCurrentSnapshot
- 3.66.2.59 **virDrvDomainHasManagedSaveImage** domainHasManagedSaveImage
- 3.66.2.60 **virDrvDomainInjectNMI** domainInjectNMI
- 3.66.2.61 **virDrvDomainInterfaceStats** domainInterfaceStats
- 3.66.2.62 **virDrvDomainsIsActive** domainsIsActive

- 3.66.2.63 **virDrvDomainIsPersistent** domainIsPersistent
- 3.66.2.64 **virDrvDomainIsUpdated** domainIsUpdated
- 3.66.2.65 **virDrvDomainListAllSnapshots** domainListAllSnapshots
- 3.66.2.66 **virDrvDomainLookupByID** domainLookupByID
- 3.66.2.67 **virDrvDomainLookupByName** domainLookupByName
- 3.66.2.68 **virDrvDomainLookupByUUID** domainLookupByUUID
- 3.66.2.69 **virDrvDomainManagedSave** domainManagedSave
- 3.66.2.70 **virDrvDomainManagedSaveRemove** domainManagedSaveRemove
- 3.66.2.71 **virDrvDomainMemoryPeek** domainMemoryPeek
- 3.66.2.72 **virDrvDomainMemoryStats** domainMemoryStats
- 3.66.2.73 **virDrvDomainMigrateBegin3** domainMigrateBegin3
- 3.66.2.74 **virDrvDomainMigrateConfirm3** domainMigrateConfirm3
- 3.66.2.75 **virDrvDomainMigrateFinish** domainMigrateFinish
- 3.66.2.76 **virDrvDomainMigrateFinish2** domainMigrateFinish2
- 3.66.2.77 **virDrvDomainMigrateFinish3** domainMigrateFinish3
- 3.66.2.78 **virDrvDomainMigrateGetMaxSpeed** domainMigrateGetMaxSpeed
- 3.66.2.79 **virDrvDomainMigratePerform** domainMigratePerform
- 3.66.2.80 **virDrvDomainMigratePerform3** domainMigratePerform3
- 3.66.2.81 **virDrvDomainMigratePrepare** domainMigratePrepare
- 3.66.2.82 **virDrvDomainMigratePrepare2** domainMigratePrepare2
- 3.66.2.83 **virDrvDomainMigratePrepare3** domainMigratePrepare3
- 3.66.2.84 **virDrvDomainMigratePrepareTunnel** domainMigratePrepareTunnel
- 3.66.2.85 **virDrvDomainMigratePrepareTunnel3** domainMigratePrepareTunnel3
- 3.66.2.86 **virDrvDomainMigrateSetMaxDowntime** domainMigrateSetMaxDowntime
- 3.66.2.87 **virDrvDomainMigrateSetMaxSpeed** domainMigrateSetMaxSpeed
- 3.66.2.88 **virDrvDomainOpenConsole** domainOpenConsole
- 3.66.2.89 **virDrvDomainOpenGraphics** domainOpenGraphics
- 3.66.2.90 **virDrvDomainPinEmulator** domainPinEmulator

- 3.66.2.91 `virDrvDomainPinVcpu` `domainPinVcpu`
- 3.66.2.92 `virDrvDomainPinVcpuFlags` `domainPinVcpuFlags`
- 3.66.2.93 `virDrvDomainPMSuspendForDuration` `domainPMSuspendForDuration`
- 3.66.2.94 `virDrvDomainPMWakeup` `domainPMWakeup`
- 3.66.2.95 `virDrvDomainReboot` `domainReboot`
- 3.66.2.96 `virDrvDomainReset` `domainReset`
- 3.66.2.97 `virDrvDomainRestore` `domainRestore`
- 3.66.2.98 `virDrvDomainRestoreFlags` `domainRestoreFlags`
- 3.66.2.99 `virDrvDomainResume` `domainResume`
- 3.66.2.100 `virDrvDomainRevertToSnapshot` `domainRevertToSnapshot`
- 3.66.2.101 `virDrvDomainSave` `domainSave`
- 3.66.2.102 `virDrvDomainSaveFlags` `domainSaveFlags`
- 3.66.2.103 `virDrvDomainSavelImageDefineXML` `domainSavelImageDefineXML`
- 3.66.2.104 `virDrvDomainSavelImageGetXMLDesc` `domainSavelImageGetXMLDesc`
- 3.66.2.105 `virDrvDomainScreenshot` `domainScreenshot`
- 3.66.2.106 `virDrvDomainSendKey` `domainSendKey`
- 3.66.2.107 `virDrvDomainSetAutostart` `domainSetAutostart`
- 3.66.2.108 `virDrvDomainSetBlkioParameters` `domainSetBlkioParameters`
- 3.66.2.109 `virDrvDomainSetBlockIoTune` `domainSetBlockIoTune`
- 3.66.2.110 `virDrvDomainSetInterfaceParameters` `domainSetInterfaceParameters`
- 3.66.2.111 `virDrvDomainSetMaxMemory` `domainSetMaxMemory`
- 3.66.2.112 `virDrvDomainSetMemory` `domainSetMemory`
- 3.66.2.113 `virDrvDomainSetMemoryFlags` `domainSetMemoryFlags`
- 3.66.2.114 `virDrvDomainSetMemoryParameters` `domainSetMemoryParameters`
- 3.66.2.115 `virDrvDomainSetMetadata` `domainSetMetadata`
- 3.66.2.116 `virDrvDomainSetNumaParameters` `domainSetNumaParameters`
- 3.66.2.117 `virDrvDomainSetSchedulerParameters` `domainSetSchedulerParameters`
- 3.66.2.118 `virDrvDomainSetSchedulerParametersFlags` `domainSetSchedulerParametersFlags`

- 3.66.2.119 **virDrvDomainSetVcpus** domainSetVcpus
- 3.66.2.120 **virDrvDomainSetVcpusFlags** domainSetVcpusFlags
- 3.66.2.121 **virDrvDomainShutdown** domainShutdown
- 3.66.2.122 **virDrvDomainShutdownFlags** domainShutdownFlags
- 3.66.2.123 **virDrvDomainSnapshotCreateXML** domainSnapshotCreateXML
- 3.66.2.124 **virDrvDomainSnapshotCurrent** domainSnapshotCurrent
- 3.66.2.125 **virDrvDomainSnapshotDelete** domainSnapshotDelete
- 3.66.2.126 **virDrvDomainSnapshotGetParent** domainSnapshotGetParent
- 3.66.2.127 **virDrvDomainSnapshotGetXMLDesc** domainSnapshotGetXMLDesc
- 3.66.2.128 **virDrvDomainSnapshotHasMetadata** domainSnapshotHasMetadata
- 3.66.2.129 **virDrvDomainSnapshotIsCurrent** domainSnapshotIsCurrent
- 3.66.2.130 **virDrvDomainSnapshotListAllChildren** domainSnapshotListAllChildren
- 3.66.2.131 **virDrvDomainSnapshotListChildrenNames** domainSnapshotListChildrenNames
- 3.66.2.132 **virDrvDomainSnapshotListNames** domainSnapshotListNames
- 3.66.2.133 **virDrvDomainSnapshotLookupByName** domainSnapshotLookupByName
- 3.66.2.134 **virDrvDomainSnapshotNum** domainSnapshotNum
- 3.66.2.135 **virDrvDomainSnapshotNumChildren** domainSnapshotNumChildren
- 3.66.2.136 **virDrvDomainSuspend** domainSuspend
- 3.66.2.137 **virDrvDomainUndefine** domainUndefine
- 3.66.2.138 **virDrvDomainUndefineFlags** domainUndefineFlags
- 3.66.2.139 **virDrvDomainUpdateDeviceFlags** domainUpdateDeviceFlags
- 3.66.2.140 **virDrvConnectDomainXMLFromNative** domainXMLFromNative
- 3.66.2.141 **virDrvConnectDomainXMLToNative** domainXMLToNative
- 3.66.2.142 **virDrvGetCapabilities** getCapabilities
- 3.66.2.143 **virDrvGetHostname** getHostname
- 3.66.2.144 **virDrvGetMaxVcpus** getMaxVcpus
- 3.66.2.145 **virDrvGetSysinfo** getSysinfo
- 3.66.2.146 **virDrvConnectIsAlive** isAlive



- 3.66.2.147 `virDrvConnectIsEncrypted` `isEncrypted`
- 3.66.2.148 `virDrvConnectIsSecure` `isSecure`
- 3.66.2.149 `virDrvGetLibVersion` `libvirtVersion`
- 3.66.2.150 `virDrvListAllDomains` `listAllDomains`
- 3.66.2.151 `virDrvListDefinedDomains` `listDefinedDomains`
- 3.66.2.152 `virDrvListDomains` `listDomains`
- 3.66.2.153 `const char*` `name`
- 3.66.2.154 `int` `no`
- 3.66.2.155 `virDrvNodeDeviceDetach` `nodeDeviceDetach`
- 3.66.2.156 `virDrvNodeDeviceReAttach` `nodeDeviceReAttach`
- 3.66.2.157 `virDrvNodeDeviceReset` `nodeDeviceReset`
- 3.66.2.158 `virDrvNodeGetCellsFreeMemory` `nodeGetCellsFreeMemory`
- 3.66.2.159 `virDrvNodeGetCPUStats` `nodeGetCPUStats`
- 3.66.2.160 `virDrvNodeGetFreeMemory` `nodeGetFreeMemory`
- 3.66.2.161 `virDrvNodeGetInfo` `nodeGetInfo`
- 3.66.2.162 `virDrvNodeGetMemoryParameters` `nodeGetMemoryParameters`
- 3.66.2.163 `virDrvNodeGetMemoryStats` `nodeGetMemoryStats`
- 3.66.2.164 `virDrvNodeGetSecurityModel` `nodeGetSecurityModel`
- 3.66.2.165 `virDrvNodeSetMemoryParameters` `nodeSetMemoryParameters`
- 3.66.2.166 `virDrvNodeSuspendForDuration` `nodeSuspendForDuration`
- 3.66.2.167 `virDrvNumOfDefinedDomains` `numOfDefinedDomains`
- 3.66.2.168 `virDrvNumOfDomains` `numOfDomains`
- 3.66.2.169 `virDrvOpen` `open`
- 3.66.2.170 `virDrvDomainQemuAgentCommand` `qemuDomainArbitraryAgentCommand`
- 3.66.2.171 `virDrvDomainQemuAttach` `qemuDomainAttach`
- 3.66.2.172 `virDrvDomainQemuMonitorCommand` `qemuDomainMonitorCommand`
- 3.66.2.173 `virDrvSetKeepAlive` `setKeepAlive`
- 3.66.2.174 `virDrvDrvSupportsFeature` `supports_feature`

3.66.2.175 **virDrvGetType** type

3.66.2.176 **virDrvGetVersion** version

The documentation for this struct was generated from the following file:

- [src/driver.h](#)

## 3.67 **\_virInterfaceDriver** Struct Reference

```
#include <driver.h>
```

### Data Fields

- `const char * name`
- `virDrvOpen open`
- `virDrvClose close`
- `virDrvNumOfInterfaces numOfInterfaces`
- `virDrvListInterfaces listInterfaces`
- `virDrvNumOfDefinedInterfaces numOfDefinedInterfaces`
- `virDrvListDefinedInterfaces listDefinedInterfaces`
- `virDrvListAllInterfaces listAllInterfaces`
- `virDrvInterfaceLookupByName interfaceLookupByName`
- `virDrvInterfaceLookupByMACString interfaceLookupByMACString`
- `virDrvInterfaceGetXMLDesc interfaceGetXMLDesc`
- `virDrvInterfaceDefineXML interfaceDefineXML`
- `virDrvInterfaceUndefine interfaceUndefine`
- `virDrvInterfaceCreate interfaceCreate`
- `virDrvInterfaceDestroy interfaceDestroy`
- `virDrvInterfaceIsActive interfaceIsActive`
- `virDrvInterfaceChangeBegin interfaceChangeBegin`
- `virDrvInterfaceChangeCommit interfaceChangeCommit`
- `virDrvInterfaceChangeRollback interfaceChangeRollback`

### 3.67.1 Detailed Description

[\\_virInterfaceDriver](#):

Structure associated to a network interface driver, defining the various entry points for it.

All drivers must support the following fields/methods:

- `open`
- `close`

### 3.67.2 Field Documentation

3.67.2.1 **virDrvClose** close

3.67.2.2 **virDrvInterfaceChangeBegin** interfaceChangeBegin

3.67.2.3 **virDrvInterfaceChangeCommit** interfaceChangeCommit

- 3.67.2.4 `virDrvInterfaceChangeRollback` `interfaceChangeRollback`
- 3.67.2.5 `virDrvInterfaceCreate` `interfaceCreate`
- 3.67.2.6 `virDrvInterfaceDefineXML` `interfaceDefineXML`
- 3.67.2.7 `virDrvInterfaceDestroy` `interfaceDestroy`
- 3.67.2.8 `virDrvInterfaceGetXMLDesc` `interfaceGetXMLDesc`
- 3.67.2.9 `virDrvInterfaceIsActive` `interfaceIsActive`
- 3.67.2.10 `virDrvInterfaceLookupByMACString` `interfaceLookupByMACString`
- 3.67.2.11 `virDrvInterfaceLookupByName` `interfaceLookupByName`
- 3.67.2.12 `virDrvInterfaceUndefine` `interfaceUndefine`
- 3.67.2.13 `virDrvListAllInterfaces` `listAllInterfaces`
- 3.67.2.14 `virDrvListDefinedInterfaces` `listDefinedInterfaces`
- 3.67.2.15 `virDrvListInterfaces` `listInterfaces`
- 3.67.2.16 `const char* name`
- 3.67.2.17 `virDrvNumOfDefinedInterfaces` `numOfDefinedInterfaces`
- 3.67.2.18 `virDrvNumOfInterfaces` `numOfInterfaces`
- 3.67.2.19 `virDrvOpen` `open`

The documentation for this struct was generated from the following file:

- [src/driver.h](#)

## 3.68 POL Mod \_virNetworkDef Struct Reference

```
#include <network_conf.h>
```

### Data Fields

- unsigned char `uuid` [[VIR\\_UUID\\_BUFLen](#)]
- bool `uuid_specified`
- char \* `name`
- int `connections`
- char \* `bridge`
- char \* `bridge_type` POL New
- char \* `source_bridge` POL New
- char \* `domain`
- unsigned long `delay`
- unsigned int `stp:1`
- virMacAddr `mac`
- bool `mac_specified`

- int [forwardType](#)
- int [managed](#)
- size\_t [nForwardPfs](#)
- [virNetworkForwardPfDefPtr](#) [forwardPfs](#)
- size\_t [nForwardIfs](#)
- [virNetworkForwardIfDefPtr](#) [forwardIfs](#)
- size\_t [nips](#)
- [virNetworkIpDefPtr](#) [ips](#)
- [virNetworkDNSDefPtr](#) [dns](#)
- [virNetDevVPortProfilePtr](#) [virtPortProfile](#)
- size\_t [nPortGroups](#)
- [virPortGroupDefPtr](#) [portGroups](#)
- [virNetDevBandwidthPtr](#) [bandwidth](#)
- [virNetDevVlan](#) [vlan](#)
- size\_t [nTunnels](#) POL New
- [virTunnelDefPtr](#) [tunnels](#) POL New

### 3.68.1 Field Documentation

- 3.68.1.1 [virNetDevBandwidthPtr](#) [bandwidth](#)
- 3.68.1.2 [char\\*](#) [bridge](#)
- 3.68.1.3 [char\\*](#) [bridge\\_type](#)
- 3.68.1.4 [int](#) [connections](#)
- 3.68.1.5 [unsigned long](#) [delay](#)
- 3.68.1.6 [virNetworkDNSDefPtr](#) [dns](#)
- 3.68.1.7 [char\\*](#) [domain](#)
- 3.68.1.8 [virNetworkForwardIfDefPtr](#) [forwardIfs](#)
- 3.68.1.9 [virNetworkForwardPfDefPtr](#) [forwardPfs](#)
- 3.68.1.10 [int](#) [forwardType](#)
- 3.68.1.11 [virNetworkIpDefPtr](#) [ips](#)
- 3.68.1.12 [virMacAddr](#) [mac](#)
- 3.68.1.13 [bool](#) [mac\\_specified](#)
- 3.68.1.14 [int](#) [managed](#)
- 3.68.1.15 [char\\*](#) [name](#)
- 3.68.1.16 [size\\_t](#) [nForwardIfs](#)
- 3.68.1.17 [size\\_t](#) [nForwardPfs](#)
- 3.68.1.18 [size\\_t](#) [nips](#)

- 3.68.1.19 `size_t nPortGroups`
- 3.68.1.20 `size_t nTunnels`
- 3.68.1.21 `virPortGroupDefPtr portGroups`
- 3.68.1.22 `char* source_bridge`
- 3.68.1.23 `unsigned int stp`
- 3.68.1.24 `virTunnelDefPtr tunnels`
- 3.68.1.25 `unsigned char uuid[VIR_UUID_BUFLen]`
- 3.68.1.26 `bool uuid_specified`
- 3.68.1.27 `virNetDevVPortProfilePtr virtPortProfile`
- 3.68.1.28 `virNetDevVlan vlan`

The documentation for this struct was generated from the following file:

- `src/conf/network_conf.h`

## 3.69 \_virNetworkDHCPHostDef Struct Reference

```
#include <network_conf.h>
```

### Data Fields

- `char * mac`
- `char * name`
- `virSocketAddr ip`

### 3.69.1 Field Documentation

- 3.69.1.1 `virSocketAddr ip`
- 3.69.1.2 `char* mac`
- 3.69.1.3 `char* name`

The documentation for this struct was generated from the following file:

- `src/conf/network_conf.h`

## 3.70 \_virNetworkDHCPRangeDef Struct Reference

```
#include <network_conf.h>
```

## Data Fields

- virSocketAddr [start](#)
- virSocketAddr [end](#)

### 3.70.1 Field Documentation

3.70.1.1 virSocketAddr end

3.70.1.2 virSocketAddr start

The documentation for this struct was generated from the following file:

- src/conf/[network\\_conf.h](#)

## 3.71 \_virNetworkDNSDef Struct Reference

```
#include <network_conf.h>
```

## Data Fields

- unsigned int [ntxtrecords](#)
- [virNetworkDNSTxtRecordsDefPtr](#) txtrecords
- unsigned int [nhosts](#)
- [virNetworkDNSHostsDefPtr](#) hosts
- unsigned int [nsrvrecords](#)
- [virNetworkDNSSrvRecordsDefPtr](#) srvrecords

### 3.71.1 Field Documentation

3.71.1.1 virNetworkDNSHostsDefPtr hosts

3.71.1.2 unsigned int nhosts

3.71.1.3 unsigned int nsrvrecords

3.71.1.4 unsigned int ntxtrecords

3.71.1.5 virNetworkDNSSrvRecordsDefPtr srvrecords

3.71.1.6 virNetworkDNSTxtRecordsDefPtr txtrecords

The documentation for this struct was generated from the following file:

- src/conf/[network\\_conf.h](#)

## 3.72 \_virNetworkDNSHostsDef Struct Reference

```
#include <network_conf.h>
```

## Data Fields

- virSocketAddr [ip](#)
- int [nnames](#)
- char \*\* [names](#)

### 3.72.1 Field Documentation

#### 3.72.1.1 virSocketAddr ip

#### 3.72.1.2 char\*\* names

#### 3.72.1.3 int nnames

The documentation for this struct was generated from the following file:

- [src/conf/network\\_conf.h](#)

## 3.73 \_virNetworkDNSSrvRecordsDef Struct Reference

```
#include <network_conf.h>
```

## Data Fields

- char \* [domain](#)
- char \* [service](#)
- char \* [protocol](#)
- char \* [target](#)
- int [port](#)
- int [priority](#)
- int [weight](#)

### 3.73.1 Field Documentation

#### 3.73.1.1 char\* domain

#### 3.73.1.2 int port

#### 3.73.1.3 int priority

#### 3.73.1.4 char\* protocol

#### 3.73.1.5 char\* service

#### 3.73.1.6 char\* target

#### 3.73.1.7 int weight

The documentation for this struct was generated from the following file:

- [src/conf/network\\_conf.h](#)

### 3.74 `_virNetworkDNSTxtRecordsDef` Struct Reference

```
#include <network_conf.h>
```

#### Data Fields

- char \* [name](#)
- char \* [value](#)

#### 3.74.1 Field Documentation

3.74.1.1 char\* name

3.74.1.2 char\* value

The documentation for this struct was generated from the following file:

- [src/conf/network\\_conf.h](#)

### 3.75 `_virNetworkDriver` Struct Reference

```
#include <driver.h>
```

#### Data Fields

- const char \* [name](#)
- [virDrvOpen](#) open
- [virDrvClose](#) close
- [virDrvNumOfNetworks](#) numOfNetworks
- [virDrvListNetworks](#) listNetworks
- [virDrvNumOfDefinedNetworks](#) numOfDefinedNetworks
- [virDrvListDefinedNetworks](#) listDefinedNetworks
- [virDrvListAllNetworks](#) listAllNetworks
- [virDrvNetworkLookupByUUID](#) networkLookupByUUID
- [virDrvNetworkLookupByName](#) networkLookupByName
- [virDrvNetworkCreateXML](#) networkCreateXML
- [virDrvNetworkDefineXML](#) networkDefineXML
- [virDrvNetworkUndefine](#) networkUndefine
- [virDrvNetworkUpdate](#) networkUpdate
- [virDrvNetworkCreate](#) networkCreate
- [virDrvNetworkDestroy](#) networkDestroy
- [virDrvNetworkGetXMLDesc](#) networkGetXMLDesc
- [virDrvNetworkGetBridgeName](#) networkGetBridgeName
- [virDrvNetworkGetBridgeType](#) networkGetBridgeType
- [virDrvNetworkGetAutostart](#) networkGetAutostart
- [virDrvNetworkSetAutostart](#) networkSetAutostart
- [virDrvNetworkIsActive](#) networkIsActive
- [virDrvNetworkIsPersistent](#) networkIsPersistent



### 3.75.1 Detailed Description

[`\_virNetworkDriver`](#):

Structure associated to a network virtualization driver, defining the various entry points for it.

All drivers must support the following fields/methods:

- `open`
- `close`

### 3.75.2 Field Documentation

- 3.75.2.1 `virDrvClose` `close`
- 3.75.2.2 `virDrvListAllNetworks` `listAllNetworks`
- 3.75.2.3 `virDrvListDefinedNetworks` `listDefinedNetworks`
- 3.75.2.4 `virDrvListNetworks` `listNetworks`
- 3.75.2.5 `const char*` `name`
- 3.75.2.6 `virDrvNetworkCreate` `networkCreate`
- 3.75.2.7 `virDrvNetworkCreateXML` `networkCreateXML`
- 3.75.2.8 `virDrvNetworkDefineXML` `networkDefineXML`
- 3.75.2.9 `virDrvNetworkDestroy` `networkDestroy`
- 3.75.2.10 `virDrvNetworkGetAutostart` `networkGetAutostart`
- 3.75.2.11 `virDrvNetworkGetBridgeName` `networkGetBridgeName`
- 3.75.2.12 `virDrvNetworkGetBridgeType` `networkGetBridgeType`
- 3.75.2.13 `virDrvNetworkGetXMLDesc` `networkGetXMLDesc`
- 3.75.2.14 `virDrvNetworkIsActive` `networkIsActive`
- 3.75.2.15 `virDrvNetworkIsPersistent` `networkIsPersistent`
- 3.75.2.16 `virDrvNetworkLookupByName` `networkLookupByName`
- 3.75.2.17 `virDrvNetworkLookupByUUID` `networkLookupByUUID`
- 3.75.2.18 `virDrvNetworkSetAutostart` `networkSetAutostart`
- 3.75.2.19 `virDrvNetworkUndefine` `networkUndefine`
- 3.75.2.20 `virDrvNetworkUpdate` `networkUpdate`
- 3.75.2.21 `virDrvNumOfDefinedNetworks` `numOfDefinedNetworks`
- 3.75.2.22 `virDrvNumOfNetworks` `numOfNetworks`

### 3.75.2.23 virDrvOpen open

The documentation for this struct was generated from the following file:

- [src/driver.h](#)

## 3.76 \_virNetworkForwardIfDef Struct Reference

```
#include <network_conf.h>
```

### Data Fields

- int [type](#)
- union {
  - virDevicePCIAddress [pci](#)
  - char \* [dev](#)
- } [device](#)
- int [connections](#)

### 3.76.1 Field Documentation

3.76.1.1 int [connections](#)

3.76.1.2 char\* [dev](#)

3.76.1.3 union { ... } [device](#)

3.76.1.4 virDevicePCIAddress [pci](#)

3.76.1.5 int [type](#)

The documentation for this struct was generated from the following file:

- [src/conf/network\\_conf.h](#)

## 3.77 \_virNetworkForwardPfDef Struct Reference

```
#include <network_conf.h>
```

### Data Fields

- char \* [dev](#)
- int [connections](#)

### 3.77.1 Field Documentation

3.77.1.1 int [connections](#)

### 3.77.1.2 char\* dev

The documentation for this struct was generated from the following file:

- [src/conf/network\\_conf.h](#)

## 3.78 \_virNetworkIpDef Struct Reference

```
#include <network_conf.h>
```

### Data Fields

- char \* [family](#)
- virSocketAddr [address](#)
- unsigned int [prefix](#)
- virSocketAddr [netmask](#)
- unsigned int [nranges](#)
- [virNetworkDHCPRangeDefPtr](#) [ranges](#)
- unsigned int [nhosts](#)
- [virNetworkDHCPHostDefPtr](#) [hosts](#)
- char \* [tftpboot](#)
- char \* [bootfile](#)
- virSocketAddr [bootserver](#)

### 3.78.1 Field Documentation

#### 3.78.1.1 virSocketAddr address

#### 3.78.1.2 char\* bootfile

#### 3.78.1.3 virSocketAddr bootserver

#### 3.78.1.4 char\* family

#### 3.78.1.5 virNetworkDHCPHostDefPtr hosts

#### 3.78.1.6 virSocketAddr netmask

#### 3.78.1.7 unsigned int nhosts

#### 3.78.1.8 unsigned int nranges

#### 3.78.1.9 unsigned int prefix

#### 3.78.1.10 virNetworkDHCPRangeDefPtr ranges

#### 3.78.1.11 char\* tftpboot

The documentation for this struct was generated from the following file:

- [src/conf/network\\_conf.h](#)

## 3.79 `_virNetworkObj` Struct Reference

```
#include <network_conf.h>
```

### Data Fields

- `virMutex` [lock](#)
- `pid_t` [dnsmasqPid](#)
- `pid_t` [radvdPid](#)
- unsigned int [active](#): 1
- unsigned int [autostart](#): 1
- unsigned int [persistent](#): 1
- [virNetworkDefPtr](#) [def](#)
- [virNetworkDefPtr](#) [newDef](#)

### 3.79.1 Field Documentation

3.79.1.1 unsigned int [active](#)

3.79.1.2 unsigned int [autostart](#)

3.79.1.3 [virNetworkDefPtr](#) [def](#)

3.79.1.4 `pid_t` [dnsmasqPid](#)

3.79.1.5 `virMutex` [lock](#)

3.79.1.6 [virNetworkDefPtr](#) [newDef](#)

3.79.1.7 unsigned int [persistent](#)

3.79.1.8 `pid_t` [radvdPid](#)

The documentation for this struct was generated from the following file:

- `src/conf/`[network\\_conf.h](#)

## 3.80 `_virNetworkObjList` Struct Reference

```
#include <network_conf.h>
```

### Data Fields

- unsigned int [count](#)
- [virNetworkObjPtr](#) \* [objs](#)

### 3.80.1 Field Documentation

3.80.1.1 unsigned int [count](#)

### 3.80.1.2 virNetworkObjPtr\* objs

The documentation for this struct was generated from the following file:

- [src/conf/network\\_conf.h](#)

## 3.81 \_virNodeCPUStats Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- char [field](#) [[VIR\\_NODE\\_CPU\\_STATS\\_FIELD\\_LENGTH](#)]
- unsigned long long [value](#)

### 3.81.1 Field Documentation

3.81.1.1 char [field](#)[[VIR\\_NODE\\_CPU\\_STATS\\_FIELD\\_LENGTH](#)]

3.81.1.2 unsigned long long [value](#)

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.82 \_virNodeInfo Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- char [model](#) [32]
- unsigned long [memory](#)
- unsigned int [cpus](#)
- unsigned int [mhz](#)
- unsigned int [nodes](#)
- unsigned int [sockets](#)
- unsigned int [cores](#)
- unsigned int [threads](#)

### 3.82.1 Field Documentation

3.82.1.1 unsigned int [cores](#)

3.82.1.2 unsigned int [cpus](#)

3.82.1.3 unsigned long [memory](#)

3.82.1.4 unsigned int [mhz](#)

3.82.1.5 char model[32]

3.82.1.6 unsigned int nodes

3.82.1.7 unsigned int sockets

3.82.1.8 unsigned int threads

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.83 \_virNodeMemoryStats Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- char [field](#) [[VIR\\_NODE\\_MEMORY\\_STATS\\_FIELD\\_LENGTH](#)]
- unsigned long long [value](#)

### 3.83.1 Field Documentation

3.83.1.1 char field[VIR\_NODE\_MEMORY\_STATS\_FIELD\_LENGTH]

3.83.1.2 unsigned long long value

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.84 \_virNWFilterDriver Struct Reference

```
#include <driver.h>
```

### Data Fields

- const char \* [name](#)
- [virDrvOpen](#) open
- [virDrvClose](#) close
- [virDrvConnectNumOfNWFilters](#) numOfNWFilters
- [virDrvConnectListNWFilters](#) listNWFilters
- [virDrvConnectListAllNWFilters](#) listAllNWFilters
- [virDrvNWFilterLookupByName](#) nwfilterLookupByName
- [virDrvNWFilterLookupByUUID](#) nwfilterLookupByUUID
- [virDrvNWFilterDefineXML](#) defineXML
- [virDrvNWFilterUndefine](#) undefine
- [virDrvNWFilterGetXMLDesc](#) getXMLDesc

### 3.84.1 Detailed Description

[\\_virNWFilterDriver](#):

Structure associated to a network filter driver, defining the various entry points for it.

All drivers must support the following fields/methods:

- `open`
- `close`

### 3.84.2 Field Documentation

3.84.2.1 `virDrvClose` `close`

3.84.2.2 `virDrvNWFilterDefineXML` `defineXML`

3.84.2.3 `virDrvNWFilterGetXMLDesc` `getXMLDesc`

3.84.2.4 `virDrvConnectListAllNWFilters` `listAllNWFilters`

3.84.2.5 `virDrvConnectListNWFilters` `listNWFilters`

3.84.2.6 `const char*` `name`

3.84.2.7 `virDrvConnectNumOfNWFilters` `numOfNWFilters`

3.84.2.8 `virDrvNWFilterLookupByName` `nwfilterLookupByName`

3.84.2.9 `virDrvNWFilterLookupByUUID` `nwfilterLookupByUUID`

3.84.2.10 `virDrvOpen` `open`

3.84.2.11 `virDrvNWFilterUndefine` `undefine`

The documentation for this struct was generated from the following file:

- `src/driver.h`

## 3.85 \_virPortGroupDef Struct Reference

```
#include <network_conf.h>
```

### Data Fields

- `char *` `name`
- `bool` `isDefault`
- `virNetDevVPortProfilePtr` `virtPortProfile`
- `virNetDevBandwidthPtr` `bandwidth`
- `virNetDevVlan` `vlan`

### 3.85.1 Field Documentation

3.85.1.1 `virNetDevBandwidthPtr` `bandwidth`

3.85.1.2 `bool` `isDefault`

3.85.1.3 `char*` `name`

3.85.1.4 `virNetDevVPortProfilePtr` `virtPortProfile`

3.85.1.5 `virNetDevVlan` `vlan`

The documentation for this struct was generated from the following file:

- `src/conf/network_conf.h`

## 3.86 `_virSecretDriver` Struct Reference

```
#include <driver.h>
```

### Data Fields

- `const char *` `name`
- `virDrvOpen` `open`
- `virDrvClose` `close`
- `virDrvNumOfSecrets` `numOfSecrets`
- `virDrvListSecrets` `listSecrets`
- `virDrvListAllSecrets` `listAllSecrets`
- `virDrvSecretLookupByUUID` `lookupByUUID`
- `virDrvSecretLookupByUsage` `lookupByUsage`
- `virDrvSecretDefineXML` `defineXML`
- `virDrvSecretGetXMLDesc` `getXMLDesc`
- `virDrvSecretSetValue` `setValue`
- `virDrvSecretGetValue` `getValue`
- `virDrvSecretUndefine` `undefine`

### 3.86.1 Detailed Description

`_virSecretDriver`:

Structure associated to a driver for storing secrets, defining the various entry points for it.

All drivers must support the following fields/methods:

- `open`
- `close`

### 3.86.2 Field Documentation

3.86.2.1 `virDrvClose` `close`

3.86.2.2 `virDrvSecretDefineXML` `defineXML`



- 3.86.2.3 `virDrvSecretGetValue` `getValue`
- 3.86.2.4 `virDrvSecretGetXMLDesc` `getXMLDesc`
- 3.86.2.5 `virDrvListAllSecrets` `listAllSecrets`
- 3.86.2.6 `virDrvListSecrets` `listSecrets`
- 3.86.2.7 `virDrvSecretLookupByUsage` `lookupByUsage`
- 3.86.2.8 `virDrvSecretLookupByUUID` `lookupByUUID`
- 3.86.2.9 `const char*` `name`
- 3.86.2.10 `virDrvNumOfSecrets` `numOfSecrets`
- 3.86.2.11 `virDrvOpen` `open`
- 3.86.2.12 `virDrvSecretSetValue` `setValue`
- 3.86.2.13 `virDrvSecretUndefine` `undefine`

The documentation for this struct was generated from the following file:

- [src/driver.h](#)

## 3.87 `_virSecurityDeviceLabelDef` Struct Reference

```
#include <domain_conf.h>
```

### Data Fields

- `char *` [model](#)
- `char *` [label](#)
- `bool` [norelabel](#)

### 3.87.1 Field Documentation

- 3.87.1.1 `char*` `label`
- 3.87.1.2 `char*` `model`
- 3.87.1.3 `bool` `norelabel`

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.88 `_virSecurityLabel` Struct Reference

```
#include <libvirt.h>
```

## Data Fields

- char [label](#) [[VIR\\_SECURITY\\_LABEL\\_BUFLLEN](#)]
- int [enforcing](#)

### 3.88.1 Detailed Description

virSecurityLabel:

a virSecurityLabel is a structure filled by [virDomainGetSecurityLabel\(\)](#), providing the security label and associated attributes for the specified domain.

### 3.88.2 Field Documentation

3.88.2.1 int [enforcing](#)

3.88.2.2 char [label](#)[[VIR\\_SECURITY\\_LABEL\\_BUFLLEN](#)]

The documentation for this struct was generated from the following file:

- include/libvirt/[libvirt.h](#)

## 3.89 \_virSecurityLabelDef Struct Reference

```
#include <domain_conf.h>
```

## Data Fields

- char \* [model](#)
- char \* [label](#)
- char \* [imagelabel](#)
- char \* [baselabel](#)
- int [type](#)
- bool [norelabel](#)
- bool [implicit](#)

### 3.89.1 Field Documentation

3.89.1.1 char\* [baselabel](#)

3.89.1.2 char\* [imagelabel](#)

3.89.1.3 bool [implicit](#)

3.89.1.4 char\* [label](#)

3.89.1.5 char\* [model](#)

3.89.1.6 bool [norelabel](#)

3.89.1.7 int [type](#)

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.h](#)

## 3.90 \_virSecurityModel Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- [char model \[VIR\\_SECURITY\\_MODEL\\_BUFLen\]](#)
- [char doi \[VIR\\_SECURITY\\_DOI\\_BUFLen\]](#)

### 3.90.1 Detailed Description

virSecurityModel:

a virSecurityModel is a structure filled by [virNodeGetSecurityModel\(\)](#), providing the per-hypervisor security model and DOI attributes for the specified domain.

### 3.90.2 Field Documentation

3.90.2.1 [char doi\[VIR\\_SECURITY\\_DOI\\_BUFLen\]](#)

3.90.2.2 [char model\[VIR\\_SECURITY\\_MODEL\\_BUFLen\]](#)

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.91 \_virStorageDriver Struct Reference

```
#include <driver.h>
```

### Data Fields

- [const char \\* name](#)
- [virDrvOpen open](#)
- [virDrvClose close](#)
- [virDrvConnectNumOfStoragePools numOfPools](#)
- [virDrvConnectListStoragePools listPools](#)
- [virDrvConnectNumOfDefinedStoragePools numOfDefinedPools](#)
- [virDrvConnectListDefinedStoragePools listDefinedPools](#)
- [virDrvConnectListAllStoragePools listAllPools](#)
- [virDrvConnectFindStoragePoolSources findPoolSources](#)
- [virDrvStoragePoolLookupByName poolLookupByName](#)
- [virDrvStoragePoolLookupByUUID poolLookupByUUID](#)
- [virDrvStoragePoolLookupByVolume poolLookupByVolume](#)
- [virDrvStoragePoolCreateXML poolCreateXML](#)
- [virDrvStoragePoolDefineXML poolDefineXML](#)
- [virDrvStoragePoolBuild poolBuild](#)
- [virDrvStoragePoolUndefine poolUndefine](#)

- [virDrvStoragePoolCreate](#) poolCreate
- [virDrvStoragePoolDestroy](#) poolDestroy
- [virDrvStoragePoolDelete](#) poolDelete
- [virDrvStoragePoolRefresh](#) poolRefresh
- [virDrvStoragePoolGetInfo](#) poolGetInfo
- [virDrvStoragePoolGetXMLDesc](#) poolGetXMLDesc
- [virDrvStoragePoolGetAutostart](#) poolGetAutostart
- [virDrvStoragePoolSetAutostart](#) poolSetAutostart
- [virDrvStoragePoolNumOfVolumes](#) poolNumOfVolumes
- [virDrvStoragePoolListVolumes](#) poolListVolumes
- [virDrvStoragePoolListAllVolumes](#) poolListAllVolumes
- [virDrvStorageVolLookupByName](#) volLookupByName
- [virDrvStorageVolLookupByKey](#) volLookupByKey
- [virDrvStorageVolLookupByPath](#) volLookupByPath
- [virDrvStorageVolCreateXML](#) volCreateXML
- [virDrvStorageVolCreateXMLFrom](#) volCreateXMLFrom
- [virDrvStorageVolDownload](#) volDownload
- [virDrvStorageVolUpload](#) volUpload
- [virDrvStorageVolDelete](#) volDelete
- [virDrvStorageVolWipe](#) volWipe
- [virDrvStorageVolWipePattern](#) volWipePattern
- [virDrvStorageVolGetInfo](#) volGetInfo
- [virDrvStorageVolGetXMLDesc](#) volGetXMLDesc
- [virDrvStorageVolGetPath](#) volGetPath
- [virDrvStorageVolResize](#) volResize
- [virDrvStoragePoolsActive](#) poolsActive
- [virDrvStoragePoolsPersistent](#) poolsPersistent

### 3.91.1 Detailed Description

[\\_virStorageDriver](#):

Structure associated to a storage driver, defining the various entry points for it.

All drivers must support the following fields/methods:

- open
- close

### 3.91.2 Field Documentation

3.91.2.1 [virDrvClose](#) close

3.91.2.2 [virDrvConnectFindStoragePoolSources](#) findPoolSources

3.91.2.3 [virDrvConnectListAllStoragePools](#) listAllPools

3.91.2.4 [virDrvConnectListDefinedStoragePools](#) listDefinedPools

3.91.2.5 [virDrvConnectListStoragePools](#) listPools

3.91.2.6 [const char\\*](#) name

3.91.2.7 [virDrvConnectNumOfDefinedStoragePools](#) numOfDefinedPools

- 3.91.2.8 `virDrvConnectNumOfStoragePools` `numOfPools`
- 3.91.2.9 `virDrvOpen` `open`
- 3.91.2.10 `virDrvStoragePoolBuild` `poolBuild`
- 3.91.2.11 `virDrvStoragePoolCreate` `poolCreate`
- 3.91.2.12 `virDrvStoragePoolCreateXML` `poolCreateXML`
- 3.91.2.13 `virDrvStoragePoolDefineXML` `poolDefineXML`
- 3.91.2.14 `virDrvStoragePoolDelete` `poolDelete`
- 3.91.2.15 `virDrvStoragePoolDestroy` `poolDestroy`
- 3.91.2.16 `virDrvStoragePoolGetAutostart` `poolGetAutostart`
- 3.91.2.17 `virDrvStoragePoolGetInfo` `poolGetInfo`
- 3.91.2.18 `virDrvStoragePoolGetXMLDesc` `poolGetXMLDesc`
- 3.91.2.19 `virDrvStoragePoolsActive` `pollsActive`
- 3.91.2.20 `virDrvStoragePoolsPersistent` `pollsPersistent`
- 3.91.2.21 `virDrvStoragePoolListAllVolumes` `poolListAllVolumes`
- 3.91.2.22 `virDrvStoragePoolListVolumes` `poolListVolumes`
- 3.91.2.23 `virDrvStoragePoolLookupByName` `poolLookupByName`
- 3.91.2.24 `virDrvStoragePoolLookupByUUID` `poolLookupByUUID`
- 3.91.2.25 `virDrvStoragePoolLookupByVolume` `poolLookupByVolume`
- 3.91.2.26 `virDrvStoragePoolNumOfVolumes` `poolNumOfVolumes`
- 3.91.2.27 `virDrvStoragePoolRefresh` `poolRefresh`
- 3.91.2.28 `virDrvStoragePoolSetAutostart` `poolSetAutostart`
- 3.91.2.29 `virDrvStoragePoolUndefine` `poolUndefine`
- 3.91.2.30 `virDrvStorageVolCreateXML` `volCreateXML`
- 3.91.2.31 `virDrvStorageVolCreateXMLFrom` `volCreateXMLFrom`
- 3.91.2.32 `virDrvStorageVolDelete` `volDelete`
- 3.91.2.33 `virDrvStorageVolDownload` `volDownload`
- 3.91.2.34 `virDrvStorageVolGetInfo` `volGetInfo`
- 3.91.2.35 `virDrvStorageVolGetPath` `volGetPath`

- 3.91.2.36 `virDrvStorageVolGetXMLDesc` `volGetXMLDesc`
- 3.91.2.37 `virDrvStorageVolLookupByKey` `volLookupByKey`
- 3.91.2.38 `virDrvStorageVolLookupByName` `volLookupByName`
- 3.91.2.39 `virDrvStorageVolLookupByPath` `volLookupByPath`
- 3.91.2.40 `virDrvStorageVolResize` `volResize`
- 3.91.2.41 `virDrvStorageVolUpload` `volUpload`
- 3.91.2.42 `virDrvStorageVolWipe` `volWipe`
- 3.91.2.43 `virDrvStorageVolWipePattern` `volWipePattern`

The documentation for this struct was generated from the following file:

- `src/driver.h`

## 3.92 `_virStoragePoolInfo` Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- `int` `state`
- `unsigned long long` `capacity`
- `unsigned long long` `allocation`
- `unsigned long long` `available`

### 3.92.1 Field Documentation

- 3.92.1.1 `unsigned long long` `allocation`
- 3.92.1.2 `unsigned long long` `available`
- 3.92.1.3 `unsigned long long` `capacity`
- 3.92.1.4 `int` `state`

The documentation for this struct was generated from the following file:

- `include/libvirt/libvirt.h`

## 3.93 `_virStorageVolInfo` Struct Reference

```
#include <libvirt.h>
```

## Data Fields

- int [type](#)
- unsigned long long [capacity](#)
- unsigned long long [allocation](#)

### 3.93.1 Field Documentation

3.93.1.1 unsigned long long [allocation](#)

3.93.1.2 unsigned long long [capacity](#)

3.93.1.3 int [type](#)

The documentation for this struct was generated from the following file:

- include/libvirt/[libvirt.h](#)

## 3.94 \_virStreamDriver Struct Reference

```
#include <driver.h>
```

## Data Fields

- [virDrvStreamSend](#) [streamSend](#)
- [virDrvStreamRecv](#) [streamRecv](#)
- [virDrvStreamEventAddCallback](#) [streamAddCallback](#)
- [virDrvStreamEventUpdateCallback](#) [streamUpdateCallback](#)
- [virDrvStreamEventRemoveCallback](#) [streamRemoveCallback](#)
- [virDrvStreamFinish](#) [streamFinish](#)
- [virDrvStreamAbort](#) [streamAbort](#)

### 3.94.1 Field Documentation

3.94.1.1 [virDrvStreamAbort](#) [streamAbort](#)

3.94.1.2 [virDrvStreamEventAddCallback](#) [streamAddCallback](#)

3.94.1.3 [virDrvStreamFinish](#) [streamFinish](#)

3.94.1.4 [virDrvStreamRecv](#) [streamRecv](#)

3.94.1.5 [virDrvStreamEventRemoveCallback](#) [streamRemoveCallback](#)

3.94.1.6 [virDrvStreamSend](#) [streamSend](#)

3.94.1.7 [virDrvStreamEventUpdateCallback](#) [streamUpdateCallback](#)

The documentation for this struct was generated from the following file:

- src/[driver.h](#)

### 3.95 **POL New** \_virTunnelDef Struct Reference

```
#include <network_conf.h>
```

#### Data Fields

- char \* [device](#)
- char \* [remote\\_ip](#)
- char \* [trunks](#)

#### 3.95.1 Field Documentation

3.95.1.1 char\* [device](#)

3.95.1.2 char\* [remote\\_ip](#)

3.95.1.3 char\* [trunks](#)

The documentation for this struct was generated from the following file:

- src/conf/[network\\_conf.h](#)

### 3.96 \_virTypedParameter Struct Reference

```
#include <libvirt.h>
```

#### Data Fields

- char [field](#) [[VIR\\_TYPED\\_PARAM\\_FIELD\\_LENGTH](#)]
- int [type](#)
- union {
  - int [i](#)
  - unsigned int [ui](#)
  - long long int [l](#)
  - unsigned long long int [ul](#)
  - double [d](#)
  - char [b](#)
  - char \* [s](#)
- } [value](#)

#### 3.96.1 Field Documentation

3.96.1.1 char [b](#)

3.96.1.2 double [d](#)

3.96.1.3 char [field](#)[[VIR\\_TYPED\\_PARAM\\_FIELD\\_LENGTH](#)]

3.96.1.4 int [i](#)



3.96.1.5 long long int l

3.96.1.6 char\* s

3.96.1.7 int type

3.96.1.8 unsigned int ui

3.96.1.9 unsigned long long int ul

3.96.1.10 union { ... } value

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.97 \_virVcpuInfo Struct Reference

```
#include <libvirt.h>
```

### Data Fields

- unsigned int [number](#)
- int [state](#)
- unsigned long long [cpuTime](#)
- int [cpu](#)

### 3.97.1 Field Documentation

3.97.1.1 int cpu

3.97.1.2 unsigned long long cpuTime

3.97.1.3 unsigned int number

3.97.1.4 int state

The documentation for this struct was generated from the following file:

- [include/libvirt/libvirt.h](#)

## 3.98 network\_driver Struct Reference

### Data Fields

- virMutex [lock](#)
- [virNetworkObjList](#) networks
- iptablesContext \* [iptables](#)
- char \* [networkConfigDir](#)
- char \* [networkAutostartDir](#)
- char \* [logDir](#)
- dnsmasqCapsPtr [dnsmasqCaps](#)

### 3.98.1 Field Documentation

3.98.1.1 `dnsmasqCapsPtr dnsmasqCaps`

3.98.1.2 `iptablesContext* iptables`

3.98.1.3 `virMutex lock`

3.98.1.4 `char* logDir`

3.98.1.5 `char* networkAutostartDir`

3.98.1.6 `char* networkConfigDir`

3.98.1.7 `virNetworkObjList networks`

The documentation for this struct was generated from the following file:

- `src/network/bridge_driver.c`

## 3.99 virDomainIDDData Struct Reference

### Data Fields

- int `numids`
- int `maxids`
- int \* `ids`

### 3.99.1 Field Documentation

3.99.1.1 `int* ids`

3.99.1.2 `int maxids`

3.99.1.3 `int numids`

The documentation for this struct was generated from the following file:

- `src/conf/domain_conf.c`

## 3.100 virDomainListData Struct Reference

### Data Fields

- `virConnectPtr conn`
- `virDomainPtr * domains`
- unsigned int `flags`
- int `ndomains`
- bool `error`

### 3.100.1 Field Documentation

- 3.100.1.1 virConnectPtr conn
- 3.100.1.2 virDomainPtr\* domains
- 3.100.1.3 bool error
- 3.100.1.4 unsigned int flags
- 3.100.1.5 int ndomains

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.c](#)

## 3.101 virDomainNameData Struct Reference

### Data Fields

- int [oom](#)
- int [numnames](#)
- int [maxnames](#)
- char \*\*const [names](#)

### 3.101.1 Field Documentation

- 3.101.1.1 int maxnames
- 3.101.1.2 char\*\* const names
- 3.101.1.3 int numnames
- 3.101.1.4 int oom

The documentation for this struct was generated from the following file:

- [src/conf/domain\\_conf.c](#)

## 3.102 vshNetworkList Struct Reference

### Data Fields

- [virNetworkPtr](#) \* [nets](#)
- size\_t [nnets](#)

### 3.102.1 Field Documentation

- 3.102.1.1 virNetworkPtr\* nets
- 3.102.1.2 size\_t nnets

The documentation for this struct was generated from the following file:

- [tools/virsh-network.c](#)

## Chapter 4

# File Documentation

### 4.1 include/libvirt/libvirt.h File Reference

```
#include <sys/types.h>
```

#### Data Structures

- struct [\\_virDomainControlInfo](#)
- struct [\\_virDomainInfo](#)
- struct [\\_virSecurityLabel](#)
- struct [\\_virSecurityModel](#)
- struct [\\_virNodeInfo](#)
- struct [\\_virNodeCPUStats](#)
- struct [\\_virNodeMemoryStats](#)
- struct [\\_virTypedParameter](#)
- struct [\\_virDomainBlockStats](#)
- struct [\\_virDomainInterfaceStats](#)
- struct [\\_virDomainMemoryStat](#)
- struct [\\_virConnectCredential](#)
- struct [\\_virConnectAuth](#)
- struct [\\_virDomainBlockInfo](#)
- struct [\\_virVcpuInfo](#)
- struct [\\_virDomainBlockJobInfo](#)
- struct [\\_virDomainDiskError](#)
- struct [\\_virStoragePoolInfo](#)
- struct [\\_virStorageVolInfo](#)
- struct [\\_virDomainJobInfo](#)
- struct [\\_virDomainEventGraphicsAddress](#)
- struct [\\_virDomainEventGraphicsSubjectIdentity](#)
- struct [\\_virDomainEventGraphicsSubject](#)

#### Macros

- [#define VIR\\_DEPRECATED](#) /\* nothing \*/
- [#define VIR\\_EXPORT\\_VAR](#) extern
- [#define VIR\\_SECURITY\\_LABEL\\_BUFLen](#) (4096 + 1)
- [#define VIR\\_SECURITY\\_MODEL\\_BUFLen](#) (256 + 1)
- [#define VIR\\_SECURITY\\_DOI\\_BUFLen](#) (256 + 1)

- #define VIR\_NODE\_CPU\_STATS\_FIELD\_LENGTH 80
- #define VIR\_NODE\_CPU\_STATS\_KERNEL "kernel"
- #define VIR\_NODE\_CPU\_STATS\_USER "user"
- #define VIR\_NODE\_CPU\_STATS\_IDLE "idle"
- #define VIR\_NODE\_CPU\_STATS\_IOWAIT "iowait"
- #define VIR\_NODE\_CPU\_STATS\_UTILIZATION "utilization"
- #define VIR\_NODE\_MEMORY\_STATS\_FIELD\_LENGTH 80
- #define VIR\_NODE\_MEMORY\_STATS\_TOTAL "total"
- #define VIR\_NODE\_MEMORY\_STATS\_FREE "free"
- #define VIR\_NODE\_MEMORY\_STATS\_BUFFERS "buffers"
- #define VIR\_NODE\_MEMORY\_STATS\_CACHED "cached"
- #define VIR\_TYPED\_PARAM\_FIELD\_LENGTH 80
- #define VIR\_DOMAIN\_SCHEDULER\_CPU\_SHARES "cpu\_shares"
- #define VIR\_DOMAIN\_SCHEDULER\_VCPU\_PERIOD "vcpu\_period"
- #define VIR\_DOMAIN\_SCHEDULER\_VCPU\_QUOTA "vcpu\_quota"
- #define VIR\_DOMAIN\_SCHEDULER\_EMULATOR\_PERIOD "emulator\_period"
- #define VIR\_DOMAIN\_SCHEDULER\_EMULATOR\_QUOTA "emulator\_quota"
- #define VIR\_DOMAIN\_SCHEDULER\_WEIGHT "weight"
- #define VIR\_DOMAIN\_SCHEDULER\_CAP "cap"
- #define VIR\_DOMAIN\_SCHEDULER\_RESERVATION "reservation"
- #define VIR\_DOMAIN\_SCHEDULER\_LIMIT "limit"
- #define VIR\_DOMAIN\_SCHEDULER\_SHARES "shares"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_FIELD\_LENGTH VIR\_TYPED\_PARAM\_FIELD\_LENGTH
- #define VIR\_DOMAIN\_BLOCK\_STATS\_READ\_BYTES "rd\_bytes"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_READ\_REQ "rd\_operations"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_READ\_TOTAL\_TIMES "rd\_total\_times"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_WRITE\_BYTES "wr\_bytes"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_WRITE\_REQ "wr\_operations"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_WRITE\_TOTAL\_TIMES "wr\_total\_times"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_FLUSH\_REQ "flush\_operations"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_FLUSH\_TOTAL\_TIMES "flush\_total\_times"
- #define VIR\_DOMAIN\_BLOCK\_STATS\_ERRS "errs"
- #define VIR\_NODEINFO\_MAXCPUS(nodeinfo) ((nodeinfo).nodes\*(nodeinfo).sockets\*(nodeinfo).cores\*(nodeinfo).threads)
- #define VIR\_UUID\_BUFLen (16)
- #define VIR\_UUID\_STRING\_BUFLen (36+1)
- #define LIBVIR\_VERSION\_NUMBER 10002
- #define VIR\_DOMAIN\_CPU\_STATS\_CPUTIME "cpu\_time"
- #define VIR\_DOMAIN\_CPU\_STATS\_USERTIME "user\_time"
- #define VIR\_DOMAIN\_CPU\_STATS\_SYSTEMTIME "system\_time"
- #define VIR\_DOMAIN\_CPU\_STATS\_VCPU\_TIME "vcpu\_time"
- #define VIR\_DOMAIN\_BLKIO\_WEIGHT "weight"
- #define VIR\_DOMAIN\_BLKIO\_DEVICE\_WEIGHT "device\_weight"
- #define VIR\_DOMAIN\_MEMORY\_PARAM\_UNLIMITED 9007199254740991LL /\* = INT64\_MAX >> 10 \*/
- #define VIR\_DOMAIN\_MEMORY\_HARD\_LIMIT "hard\_limit"
- #define VIR\_DOMAIN\_MEMORY\_SOFT\_LIMIT "soft\_limit"
- #define VIR\_DOMAIN\_MEMORY\_MIN\_GUARANTEE "min\_guarantee"
- #define VIR\_DOMAIN\_MEMORY\_SWAP\_HARD\_LIMIT "swap\_hard\_limit"
- #define VIR\_DOMAIN\_NUMA\_NODESET "numa\_nodeset"
- #define VIR\_DOMAIN\_NUMA\_MODE "numa\_mode"
- #define VIR\_DOMAIN\_BANDWIDTH\_IN\_AVERAGE "inbound.average"
- #define VIR\_DOMAIN\_BANDWIDTH\_IN\_PEAK "inbound.peak"
- #define VIR\_DOMAIN\_BANDWIDTH\_IN\_BURST "inbound.burst"
- #define VIR\_DOMAIN\_BANDWIDTH\_OUT\_AVERAGE "outbound.average"
- #define VIR\_DOMAIN\_BANDWIDTH\_OUT\_PEAK "outbound.peak"
- #define VIR\_DOMAIN\_BANDWIDTH\_OUT\_BURST "outbound.burst"

- #define `VIR_USE_CPU`(cpumap, cpu) (`cpumap[(cpu)/8] | = (1 < ((cpu)%8))`)
- #define `VIR_UNUSE_CPU`(cpumap, cpu) (`cpumap[(cpu)/8] &= ~(1 < ((cpu)%8))`)
- #define `VIR_CPU_MAPLEN`(cpu) (((cpu)+7)/8)
- #define `VIR_CPU_USABLE`(cpumaps, maplen, vcpu, cpu) (`cpumaps[((vcpu)*(maplen))+((cpu)/8)] & (1 < ((cpu)%8))`)
- #define `VIR_COPY_CPUMAP`(cpumaps, maplen, vcpu, cpumap) `memcpy(cpumap, &(cpumaps[(vcpu)*(maplen)]), (maplen))`
- #define `VIR_GET_CPUMAP`(cpumaps, maplen, vcpu) `&(cpumaps[(vcpu)*(maplen)])`
- #define `VIR_DOMAIN_BLOCK_IOTUNE_TOTAL_BYTES_SEC` "total\_bytes\_sec"
- #define `VIR_DOMAIN_BLOCK_IOTUNE_READ_BYTES_SEC` "read\_bytes\_sec"
- #define `VIR_DOMAIN_BLOCK_IOTUNE_WRITE_BYTES_SEC` "write\_bytes\_sec"
- #define `VIR_DOMAIN_BLOCK_IOTUNE_TOTAL_IOPS_SEC` "total\_iops\_sec"
- #define `VIR_DOMAIN_BLOCK_IOTUNE_READ_IOPS_SEC` "read\_iops\_sec"
- #define `VIR_DOMAIN_BLOCK_IOTUNE_WRITE_IOPS_SEC` "write\_iops\_sec"
- #define `VIR_DOMAIN_SEND_KEY_MAX_KEYS` 16
- #define `VIR_DOMAIN_EVENT_CALLBACK`(cb) (`(virConnectDomainEventGenericCallback)(cb)`)
- #define `VIR_DOMAIN_SCHED_FIELD_LENGTH` `VIR_TYPED_PARAM_FIELD_LENGTH`
- #define `_virSchedParameter` `_virTypedParameter`
- #define `VIR_DOMAIN_BLKIO_FIELD_LENGTH` `VIR_TYPED_PARAM_FIELD_LENGTH`
- #define `_virBlkioParameter` `_virTypedParameter`
- #define `VIR_DOMAIN_MEMORY_FIELD_LENGTH` `VIR_TYPED_PARAM_FIELD_LENGTH`
- #define `_virMemoryParameter` `_virTypedParameter`
- #define `VIR_NODE_MEMORY_SHARED_PAGES_TO_SCAN` "shm\_pages\_to\_scan"
- #define `VIR_NODE_MEMORY_SHARED_SLEEP_MILLISECS` "shm\_sleep\_millisecs"
- #define `VIR_NODE_MEMORY_SHARED_PAGES_SHARED` "shm\_pages\_shared"
- #define `VIR_NODE_MEMORY_SHARED_PAGES_SHARING` "shm\_pages\_sharing"
- #define `VIR_NODE_MEMORY_SHARED_PAGES_UNSHARED` "shm\_pages\_unshared"
- #define `VIR_NODE_MEMORY_SHARED_PAGES_VOLATILE` "shm\_pages\_volatile"
- #define `VIR_NODE_MEMORY_SHARED_FULL_SCANS` "shm\_full\_scans"

## Typedefs

- typedef void(\* `virFreeCallback` )(void \*opaque)
- typedef struct \_virConnect `virConnect`
- typedef `virConnect *` `virConnectPtr`
- typedef struct \_virDomain `virDomain`
- typedef `virDomain *` `virDomainPtr`
- typedef struct  
    `_virDomainControlInfo` `virDomainControlInfo`
- typedef `virDomainControlInfo *` `virDomainControlInfoPtr`
- typedef struct \_virDomainInfo `virDomainInfo`
- typedef `virDomainInfo *` `virDomainInfoPtr`
- typedef struct \_virStream `virStream`
- typedef `virStream *` `virStreamPtr`
- typedef struct \_virSecurityLabel `virSecurityLabel`
- typedef `virSecurityLabel *` `virSecurityLabelPtr`
- typedef struct \_virSecurityModel `virSecurityModel`
- typedef `virSecurityModel *` `virSecurityModelPtr`
- typedef struct \_virNodeInfo `virNodeInfo`
- typedef struct \_virNodeCPUStats `virNodeCPUStats`
- typedef struct \_virNodeMemoryStats `virNodeMemoryStats`
- typedef struct \_virTypedParameter `virTypedParameter`
- typedef `virTypedParameter *` `virTypedParameterPtr`
- typedef struct \_virDomainBlockStats `virDomainBlockStatsStruct`

- typedef [virDomainBlockStatsStruct](#) \* [virDomainBlockStatsPtr](#)
- typedef struct  
    [\\_virDomainInterfaceStats](#) [virDomainInterfaceStatsStruct](#)
- typedef  
    [virDomainInterfaceStatsStruct](#) \* [virDomainInterfaceStatsPtr](#)
- typedef struct [\\_virDomainMemoryStat](#) [virDomainMemoryStatStruct](#)
- typedef [virDomainMemoryStatStruct](#) \* [virDomainMemoryStatPtr](#)
- typedef [virNodeInfo](#) \* [virNodeInfoPtr](#)
- typedef [virNodeCPUStats](#) \* [virNodeCPUStatsPtr](#)
- typedef [virNodeMemoryStats](#) \* [virNodeMemoryStatsPtr](#)
- typedef struct  
    [\\_virConnectCredential](#) [virConnectCredential](#)
- typedef [virConnectCredential](#) \* [virConnectCredentialPtr](#)
- typedef int(\* [virConnectAuthCallbackPtr](#) )( [virConnectCredentialPtr](#) cred, unsigned int ncred, void \*cbdata)
- typedef struct [\\_virConnectAuth](#) [virConnectAuth](#)
- typedef [virConnectAuth](#) \* [virConnectAuthPtr](#)
- typedef void(\* [virConnectCloseFunc](#) )( [virConnectPtr](#) conn, int reason, void \*opaque)
- typedef struct [\\_virDomainBlockInfo](#) [virDomainBlockInfo](#)
- typedef [virDomainBlockInfo](#) \* [virDomainBlockInfoPtr](#)
- typedef struct [\\_virVcpuInfo](#) [virVcpuInfo](#)
- typedef [virVcpuInfo](#) \* [virVcpuInfoPtr](#)
- typedef unsigned long long [virDomainBlockJobCursor](#)
- typedef struct  
    [\\_virDomainBlockJobInfo](#) [virDomainBlockJobInfo](#)
- typedef [virDomainBlockJobInfo](#) \* [virDomainBlockJobInfoPtr](#)
- typedef struct [\\_virDomainDiskError](#) [virDomainDiskError](#)
- typedef [virDomainDiskError](#) \* [virDomainDiskErrorPtr](#)
- typedef struct [\\_virNetwork](#) [virNetwork](#)
- typedef [virNetwork](#) \* [virNetworkPtr](#)
- typedef struct [\\_virInterface](#) [virInterface](#)
- typedef [virInterface](#) \* [virInterfacePtr](#)
- typedef struct [\\_virStoragePool](#) [virStoragePool](#)
- typedef [virStoragePool](#) \* [virStoragePoolPtr](#)
- typedef struct [\\_virStoragePoolInfo](#) [virStoragePoolInfo](#)
- typedef [virStoragePoolInfo](#) \* [virStoragePoolInfoPtr](#)
- typedef struct [\\_virStorageVol](#) [virStorageVol](#)
- typedef [virStorageVol](#) \* [virStorageVolPtr](#)
- typedef struct [\\_virStorageVolInfo](#) [virStorageVolInfo](#)
- typedef [virStorageVolInfo](#) \* [virStorageVolInfoPtr](#)
- typedef struct [\\_virNodeDevice](#) [virNodeDevice](#)
- typedef [virNodeDevice](#) \* [virNodeDevicePtr](#)
- typedef int(\* [virConnectDomainEventCallback](#) )( [virConnectPtr](#) conn, [virDomainPtr](#) dom, int event, int detail, void \*opaque)
- typedef void(\* [virEventHandleCallback](#) )(int watch, int fd, int events, void \*opaque)
- typedef int(\* [virEventAddHandleFunc](#) )(int fd, int event, [virEventHandleCallback](#) cb, void \*opaque, [virFreeCallback](#) ff)
- typedef void(\* [virEventUpdateHandleFunc](#) )(int watch, int event)
- typedef int(\* [virEventRemoveHandleFunc](#) )(int watch)
- typedef void(\* [virEventTimeoutCallback](#) )(int timer, void \*opaque)
- typedef int(\* [virEventAddTimeoutFunc](#) )(int timeout, [virEventTimeoutCallback](#) cb, void \*opaque, [virFreeCallback](#) ff)
- typedef void(\* [virEventUpdateTimeoutFunc](#) )(int timer, int timeout)
- typedef int(\* [virEventRemoveTimeoutFunc](#) )(int timer)
- typedef struct [\\_virSecret](#) [virSecret](#)
- typedef [virSecret](#) \* [virSecretPtr](#)



- typedef int(\* [virStreamSourceFunc](#) )(virStreamPtr st, char \*data, size\_t nbytes, void \*opaque)
- typedef int(\* [virStreamSinkFunc](#) )(virStreamPtr st, const char \*data, size\_t nbytes, void \*opaque)
- typedef void(\* [virStreamEventCallback](#) )(virStreamPtr stream, int events, void \*opaque)
- typedef struct [\\_virDomainJobInfo](#) virDomainJobInfo
- typedef [virDomainJobInfo](#) \* [virDomainJobInfoPtr](#)
- typedef struct [\\_virDomainSnapshot](#) virDomainSnapshot
- typedef [virDomainSnapshot](#) \* [virDomainSnapshotPtr](#)
- typedef void(\* [virConnectDomainEventGenericCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, void \*opaque)
- typedef void(\* [virConnectDomainEventRTCChangeCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, long long utcoffset, void \*opaque)
- typedef void(\* [virConnectDomainEventWatchdogCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, int action, void \*opaque)
- typedef void(\* [virConnectDomainEventIOErrorCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, const char \*srcPath, const char \*devAlias, int action, void \*opaque)
- typedef void(\* [virConnectDomainEventIOErrorReasonCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, const char \*srcPath, const char \*devAlias, int action, const char \*reason, void \*opaque)
- typedef struct [\\_virDomainEventGraphicsAddress](#) virDomainEventGraphicsAddress
- typedef [virDomainEventGraphicsAddress](#) \* [virDomainEventGraphicsAddressPtr](#)
- typedef struct [\\_virDomainEventGraphicsSubjectIdentity](#) virDomainEventGraphicsSubjectIdentity
- typedef [virDomainEventGraphicsSubjectIdentity](#) \* [virDomainEventGraphicsSubjectIdentityPtr](#)
- typedef struct [\\_virDomainEventGraphicsSubject](#) virDomainEventGraphicsSubject
- typedef [virDomainEventGraphicsSubject](#) \* [virDomainEventGraphicsSubjectPtr](#)
- typedef void(\* [virConnectDomainEventGraphicsCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, int phase, [virDomainEventGraphicsAddressPtr](#) local, [virDomainEventGraphicsAddressPtr](#) remote, const char \*authScheme, [virDomainEventGraphicsSubjectPtr](#) subject, void \*opaque)
- typedef void(\* [virConnectDomainEventBlockJobCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, const char \*disk, int type, int status, void \*opaque)
- typedef void(\* [virConnectDomainEventDiskChangeCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, const char \*oldSrcPath, const char \*newSrcPath, const char \*devAlias, int reason, void \*opaque)
- typedef void(\* [virConnectDomainEventTrayChangeCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, const char \*devAlias, int reason, void \*opaque)
- typedef void(\* [virConnectDomainEventPMWakeupCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, int reason, void \*opaque)
- typedef void(\* [virConnectDomainEventPMSuspendCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, int reason, void \*opaque)
- typedef void(\* [virConnectDomainEventBalloonChangeCallback](#) )(virConnectPtr conn, [virDomainPtr](#) dom, unsigned long long actual, void \*opaque)
- typedef struct [\\_virNWFilter](#) virNWFilter
- typedef [virNWFilter](#) \* [virNWFilterPtr](#)
- typedef struct [\\_virTypedParameter](#) virSchedParameter
- typedef [virSchedParameter](#) \* [virSchedParameterPtr](#)
- typedef struct [\\_virTypedParameter](#) virBlkioParameter
- typedef [virBlkioParameter](#) \* [virBlkioParameterPtr](#)
- typedef struct [\\_virTypedParameter](#) virMemoryParameter
- typedef [virMemoryParameter](#) \* [virMemoryParameterPtr](#)

## Enumerations

- enum `virDomainState` {  
`VIR_DOMAIN_NOSTATE` = 0, `VIR_DOMAIN_RUNNING` = 1, `VIR_DOMAIN_BLOCKED` = 2, `VIR_DOMAIN_PAUSED` = 3,  
`VIR_DOMAIN_SHUTDOWN` = 4, `VIR_DOMAIN_SHUTOFF` = 5, `VIR_DOMAIN_CRASHED` = 6, `VIR_DOMAIN_PMSUSPENDED` = 7 }
- enum `virDomainNostateReason` { `VIR_DOMAIN_NOSTATE_UNKNOWN` = 0 }
- enum `virDomainRunningReason` {  
`VIR_DOMAIN_RUNNING_UNKNOWN` = 0, `VIR_DOMAIN_RUNNING_BOOTED` = 1, `VIR_DOMAIN_RUNNING_MIGRATED` = 2, `VIR_DOMAIN_RUNNING_RESTORED` = 3,  
`VIR_DOMAIN_RUNNING_FROM_SNAPSHOT` = 4, `VIR_DOMAIN_RUNNING_UNPAUSED` = 5, `VIR_DOMAIN_RUNNING_MIGRATION_CANCELED` = 6, `VIR_DOMAIN_RUNNING_SAVE_CANCELED` = 7,  
`VIR_DOMAIN_RUNNING_WAKEUP` = 8 }
- enum `virDomainBlockedReason` { `VIR_DOMAIN_BLOCKED_UNKNOWN` = 0 }
- enum `virDomainPausedReason` {  
`VIR_DOMAIN_PAUSED_UNKNOWN` = 0, `VIR_DOMAIN_PAUSED_USER` = 1, `VIR_DOMAIN_PAUSED_MIGRATION` = 2, `VIR_DOMAIN_PAUSED_SAVE` = 3,  
`VIR_DOMAIN_PAUSED_DUMP` = 4, `VIR_DOMAIN_PAUSED_IOERROR` = 5, `VIR_DOMAIN_PAUSED_WATCHDOG` = 6, `VIR_DOMAIN_PAUSED_FROM_SNAPSHOT` = 7,  
`VIR_DOMAIN_PAUSED_SHUTTING_DOWN` = 8 }
- enum `virDomainShutdownReason` { `VIR_DOMAIN_SHUTDOWN_UNKNOWN` = 0, `VIR_DOMAIN_SHUTDOWN_USER` = 1 }
- enum `virDomainShutoffReason` {  
`VIR_DOMAIN_SHUTOFF_UNKNOWN` = 0, `VIR_DOMAIN_SHUTOFF_SHUTDOWN` = 1, `VIR_DOMAIN_SHUTOFF_DESTROYED` = 2, `VIR_DOMAIN_SHUTOFF_CRASHED` = 3,  
`VIR_DOMAIN_SHUTOFF_MIGRATED` = 4, `VIR_DOMAIN_SHUTOFF_SAVED` = 5, `VIR_DOMAIN_SHUTOFF_FAILED` = 6, `VIR_DOMAIN_SHUTOFF_FROM_SNAPSHOT` = 7 }
- enum `virDomainCrashedReason` { `VIR_DOMAIN_CRASHED_UNKNOWN` = 0 }
- enum `virDomainPMSuspendedReason` { `VIR_DOMAIN_PMSUSPENDED_UNKNOWN` = 0 }
- enum `virDomainControlState` { `VIR_DOMAIN_CONTROL_OK` = 0, `VIR_DOMAIN_CONTROL_JOB` = 1, `VIR_DOMAIN_CONTROL_OCCUPIED` = 2, `VIR_DOMAIN_CONTROL_ERROR` = 3 }
- enum `virDomainModificationImpact` { `VIR_DOMAIN_AFFECT_CURRENT` = 0, `VIR_DOMAIN_AFFECT_LIVE` = 1 << 0, `VIR_DOMAIN_AFFECT_CONFIG` = 1 << 1 }
- enum `virDomainCreateFlags` {  
`VIR_DOMAIN_NONE` = 0, `VIR_DOMAIN_START_PAUSED` = 1 << 0, `VIR_DOMAIN_START_AUTODESTROY` = 1 << 1, `VIR_DOMAIN_START_BYPASS_CACHE` = 1 << 2,  
`VIR_DOMAIN_START_FORCE_BOOT` = 1 << 3 }
- enum `virNodeSuspendTarget` { `VIR_NODE_SUSPEND_TARGET_MEM` = 0, `VIR_NODE_SUSPEND_TARGET_DISK` = 1, `VIR_NODE_SUSPEND_TARGET_HYBRID` = 2 }
- enum `virNodeGetCPUStatsAllCPUs` { `VIR_NODE_CPU_STATS_ALL_CPUS` = -1 }
- enum `virNodeGetMemoryStatsAllCells` { `VIR_NODE_MEMORY_STATS_ALL_CELLS` = -1 }
- enum `virTypedParameterType` {  
`VIR_TYPED_PARAM_INT` = 1, `VIR_TYPED_PARAM_UINT` = 2, `VIR_TYPED_PARAM_LLONG` = 3, `VIR_TYPED_PARAM_ULLONG` = 4,  
`VIR_TYPED_PARAM_DOUBLE` = 5, `VIR_TYPED_PARAM_BOOLEAN` = 6, `VIR_TYPED_PARAM_STRING` = 7 }
- enum `virTypedParameterFlags` { `VIR_TYPED_PARAM_STRING_OKAY` = 1 << 2 }
- enum `virDomainMemoryStatTags` {  
`VIR_DOMAIN_MEMORY_STAT_SWAP_IN` = 0, `VIR_DOMAIN_MEMORY_STAT_SWAP_OUT` = 1, `VIR_DOMAIN_MEMORY_STAT_MAJOR_FAULT` = 2, `VIR_DOMAIN_MEMORY_STAT_MINOR_FAULT` = 3,  
`VIR_DOMAIN_MEMORY_STAT_UNUSED` = 4, `VIR_DOMAIN_MEMORY_STAT_AVAILABLE` = 5, `VIR_DOMAIN_MEMORY_STAT_ACTUAL_BALLOON` = 6, `VIR_DOMAIN_MEMORY_STAT_RSS` = 7,  
`VIR_DOMAIN_MEMORY_STAT_NR` = 8 }
- enum `virDomainCoreDumpFlags` {  
`VIR_DUMP_CRASH` = (1 << 0), `VIR_DUMP_LIVE` = (1 << 1), `VIR_DUMP_BYPASS_CACHE` = (1 << 2),  
`VIR_DUMP_RESET` = (1 << 3),  
`VIR_DUMP_MEMORY_ONLY` = (1 << 4) }

- enum `virDomainMigrateFlags` {  
`VIR_MIGRATE_LIVE` = (1 << 0), `VIR_MIGRATE_PEER2PEER` = (1 << 1), `VIR_MIGRATE_TUNNELLED` = (1 << 2), `VIR_MIGRATE_PERSIST_DEST` = (1 << 3),  
`VIR_MIGRATE_UNDEFINE_SOURCE` = (1 << 4), `VIR_MIGRATE_PAUSED` = (1 << 5), `VIR_MIGRATE_NON_SHARED_DISK` = (1 << 6), `VIR_MIGRATE_NON_SHARED_INC` = (1 << 7),  
`VIR_MIGRATE_CHANGE_PROTECTION` = (1 << 8), `VIR_MIGRATE_UNSAFE` = (1 << 9) }
- enum `virConnectFlags` { `VIR_CONNECT_RO` = (1 << 0), `VIR_CONNECT_NO_ALIASES` = (1 << 1) }
- enum `virConnectCredentialType` {  
`VIR_CRED_USERNAME` = 1, `VIR_CRED_AUTHNAME` = 2, `VIR_CRED_LANGUAGE` = 3, `VIR_CRED_CN-ONCE` = 4,  
`VIR_CRED_PASSPHRASE` = 5, `VIR_CRED_ECHOPROMPT` = 6, `VIR_CRED_NOECHOPROMPT` = 7, `VIR_CRED_REALM` = 8,  
`VIR_CRED_EXTERNAL` = 9 }
- enum `virConnectCloseReason` { `VIR_CONNECT_CLOSE_REASON_ERROR` = 0, `VIR_CONNECT_CLOSE_REASON_EOF` = 1, `VIR_CONNECT_CLOSE_REASON_KEEPLIVE` = 2, `VIR_CONNECT_CLOSE_REASON_CLIENT` = 3 }
- enum `virDomainShutdownFlagValues` { `VIR_DOMAIN_SHUTDOWN_DEFAULT` = 0, `VIR_DOMAIN_SHUTDOWN_ACPI_POWER_BTN` = (1 << 0), `VIR_DOMAIN_SHUTDOWN_GUEST_AGENT` = (1 << 1) }
- enum `virDomainRebootFlagValues` { `VIR_DOMAIN_REBOOT_DEFAULT` = 0, `VIR_DOMAIN_REBOOT_ACPI_POWER_BTN` = (1 << 0), `VIR_DOMAIN_REBOOT_GUEST_AGENT` = (1 << 1) }
- enum `virDomainDestroyFlagsValues` { `VIR_DOMAIN_DESTROY_DEFAULT` = 0, `VIR_DOMAIN_DESTROY_GRACEFUL` = 1 << 0 }
- enum `virDomainSaveRestoreFlags` { `VIR_DOMAIN_SAVE_BYPASS_CACHE` = 1 << 0, `VIR_DOMAIN_SAVE_RUNNING` = 1 << 1, `VIR_DOMAIN_SAVE_PAUSED` = 1 << 2 }
- enum `virDomainMemoryModFlags` { `VIR_DOMAIN_MEM_CURRENT` = `VIR_DOMAIN_AFFECT_CURRENT`, `VIR_DOMAIN_MEM_LIVE` = `VIR_DOMAIN_AFFECT_LIVE`, `VIR_DOMAIN_MEM_CONFIG` = `VIR_DOMAIN_AFFECT_CONFIG`, `VIR_DOMAIN_MEM_MAXIMUM` = (1 << 2) }
- enum `virDomainNumatuneMemMode` { `VIR_DOMAIN_NUMATUNE_MEM_STRICT` = 0, `VIR_DOMAIN_NUMATUNE_MEM_PREFERRED` = 1, `VIR_DOMAIN_NUMATUNE_MEM_INTERLEAVE` = 2 }
- enum `virDomainMetadataType` { `VIR_DOMAIN_METADATA_DESCRIPTION` = 0, `VIR_DOMAIN_METADATA_TITLE` = 1, `VIR_DOMAIN_METADATA_ELEMENT` = 2 }
- enum `virDomainXMLFlags` { `VIR_DOMAIN_XML_SECURE` = (1 << 0), `VIR_DOMAIN_XML_INACTIVE` = (1 << 1), `VIR_DOMAIN_XML_UPDATE_CPU` = (1 << 2) }
- enum `virDomainBlockResizeFlags` { `VIR_DOMAIN_BLOCK_RESIZE_BYTES` = 1 << 0 }
- enum `virDomainMemoryFlags` { `VIR_MEMORY_VIRTUAL` = 1 << 0, `VIR_MEMORY_PHYSICAL` = 1 << 1 }
- enum `virDomainUndefineFlagsValues` { `VIR_DOMAIN_UNDEFINE_MANAGED_SAVE` = (1 << 0), `VIR_DOMAIN_UNDEFINE_SNAPSHOTS_METADATA` = (1 << 1) }
- enum `virConnectListAllDomainsFlags` {  
`VIR_CONNECT_LIST_DOMAINS_ACTIVE` = 1 << 0, `VIR_CONNECT_LIST_DOMAINS_INACTIVE` = 1 << 1, `VIR_CONNECT_LIST_DOMAINS_PERSISTENT` = 1 << 2, `VIR_CONNECT_LIST_DOMAINS_TRANSIENT` = 1 << 3,  
`VIR_CONNECT_LIST_DOMAINS_RUNNING` = 1 << 4, `VIR_CONNECT_LIST_DOMAINS_PAUSED` = 1 << 5, `VIR_CONNECT_LIST_DOMAINS_SHUTOFF` = 1 << 6, `VIR_CONNECT_LIST_DOMAINS_OTHER` = 1 << 7,  
`VIR_CONNECT_LIST_DOMAINS_MANAGEDSAVE` = 1 << 8, `VIR_CONNECT_LIST_DOMAINS_NO_MANAGEDSAVE` = 1 << 9, `VIR_CONNECT_LIST_DOMAINS_AUTOSTART` = 1 << 10, `VIR_CONNECT_LIST_DOMAINS_NO_AUTOSTART` = 1 << 11,  
`VIR_CONNECT_LIST_DOMAINS_HAS_SNAPSHOT` = 1 << 12, `VIR_CONNECT_LIST_DOMAINS_NO_SNAPSHOT` = 1 << 13 }
- enum `virVcpuState` { `VIR_VCPU_OFFLINE` = 0, `VIR_VCPU_RUNNING` = 1, `VIR_VCPU_BLOCKED` = 2 }
- enum `virDomainVcpuFlags` { `VIR_DOMAIN_VCPU_CURRENT` = `VIR_DOMAIN_AFFECT_CURRENT`, `VIR_DOMAIN_VCPU_LIVE` = `VIR_DOMAIN_AFFECT_LIVE`, `VIR_DOMAIN_VCPU_CONFIG` = `VIR_DOMAIN_AFFECT_CONFIG`, `VIR_DOMAIN_VCPU_MAXIMUM` = (1 << 2) }
- enum `virDomainDeviceModifyFlags` { `VIR_DOMAIN_DEVICE_MODIFY_CURRENT` = `VIR_DOMAIN_AFFECT_CURRENT`, `VIR_DOMAIN_DEVICE_MODIFY_LIVE` = `VIR_DOMAIN_AFFECT_LIVE`, `VIR_DOMAIN_DEVICE_MODIFY_CONFIG` = `VIR_DOMAIN_AFFECT_CONFIG`, `VIR_DOMAIN_DEVICE_MODIFY_FORCE` = (1 << 2) }

- enum `virDomainBlockJobType` { `VIR_DOMAIN_BLOCK_JOB_TYPE_UNKNOWN` = 0, `VIR_DOMAIN_BLOCK_JOB_TYPE_PULL` = 1, `VIR_DOMAIN_BLOCK_JOB_TYPE_COPY` = 2, `VIR_DOMAIN_BLOCK_JOB_TYPE_COMMIT` = 3 }
- enum `virDomainBlockJobAbortFlags` { `VIR_DOMAIN_BLOCK_JOB_ABORT_ASYNC` = 1 << 0, `VIR_DOMAIN_BLOCK_JOB_ABORT_PIVOT` = 1 << 1 }
- enum `virDomainBlockRebaseFlags` { `VIR_DOMAIN_BLOCK_REBASE_SHALLOW` = 1 << 0, `VIR_DOMAIN_BLOCK_REBASE_REUSE_EXT` = 1 << 1, `VIR_DOMAIN_BLOCK_REBASE_COPY_RAW` = 1 << 2, `VIR_DOMAIN_BLOCK_REBASE_COPY` = 1 << 3 }
- enum `virDomainBlockCommitFlags` { `VIR_DOMAIN_BLOCK_COMMIT_SHALLOW` = 1 << 0, `VIR_DOMAIN_BLOCK_COMMIT_DELETE` = 1 << 1 }
- enum `virDomainDiskErrorCode` { `VIR_DOMAIN_DISK_ERROR_NONE` = 0, `VIR_DOMAIN_DISK_ERROR_UNSPEC` = 1, `VIR_DOMAIN_DISK_ERROR_NO_SPACE` = 2 }
- enum `virNetworkXMLFlags` { `VIR_NETWORK_XML_INACTIVE` = (1 << 0) }
- enum `virConnectListAllNetworksFlags` { `VIR_CONNECT_LIST_NETWORKS_INACTIVE` = 1 << 0, `VIR_CONNECT_LIST_NETWORKS_ACTIVE` = 1 << 1, `VIR_CONNECT_LIST_NETWORKS_PERSISTENT` = 1 << 2, `VIR_CONNECT_LIST_NETWORKS_TRANSIENT` = 1 << 3, `VIR_CONNECT_LIST_NETWORKS_AUTOSTART` = 1 << 4, `VIR_CONNECT_LIST_NETWORKS_NO_AUTOSTART` = 1 << 5 }
- enum `virNetworkUpdateCommand` { `VIR_NETWORK_UPDATE_COMMAND_NONE` = 0, `VIR_NETWORK_UPDATE_COMMAND_MODIFY` = 1, `VIR_NETWORK_UPDATE_COMMAND_DELETE` = 2, `VIR_NETWORK_UPDATE_COMMAND_ADD_LAST` = 3, `VIR_NETWORK_UPDATE_COMMAND_ADD_FIRST` = 4 }
- enum `virNetworkUpdateSection` { `VIR_NETWORK_SECTION_NONE` = 0, `VIR_NETWORK_SECTION_BRIDGE` = 1, `VIR_NETWORK_SECTION_DOMAIN` = 2, `VIR_NETWORK_SECTION_IP` = 3, `VIR_NETWORK_SECTION_IP_DHCP_HOST` = 4, `VIR_NETWORK_SECTION_IP_DHCP_RANGE` = 5, `VIR_NETWORK_SECTION_FORWARD` = 6, `VIR_NETWORK_SECTION_FORWARD_INTERFACE` = 7, `VIR_NETWORK_SECTION_FORWARD_PF` = 8, `VIR_NETWORK_SECTION_PORTGROUP` = 9, `VIR_NETWORK_SECTION_DNS_HOST` = 10, `VIR_NETWORK_SECTION_DNS_TXT` = 11, `VIR_NETWORK_SECTION_DNS_SRV` = 12, **POL New** `VIR_NETWORK_SECTION_TUNNEL` = 13 }
- enum `virNetworkUpdateFlags` { `VIR_NETWORK_UPDATE_AFFECT_CURRENT` = 0, `VIR_NETWORK_UPDATE_AFFECT_LIVE` = 1 << 0, `VIR_NETWORK_UPDATE_AFFECT_CONFIG` = 1 << 1 }
- enum `virConnectListAllInterfacesFlags` { `VIR_CONNECT_LIST_INTERFACES_INACTIVE` = 1 << 0, `VIR_CONNECT_LIST_INTERFACES_ACTIVE` = 1 << 1 }
- enum `virInterfaceXMLFlags` { `VIR_INTERFACE_XML_INACTIVE` = 1 << 0 }
- enum `virStoragePoolState` { `VIR_STORAGE_POOL_INACTIVE` = 0, `VIR_STORAGE_POOL_BUILDING` = 1, `VIR_STORAGE_POOL_RUNNING` = 2, `VIR_STORAGE_POOL_DEGRADED` = 3, `VIR_STORAGE_POOL_INACCESSIBLE` = 4 }
- enum `virStoragePoolBuildFlags` { `VIR_STORAGE_POOL_BUILD_NEW` = 0, `VIR_STORAGE_POOL_BUILD_REPAIR` = (1 << 0), `VIR_STORAGE_POOL_BUILD_RESIZE` = (1 << 1), `VIR_STORAGE_POOL_BUILD_NO_OVERWRITE` = (1 << 2), `VIR_STORAGE_POOL_BUILD_OVERWRITE` = (1 << 3) }
- enum `virStoragePoolDeleteFlags` { `VIR_STORAGE_POOL_DELETE_NORMAL` = 0, `VIR_STORAGE_POOL_DELETE_ZEROED` = 1 << 0 }
- enum `virStorageVolType` { `VIR_STORAGE_VOL_FILE` = 0, `VIR_STORAGE_VOL_BLOCK` = 1, `VIR_STORAGE_VOL_DIR` = 2, `VIR_STORAGE_VOL_NETWORK` = 3 }
- enum `virStorageVolDeleteFlags` { `VIR_STORAGE_VOL_DELETE_NORMAL` = 0, `VIR_STORAGE_VOL_DELETE_ZEROED` = 1 << 0 }
- enum `virStorageVolWipeAlgorithm` { `VIR_STORAGE_VOL_WIPE_ALG_ZERO` = 0, `VIR_STORAGE_VOL_WIPE_ALG_NNSA` = 1, `VIR_STORAGE_VOL_WIPE_ALG_DOD` = 2, `VIR_STORAGE_VOL_WIPE_ALG_BSI` = 3, `VIR_STORAGE_VOL_WIPE_ALG_GUTMANN` = 4, `VIR_STORAGE_VOL_WIPE_ALG_SCHNEIER` = 5, `VIR_STORAGE_VOL_WIPE_ALG_PFITZNER7` = 6, `VIR_STORAGE_VOL_WIPE_ALG_PFITZNER33` = 7, `VIR_STORAGE_VOL_WIPE_ALG_RANDOM` = 8 }

- enum `virStorageXMLFlags` { `VIR_STORAGE_XML_INACTIVE` = (1 << 0) }
- enum `virConnectListAllStoragePoolsFlags` {  
`VIR_CONNECT_LIST_STORAGE_POOLS_INACTIVE` = 1 << 0, `VIR_CONNECT_LIST_STORAGE_POOLS_ACTIVE` = 1 << 1, `VIR_CONNECT_LIST_STORAGE_POOLS_PERSISTENT` = 1 << 2, `VIR_CONNECT_LIST_STORAGE_POOLS_TRANSIENT` = 1 << 3,  
`VIR_CONNECT_LIST_STORAGE_POOLS_AUTOSTART` = 1 << 4, `VIR_CONNECT_LIST_STORAGE_POOLS_NO_AUTOSTART` = 1 << 5, `VIR_CONNECT_LIST_STORAGE_POOLS_DIR` = 1 << 6, `VIR_CONNECT_LIST_STORAGE_POOLS_FS` = 1 << 7,  
`VIR_CONNECT_LIST_STORAGE_POOLS_NETFS` = 1 << 8, `VIR_CONNECT_LIST_STORAGE_POOLS_LOGICAL` = 1 << 9, `VIR_CONNECT_LIST_STORAGE_POOLS_DISK` = 1 << 10, `VIR_CONNECT_LIST_STORAGE_POOLS_ISCSI` = 1 << 11,  
`VIR_CONNECT_LIST_STORAGE_POOLS_SCSI` = 1 << 12, `VIR_CONNECT_LIST_STORAGE_POOLS_MPATH` = 1 << 13, `VIR_CONNECT_LIST_STORAGE_POOLS_RBD` = 1 << 14, `VIR_CONNECT_LIST_STORAGE_POOLS_SHEEPDOG` = 1 << 15 }
- enum `virStorageVolResizeFlags` { `VIR_STORAGE_VOL_RESIZE_ALLOCATE` = 1 << 0, `VIR_STORAGE_VOL_RESIZE_DELTA` = 1 << 1, `VIR_STORAGE_VOL_RESIZE_SHRINK` = 1 << 2 }
- enum `virKeycodeSet` {  
`VIR_KEYCODE_SET_LINUX` = 0, `VIR_KEYCODE_SET_XT` = 1, `VIR_KEYCODE_SET_ATSET1` = 2, `VIR_KEYCODE_SET_ATSET2` = 3,  
`VIR_KEYCODE_SET_ATSET3` = 4, `VIR_KEYCODE_SET_OSX` = 5, `VIR_KEYCODE_SET_XT_KBD` = 6,  
`VIR_KEYCODE_SET_USB` = 7,  
`VIR_KEYCODE_SET_WIN32` = 8, `VIR_KEYCODE_SET_RFB` = 9 }
- enum `virConnectListAllNodeDeviceFlags` {  
`VIR_CONNECT_LIST_NODE_DEVICES_CAP_SYSTEM` = 1 << 0, `VIR_CONNECT_LIST_NODE_DEVICES_CAP_PCI_DEV` = 1 << 1, `VIR_CONNECT_LIST_NODE_DEVICES_CAP_USB_DEV` = 1 << 2, `VIR_CONNECT_LIST_NODE_DEVICES_CAP_USB_INTERFACE` = 1 << 3,  
`VIR_CONNECT_LIST_NODE_DEVICES_CAP_NET` = 1 << 4, `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI_HOST` = 1 << 5, `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI_TARGET` = 1 << 6,  
`VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI` = 1 << 7,  
`VIR_CONNECT_LIST_NODE_DEVICES_CAP_STORAGE` = 1 << 8 }
- enum `virDomainEventType` {  
`VIR_DOMAIN_EVENT_DEFINED` = 0, `VIR_DOMAIN_EVENT_UNDEFINED` = 1, `VIR_DOMAIN_EVENT_STARTED` = 2, `VIR_DOMAIN_EVENT_SUSPENDED` = 3,  
`VIR_DOMAIN_EVENT_RESUMED` = 4, `VIR_DOMAIN_EVENT_STOPPED` = 5, `VIR_DOMAIN_EVENT_SHUTDOWN` = 6, `VIR_DOMAIN_EVENT_PMSUSPENDED` = 7 }
- enum `virDomainEventDefinedDetailType` { `VIR_DOMAIN_EVENT_DEFINED_ADDED` = 0, `VIR_DOMAIN_EVENT_DEFINED_UPDATED` = 1 }
- enum `virDomainEventUndefinedDetailType` { `VIR_DOMAIN_EVENT_UNDEFINED_REMOVED` = 0 }
- enum `virDomainEventStartedDetailType` {  
`VIR_DOMAIN_EVENT_STARTED_BOOTED` = 0, `VIR_DOMAIN_EVENT_STARTED_MIGRATED` = 1, `VIR_DOMAIN_EVENT_STARTED_RESTORED` = 2, `VIR_DOMAIN_EVENT_STARTED_FROM_SNAPSHOT` = 3,  
`VIR_DOMAIN_EVENT_STARTED_WAKEUP` = 4 }
- enum `virDomainEventSuspendedDetailType` {  
`VIR_DOMAIN_EVENT_SUSPENDED_PAUSED` = 0, `VIR_DOMAIN_EVENT_SUSPENDED_MIGRATED` = 1, `VIR_DOMAIN_EVENT_SUSPENDED_IOERROR` = 2, `VIR_DOMAIN_EVENT_SUSPENDED_WATCHDOG` = 3,  
`VIR_DOMAIN_EVENT_SUSPENDED_RESTORED` = 4, `VIR_DOMAIN_EVENT_SUSPENDED_FROM_SNAPSHOT` = 5 }
- enum `virDomainEventResumedDetailType` { `VIR_DOMAIN_EVENT_RESUMED_UNPAUSED` = 0, `VIR_DOMAIN_EVENT_RESUMED_MIGRATED` = 1, `VIR_DOMAIN_EVENT_RESUMED_FROM_SNAPSHOT` = 2 }
- enum `virDomainEventStoppedDetailType` {  
`VIR_DOMAIN_EVENT_STOPPED_SHUTDOWN` = 0, `VIR_DOMAIN_EVENT_STOPPED_DESTROYED` = 1, `VIR_DOMAIN_EVENT_STOPPED_CRASHED` = 2, `VIR_DOMAIN_EVENT_STOPPED_MIGRATED` = 3,  
`VIR_DOMAIN_EVENT_STOPPED_SAVED` = 4, `VIR_DOMAIN_EVENT_STOPPED_FAILED` = 5, `VIR_DOMAIN_EVENT_STOPPED_FROM_SNAPSHOT` = 6 }
- enum `virDomainEventShutdownDetailType` { `VIR_DOMAIN_EVENT_SHUTDOWN_FINISHED` = 0 }

- enum `virDomainEventPMSuspendedDetailType` { `VIR_DOMAIN_EVENT_PMSUSPENDED_MEMORY` = 0 }
- enum `virEventHandleType` { `VIR_EVENT_HANDLE_READABLE` = (1 << 0), `VIR_EVENT_HANDLE_WRITABLE` = (1 << 1), `VIR_EVENT_HANDLE_ERROR` = (1 << 2), `VIR_EVENT_HANDLE_HANGUP` = (1 << 3) }
- enum `virSecretUsageType` { `VIR_SECRET_USAGE_TYPE_NONE` = 0, `VIR_SECRET_USAGE_TYPE_VOLUME` = 1, `VIR_SECRET_USAGE_TYPE_CEPH` = 2 }
- enum `virConnectListAllSecretsFlags` { `VIR_CONNECT_LIST_SECRETS_EPHEMERAL` = 1 << 0, `VIR_CONNECT_LIST_SECRETS_NO_EPHEMERAL` = 1 << 1, `VIR_CONNECT_LIST_SECRETS_PRIVATE` = 1 << 2, `VIR_CONNECT_LIST_SECRETS_NO_PRIVATE` = 1 << 3 }
- enum `virStreamFlags` { `VIR_STREAM_NONBLOCK` = (1 << 0) }
- enum `virStreamEventType` { `VIR_STREAM_EVENT_READABLE` = (1 << 0), `VIR_STREAM_EVENT_WRITABLE` = (1 << 1), `VIR_STREAM_EVENT_ERROR` = (1 << 2), `VIR_STREAM_EVENT_HANGUP` = (1 << 3) }
- enum `virCPUCompareResult` { `VIR_CPU_COMPARE_ERROR` = -1, `VIR_CPU_COMPARE_INCOMPATIBLE` = 0, `VIR_CPU_COMPARE_IDENTICAL` = 1, `VIR_CPU_COMPARE_SUPERSET` = 2 }
- enum `virDomainJobType` { `VIR_DOMAIN_JOB_NONE` = 0, `VIR_DOMAIN_JOB_BOUNDED` = 1, `VIR_DOMAIN_JOB_UNBOUNDED` = 2, `VIR_DOMAIN_JOB_COMPLETED` = 3, `VIR_DOMAIN_JOB_FAILED` = 4, `VIR_DOMAIN_JOB_CANCELLED` = 5 }
- enum `virDomainSnapshotCreateFlags` { `VIR_DOMAIN_SNAPSHOT_CREATE_REDEFINE` = (1 << 0), `VIR_DOMAIN_SNAPSHOT_CREATE_CURRENT` = (1 << 1), `VIR_DOMAIN_SNAPSHOT_CREATE_NO_METADATA` = (1 << 2), `VIR_DOMAIN_SNAPSHOT_CREATE_HALT` = (1 << 3), `VIR_DOMAIN_SNAPSHOT_CREATE_DISK_ONLY` = (1 << 4), `VIR_DOMAIN_SNAPSHOT_CREATE_REUSE_EXT` = (1 << 5), `VIR_DOMAIN_SNAPSHOT_CREATE_QUIESCE` = (1 << 6), `VIR_DOMAIN_SNAPSHOT_CREATE_ATOMIC` = (1 << 7) }
- enum `virDomainSnapshotListFlags` { `VIR_DOMAIN_SNAPSHOT_LIST_ROOTS` = (1 << 0), `VIR_DOMAIN_SNAPSHOT_LIST_DESCENDANTS` = (1 << 0), `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` = (1 << 2), `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES` = (1 << 3), `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` = (1 << 1), `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA` = (1 << 4) }
- enum `virDomainSnapshotRevertFlags` { `VIR_DOMAIN_SNAPSHOT_REVERT_RUNNING` = 1 << 0, `VIR_DOMAIN_SNAPSHOT_REVERT_PAUSED` = 1 << 1, `VIR_DOMAIN_SNAPSHOT_REVERT_FORCE` = 1 << 2 }
- enum `virDomainSnapshotDeleteFlags` { `VIR_DOMAIN_SNAPSHOT_DELETE_CHILDREN` = (1 << 0), `VIR_DOMAIN_SNAPSHOT_DELETE_METADATA_ONLY` = (1 << 1), `VIR_DOMAIN_SNAPSHOT_DELETE_CHILDREN_ONLY` = (1 << 2) }
- enum `virDomainEventWatchdogAction` { `VIR_DOMAIN_EVENT_WATCHDOG_NONE` = 0, `VIR_DOMAIN_EVENT_WATCHDOG_PAUSE`, `VIR_DOMAIN_EVENT_WATCHDOG_RESET`, `VIR_DOMAIN_EVENT_WATCHDOG_POWEROFF`, `VIR_DOMAIN_EVENT_WATCHDOG_SHUTDOWN`, `VIR_DOMAIN_EVENT_WATCHDOG_DEBUG` }
- enum `virDomainEventIOErrorAction` { `VIR_DOMAIN_EVENT_IO_ERROR_NONE` = 0, `VIR_DOMAIN_EVENT_IO_ERROR_PAUSE`, `VIR_DOMAIN_EVENT_IO_ERROR_REPORT` }
- enum `virDomainEventGraphicsPhase` { `VIR_DOMAIN_EVENT_GRAPHICS_CONNECT` = 0, `VIR_DOMAIN_EVENT_GRAPHICS_INITIALIZE`, `VIR_DOMAIN_EVENT_GRAPHICS_DISCONNECT` }
- enum `virDomainEventGraphicsAddressType` { `VIR_DOMAIN_EVENT_GRAPHICS_ADDRESS_IPV4`, `VIR_DOMAIN_EVENT_GRAPHICS_ADDRESS_IPV6`, `VIR_DOMAIN_EVENT_GRAPHICS_ADDRESS_UNIX` }
- enum `virConnectDomainEventBlockJobStatus` { `VIR_DOMAIN_BLOCK_JOB_COMPLETED` = 0, `VIR_DOMAIN_BLOCK_JOB_FAILED` = 1, `VIR_DOMAIN_BLOCK_JOB_CANCELED` = 2 }
- enum `virConnectDomainEventDiskChangeReason` { `VIR_DOMAIN_EVENT_DISK_CHANGE_MISSING_ON_START` = 0 }
- enum `virDomainEventTrayChangeReason` { `VIR_DOMAIN_EVENT_TRAY_CHANGE_OPEN` = 0, `VIR_DOMAIN_EVENT_TRAY_CHANGE_CLOSE` }



- enum [virDomainEventID](#) {  
[VIR\\_DOMAIN\\_EVENT\\_ID\\_LIFECYCLE](#) = 0, [VIR\\_DOMAIN\\_EVENT\\_ID\\_REBOOT](#) = 1, [VIR\\_DOMAIN\\_EVENT\\_ID\\_RTC\\_CHANGE](#) = 2, [VIR\\_DOMAIN\\_EVENT\\_ID\\_WATCHDOG](#) = 3,  
[VIR\\_DOMAIN\\_EVENT\\_ID\\_IO\\_ERROR](#) = 4, [VIR\\_DOMAIN\\_EVENT\\_ID\\_GRAPHICS](#) = 5, [VIR\\_DOMAIN\\_EVENT\\_ID\\_IO\\_ERROR\\_REASON](#) = 6, [VIR\\_DOMAIN\\_EVENT\\_ID\\_CONTROL\\_ERROR](#) = 7,  
[VIR\\_DOMAIN\\_EVENT\\_ID\\_BLOCK\\_JOB](#) = 8, [VIR\\_DOMAIN\\_EVENT\\_ID\\_DISK\\_CHANGE](#) = 9, [VIR\\_DOMAIN\\_EVENT\\_ID\\_TRAY\\_CHANGE](#) = 10, [VIR\\_DOMAIN\\_EVENT\\_ID\\_PMWAKEUP](#) = 11,  
[VIR\\_DOMAIN\\_EVENT\\_ID\\_PMSUSPEND](#) = 12, [VIR\\_DOMAIN\\_EVENT\\_ID\\_BALLOON\\_CHANGE](#) = 13 }
- enum [virDomainConsoleFlags](#) { [VIR\\_DOMAIN\\_CONSOLE\\_FORCE](#) = (1 << 0), [VIR\\_DOMAIN\\_CONSOLE\\_SAFE](#) = (1 << 1) }
- enum [virDomainOpenGraphicsFlags](#) { [VIR\\_DOMAIN\\_OPEN\\_GRAPHICS\\_SKIPAUTH](#) = (1 << 0) }
- enum [virSchedParameterType](#) {  
[VIR\\_DOMAIN\\_SCHED\\_FIELD\\_INT](#) = [VIR\\_TYPED\\_PARAM\\_INT](#), [VIR\\_DOMAIN\\_SCHED\\_FIELD\\_UINT](#) = [VIR\\_TYPED\\_PARAM\\_UINT](#), [VIR\\_DOMAIN\\_SCHED\\_FIELD\\_LLONG](#) = [VIR\\_TYPED\\_PARAM\\_LLONG](#), [VIR\\_DOMAIN\\_SCHED\\_FIELD\\_ULLONG](#) = [VIR\\_TYPED\\_PARAM\\_ULLONG](#),  
[VIR\\_DOMAIN\\_SCHED\\_FIELD\\_DOUBLE](#) = [VIR\\_TYPED\\_PARAM\\_DOUBLE](#), [VIR\\_DOMAIN\\_SCHED\\_FIELD\\_BOOLEAN](#) = [VIR\\_TYPED\\_PARAM\\_BOOLEAN](#) }
- enum [virBlkioParameterType](#) {  
[VIR\\_DOMAIN\\_BLKIO\\_PARAM\\_INT](#) = [VIR\\_TYPED\\_PARAM\\_INT](#), [VIR\\_DOMAIN\\_BLKIO\\_PARAM\\_UINT](#) = [VIR\\_TYPED\\_PARAM\\_UINT](#), [VIR\\_DOMAIN\\_BLKIO\\_PARAM\\_LLONG](#) = [VIR\\_TYPED\\_PARAM\\_LLONG](#), [VIR\\_DOMAIN\\_BLKIO\\_PARAM\\_ULLONG](#) = [VIR\\_TYPED\\_PARAM\\_ULLONG](#),  
[VIR\\_DOMAIN\\_BLKIO\\_PARAM\\_DOUBLE](#) = [VIR\\_TYPED\\_PARAM\\_DOUBLE](#), [VIR\\_DOMAIN\\_BLKIO\\_PARAM\\_BOOLEAN](#) = [VIR\\_TYPED\\_PARAM\\_BOOLEAN](#) }
- enum [virMemoryParameterType](#) {  
[VIR\\_DOMAIN\\_MEMORY\\_PARAM\\_INT](#) = [VIR\\_TYPED\\_PARAM\\_INT](#), [VIR\\_DOMAIN\\_MEMORY\\_PARAM\\_UINT](#) = [VIR\\_TYPED\\_PARAM\\_UINT](#), [VIR\\_DOMAIN\\_MEMORY\\_PARAM\\_LLONG](#) = [VIR\\_TYPED\\_PARAM\\_LLONG](#), [VIR\\_DOMAIN\\_MEMORY\\_PARAM\\_ULLONG](#) = [VIR\\_TYPED\\_PARAM\\_ULLONG](#),  
[VIR\\_DOMAIN\\_MEMORY\\_PARAM\\_DOUBLE](#) = [VIR\\_TYPED\\_PARAM\\_DOUBLE](#), [VIR\\_DOMAIN\\_MEMORY\\_PARAM\\_BOOLEAN](#) = [VIR\\_TYPED\\_PARAM\\_BOOLEAN](#) }

## Functions

- int [virDomainGetSchedulerParameters](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, int \*nparams)
- int [virDomainGetSchedulerParametersFlags](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- int [virDomainSetSchedulerParameters](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, int nparams)
- int [virDomainSetSchedulerParametersFlags](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- [virDomainPtr](#) [virDomainMigrate](#) ([virDomainPtr](#) domain, [virConnectPtr](#) dconn, unsigned long flags, const char \*dname, const char \*uri, unsigned long bandwidth)
- [virDomainPtr](#) [virDomainMigrate2](#) ([virDomainPtr](#) domain, [virConnectPtr](#) dconn, const char \*dxml, unsigned long flags, const char \*dname, const char \*uri, unsigned long bandwidth)
- int [virDomainMigrateToURI](#) ([virDomainPtr](#) domain, const char \*duri, unsigned long flags, const char \*dname, unsigned long bandwidth)
- int [virDomainMigrateToURI2](#) ([virDomainPtr](#) domain, const char \*dconnuri, const char \*miguri, const char \*dxml, unsigned long flags, const char \*dname, unsigned long bandwidth)
- int [virDomainMigrateSetMaxDowntime](#) ([virDomainPtr](#) domain, unsigned long long downtime, unsigned int flags)
- int [virDomainMigrateSetMaxSpeed](#) ([virDomainPtr](#) domain, unsigned long bandwidth, unsigned int flags)
- int [virDomainMigrateGetMaxSpeed](#) ([virDomainPtr](#) domain, unsigned long \*bandwidth, unsigned int flags)
- int [virGetVersion](#) (unsigned long \*libVer, const char \*type, unsigned long \*typeVer)
- int [virInitialize](#) (void)
- [virConnectPtr](#) [virConnectOpen](#) (const char \*name)
- [virConnectPtr](#) [virConnectOpenReadOnly](#) (const char \*name)
- [virConnectPtr](#) [virConnectOpenAuth](#) (const char \*name, [virConnectAuthPtr](#) auth, unsigned int flags)
- int [virConnectRef](#) ([virConnectPtr](#) conn)

- int [virConnectClose](#) ([virConnectPtr](#) conn)
- const char \* [virConnectGetType](#) ([virConnectPtr](#) conn)
- int [virConnectGetVersion](#) ([virConnectPtr](#) conn, unsigned long \*hvVer)
- int [virConnectGetLibVersion](#) ([virConnectPtr](#) conn, unsigned long \*libVer)
- char \* [virConnectGetHostname](#) ([virConnectPtr](#) conn)
- char \* [virConnectGetURI](#) ([virConnectPtr](#) conn)
- char \* [virConnectGetSysinfo](#) ([virConnectPtr](#) conn, unsigned int flags)
- int [virConnectSetKeepAlive](#) ([virConnectPtr](#) conn, int interval, unsigned int count)
- int [virConnectRegisterCloseCallback](#) ([virConnectPtr](#) conn, [virConnectCloseFunc](#) cb, void \*opaque, [virFreeCallback](#) freecb)
- int [virConnectUnregisterCloseCallback](#) ([virConnectPtr](#) conn, [virConnectCloseFunc](#) cb)
- int [virConnectGetMaxVcpus](#) ([virConnectPtr](#) conn, const char \*type)
- int [virNodeGetInfo](#) ([virConnectPtr](#) conn, [virNodeInfoPtr](#) info)
- char \* [virConnectGetCapabilities](#) ([virConnectPtr](#) conn)
- int [virNodeGetCPUStats](#) ([virConnectPtr](#) conn, int cpuNum, [virNodeCPUStatsPtr](#) params, int \*nparams, unsigned int flags)
- int [virNodeGetMemoryStats](#) ([virConnectPtr](#) conn, int cellNum, [virNodeMemoryStatsPtr](#) params, int \*nparams, unsigned int flags)
- unsigned long long [virNodeGetFreeMemory](#) ([virConnectPtr](#) conn)
- int [virNodeGetSecurityModel](#) ([virConnectPtr](#) conn, [virSecurityModelPtr](#) secmodel)
- int [virNodeSuspendForDuration](#) ([virConnectPtr](#) conn, unsigned int target, unsigned long long duration, unsigned int flags)
- int [virConnectListDomains](#) ([virConnectPtr](#) conn, int \*ids, int maxids)
- int [virConnectNumOfDomains](#) ([virConnectPtr](#) conn)
- [virConnectPtr](#) [virDomainGetConnect](#) ([virDomainPtr](#) domain)
- [virDomainPtr](#) [virDomainCreateXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- [virDomainPtr](#) [virDomainLookupByName](#) ([virConnectPtr](#) conn, const char \*name)
- [virDomainPtr](#) [virDomainLookupByID](#) ([virConnectPtr](#) conn, int id)
- [virDomainPtr](#) [virDomainLookupByUUID](#) ([virConnectPtr](#) conn, const unsigned char \*uuid)
- [virDomainPtr](#) [virDomainLookupByUUIDString](#) ([virConnectPtr](#) conn, const char \*uuid)
- int [virDomainShutdown](#) ([virDomainPtr](#) domain)
- int [virDomainShutdownFlags](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainReboot](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainReset](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainDestroy](#) ([virDomainPtr](#) domain)
- int [virDomainDestroyFlags](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainRef](#) ([virDomainPtr](#) domain)
- int [virDomainFree](#) ([virDomainPtr](#) domain)
- int [virDomainSuspend](#) ([virDomainPtr](#) domain)
- int [virDomainResume](#) ([virDomainPtr](#) domain)
- int [virDomainPMSuspendForDuration](#) ([virDomainPtr](#) domain, unsigned int target, unsigned long long duration, unsigned int flags)
- int [virDomainPMWakeUp](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainSave](#) ([virDomainPtr](#) domain, const char \*to)
- int [virDomainSaveFlags](#) ([virDomainPtr](#) domain, const char \*to, const char \*dxml, unsigned int flags)
- int [virDomainRestore](#) ([virConnectPtr](#) conn, const char \*from)
- int [virDomainRestoreFlags](#) ([virConnectPtr](#) conn, const char \*from, const char \*dxml, unsigned int flags)
- char \* [virDomainSavImageGetXMLDesc](#) ([virConnectPtr](#) conn, const char \*file, unsigned int flags)
- int [virDomainSavImageDefineXML](#) ([virConnectPtr](#) conn, const char \*file, const char \*dxml, unsigned int flags)
- int [virDomainManagedSave](#) ([virDomainPtr](#) dom, unsigned int flags)
- int [virDomainHasManagedSavImage](#) ([virDomainPtr](#) dom, unsigned int flags)
- int [virDomainManagedSaveRemove](#) ([virDomainPtr](#) dom, unsigned int flags)
- int [virDomainCoreDump](#) ([virDomainPtr](#) domain, const char \*to, unsigned int flags)



- char \* [virDomainScreenshot](#) (virDomainPtr domain, virStreamPtr stream, unsigned int screen, unsigned int flags)
- int [virDomainGetInfo](#) (virDomainPtr domain, virDomainInfoPtr info)
- int [virDomainGetState](#) (virDomainPtr domain, int \*state, int \*reason, unsigned int flags)
- int [virDomainGetCPUStats](#) (virDomainPtr domain, virTypedParameterPtr params, unsigned int nparams, int start\_cpu, unsigned int ncpus, unsigned int flags)
- int [virDomainGetControllInfo](#) (virDomainPtr domain, virDomainControllInfoPtr info, unsigned int flags)
- char \* [virDomainGetSchedulerType](#) (virDomainPtr domain, int \*nparams)
- int [virDomainSetBlkioParameters](#) (virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)
- int [virDomainGetBlkioParameters](#) (virDomainPtr domain, virTypedParameterPtr params, int \*nparams, unsigned int flags)
- int [virDomainSetMemoryParameters](#) (virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)
- int [virDomainGetMemoryParameters](#) (virDomainPtr domain, virTypedParameterPtr params, int \*nparams, unsigned int flags)
- int [virDomainSetNumaParameters](#) (virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)
- int [virDomainGetNumaParameters](#) (virDomainPtr domain, virTypedParameterPtr params, int \*nparams, unsigned int flags)
- const char \* [virDomainGetName](#) (virDomainPtr domain)
- unsigned int [virDomainGetID](#) (virDomainPtr domain)
- int [virDomainGetUUID](#) (virDomainPtr domain, unsigned char \*uuid)
- int [virDomainGetUUIDString](#) (virDomainPtr domain, char \*buf)
- char \* [virDomainGetOSType](#) (virDomainPtr domain)
- unsigned long [virDomainGetMaxMemory](#) (virDomainPtr domain)
- int [virDomainSetMaxMemory](#) (virDomainPtr domain, unsigned long memory)
- int [virDomainSetMemory](#) (virDomainPtr domain, unsigned long memory)
- int [virDomainSetMemoryFlags](#) (virDomainPtr domain, unsigned long memory, unsigned int flags)
- int [virDomainGetMaxVcpus](#) (virDomainPtr domain)
- int [virDomainGetSecurityLabel](#) (virDomainPtr domain, virSecurityLabelPtr seclabel)
- char \* [virDomainGetHostname](#) (virDomainPtr domain, unsigned int flags)
- int [virDomainGetSecurityLabelList](#) (virDomainPtr domain, virSecurityLabelPtr \*seclabels)
- int [virDomainSetMetadata](#) (virDomainPtr domain, int type, const char \*metadata, const char \*key, const char \*uri, unsigned int flags)
- char \* [virDomainGetMetadata](#) (virDomainPtr domain, int type, const char \*uri, unsigned int flags)
- char \* [virDomainGetXMLDesc](#) (virDomainPtr domain, unsigned int flags)
- char \* [virConnectDomainXMLFromNative](#) (virConnectPtr conn, const char \*nativeFormat, const char \*nativeConfig, unsigned int flags)
- char \* [virConnectDomainXMLToNative](#) (virConnectPtr conn, const char \*nativeFormat, const char \*domainXml, unsigned int flags)
- int [virDomainBlockStats](#) (virDomainPtr dom, const char \*disk, virDomainBlockStatsPtr stats, size\_t size)
- int [virDomainBlockStatsFlags](#) (virDomainPtr dom, const char \*disk, virTypedParameterPtr params, int \*nparams, unsigned int flags)
- int [virDomainInterfaceStats](#) (virDomainPtr dom, const char \*path, virDomainInterfaceStatsPtr stats, size\_t size)
- int [virDomainSetInterfaceParameters](#) (virDomainPtr dom, const char \*device, virTypedParameterPtr params, int nparams, unsigned int flags)
- int [virDomainGetInterfaceParameters](#) (virDomainPtr dom, const char \*device, virTypedParameterPtr params, int \*nparams, unsigned int flags)
- int [virDomainBlockPeek](#) (virDomainPtr dom, const char \*disk, unsigned long long offset, size\_t size, void \*buffer, unsigned int flags)
- int [virDomainBlockResize](#) (virDomainPtr dom, const char \*disk, unsigned long long size, unsigned int flags)
- int [virDomainGetBlockInfo](#) (virDomainPtr dom, const char \*disk, virDomainBlockInfoPtr info, unsigned int flags)

- int [virDomainMemoryStats](#) ([virDomainPtr](#) dom, [virDomainMemoryStatPtr](#) stats, unsigned int nr\_stats, unsigned int flags)
- int [virDomainMemoryPeek](#) ([virDomainPtr](#) dom, unsigned long long start, size\_t size, void \*buffer, unsigned int flags)
- [virDomainPtr](#) [virDomainDefineXML](#) ([virConnectPtr](#) conn, const char \*xml)
- int [virDomainUndefine](#) ([virDomainPtr](#) domain)
- int [virDomainUndefineFlags](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virConnectNumOfDefinedDomains](#) ([virConnectPtr](#) conn)
- int [virConnectListDefinedDomains](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)
- int [virConnectListAllDomains](#) ([virConnectPtr](#) conn, [virDomainPtr](#) \*\*domains, unsigned int flags)
- int [virDomainCreate](#) ([virDomainPtr](#) domain)
- int [virDomainCreateWithFlags](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainGetAutostart](#) ([virDomainPtr](#) domain, int \*autostart)
- int [virDomainSetAutostart](#) ([virDomainPtr](#) domain, int autostart)
- int [virDomainSetVcpus](#) ([virDomainPtr](#) domain, unsigned int nvcpus)
- int [virDomainSetVcpusFlags](#) ([virDomainPtr](#) domain, unsigned int nvcpus, unsigned int flags)
- int [virDomainGetVcpusFlags](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainPinVcpu](#) ([virDomainPtr](#) domain, unsigned int vcpu, unsigned char \*cpumap, int maplen)
- int [virDomainPinVcpuFlags](#) ([virDomainPtr](#) domain, unsigned int vcpu, unsigned char \*cpumap, int maplen, unsigned int flags)
- int [virDomainGetVcpuPinInfo](#) ([virDomainPtr](#) domain, int ncpumaps, unsigned char \*cpumaps, int maplen, unsigned int flags)
- int [virDomainPinEmulator](#) ([virDomainPtr](#) domain, unsigned char \*cpumap, int maplen, unsigned int flags)
- int [virDomainGetEmulatorPinInfo](#) ([virDomainPtr](#) domain, unsigned char \*cpumaps, int maplen, unsigned int flags)
- int [virDomainGetVcpus](#) ([virDomainPtr](#) domain, [virVcpuInfoPtr](#) info, int maxinfo, unsigned char \*cpumaps, int maplen)
- int [virDomainAttachDevice](#) ([virDomainPtr](#) domain, const char \*xml)
- int [virDomainDetachDevice](#) ([virDomainPtr](#) domain, const char \*xml)
- int [virDomainAttachDeviceFlags](#) ([virDomainPtr](#) domain, const char \*xml, unsigned int flags)
- int [virDomainDetachDeviceFlags](#) ([virDomainPtr](#) domain, const char \*xml, unsigned int flags)
- int [virDomainUpdateDeviceFlags](#) ([virDomainPtr](#) domain, const char \*xml, unsigned int flags)
- int [virDomainBlockJobAbort](#) ([virDomainPtr](#) dom, const char \*disk, unsigned int flags)
- int [virDomainGetBlockJobInfo](#) ([virDomainPtr](#) dom, const char \*disk, [virDomainBlockJobInfoPtr](#) info, unsigned int flags)
- int [virDomainBlockJobSetSpeed](#) ([virDomainPtr](#) dom, const char \*disk, unsigned long bandwidth, unsigned int flags)
- int [virDomainBlockPull](#) ([virDomainPtr](#) dom, const char \*disk, unsigned long bandwidth, unsigned int flags)
- int [virDomainBlockRebase](#) ([virDomainPtr](#) dom, const char \*disk, const char \*base, unsigned long bandwidth, unsigned int flags)
- int [virDomainBlockCommit](#) ([virDomainPtr](#) dom, const char \*disk, const char \*base, const char \*top, unsigned long bandwidth, unsigned int flags)
- int [virDomainSetBlockIoTune](#) ([virDomainPtr](#) dom, const char \*disk, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- int [virDomainGetBlockIoTune](#) ([virDomainPtr](#) dom, const char \*disk, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- int [virDomainGetDiskErrors](#) ([virDomainPtr](#) dom, [virDomainDiskErrorPtr](#) errors, unsigned int maxerrors, unsigned int flags)
- int [virNodeGetCellsFreeMemory](#) ([virConnectPtr](#) conn, unsigned long long \*freeMems, int startCell, int maxCells)
- [virConnectPtr](#) [virNetworkGetConnect](#) ([virNetworkPtr](#) network)
- int [virConnectNumOfNetworks](#) ([virConnectPtr](#) conn)
- int [virConnectListNetworks](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)
- int [virConnectNumOfDefinedNetworks](#) ([virConnectPtr](#) conn)
- int [virConnectListDefinedNetworks](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)

- [int virConnectListAllNetworks](#) ([virConnectPtr](#) conn, [virNetworkPtr](#) \*\*nets, unsigned int flags)
- [virNetworkPtr virNetworkLookupByName](#) ([virConnectPtr](#) conn, const char \*name)
- [virNetworkPtr virNetworkLookupByUUID](#) ([virConnectPtr](#) conn, const unsigned char \*uuid)
- [virNetworkPtr virNetworkLookupByUUIDString](#) ([virConnectPtr](#) conn, const char \*uuid)
- [virNetworkPtr virNetworkCreateXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc)
- [virNetworkPtr virNetworkDefineXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc)
- [int virNetworkUndefine](#) ([virNetworkPtr](#) network)
- [int virNetworkUpdate](#) ([virNetworkPtr](#) network, unsigned int command, unsigned int section, int parentIndex, const char \*xml, unsigned int flags)
- [int virNetworkCreate](#) ([virNetworkPtr](#) network)
- [int virNetworkDestroy](#) ([virNetworkPtr](#) network)
- [int virNetworkRef](#) ([virNetworkPtr](#) network)
- [int virNetworkFree](#) ([virNetworkPtr](#) network)
- const char \* [virNetworkGetName](#) ([virNetworkPtr](#) network)
- [int virNetworkGetUUID](#) ([virNetworkPtr](#) network, unsigned char \*uuid)
- [int virNetworkGetUUIDString](#) ([virNetworkPtr](#) network, char \*buf)
- char \* [virNetworkGetXMLDesc](#) ([virNetworkPtr](#) network, unsigned int flags)
- char \* [virNetworkGetBridgeName](#) ([virNetworkPtr](#) network)
- **POL New** char \* [virNetworkGetBridgeType](#) ([virNetworkPtr](#) network)
- [int virNetworkGetAutostart](#) ([virNetworkPtr](#) network, int \*autostart)
- [int virNetworkSetAutostart](#) ([virNetworkPtr](#) network, int autostart)
- [virConnectPtr virInterfaceGetConnect](#) ([virInterfacePtr](#) iface)
- [int virConnectNumOfInterfaces](#) ([virConnectPtr](#) conn)
- [int virConnectListInterfaces](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)
- [int virConnectNumOfDefinedInterfaces](#) ([virConnectPtr](#) conn)
- [int virConnectListDefinedInterfaces](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)
- [int virConnectListAllInterfaces](#) ([virConnectPtr](#) conn, [virInterfacePtr](#) \*\*ifaces, unsigned int flags)
- [virInterfacePtr virInterfaceLookupByName](#) ([virConnectPtr](#) conn, const char \*name)
- [virInterfacePtr virInterfaceLookupByMACString](#) ([virConnectPtr](#) conn, const char \*mac)
- const char \* [virInterfaceGetName](#) ([virInterfacePtr](#) iface)
- const char \* [virInterfaceGetMACString](#) ([virInterfacePtr](#) iface)
- char \* [virInterfaceGetXMLDesc](#) ([virInterfacePtr](#) iface, unsigned int flags)
- [virInterfacePtr virInterfaceDefineXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- [int virInterfaceUndefine](#) ([virInterfacePtr](#) iface)
- [int virInterfaceCreate](#) ([virInterfacePtr](#) iface, unsigned int flags)
- [int virInterfaceDestroy](#) ([virInterfacePtr](#) iface, unsigned int flags)
- [int virInterfaceRef](#) ([virInterfacePtr](#) iface)
- [int virInterfaceFree](#) ([virInterfacePtr](#) iface)
- [int virInterfaceChangeBegin](#) ([virConnectPtr](#) conn, unsigned int flags)
- [int virInterfaceChangeCommit](#) ([virConnectPtr](#) conn, unsigned int flags)
- [int virInterfaceChangeRollback](#) ([virConnectPtr](#) conn, unsigned int flags)
- [virConnectPtr virStoragePoolGetConnect](#) ([virStoragePoolPtr](#) pool)
- [int virConnectNumOfStoragePools](#) ([virConnectPtr](#) conn)
- [int virConnectListStoragePools](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)
- [int virConnectNumOfDefinedStoragePools](#) ([virConnectPtr](#) conn)
- [int virConnectListDefinedStoragePools](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)
- [int virConnectListAllStoragePools](#) ([virConnectPtr](#) conn, [virStoragePoolPtr](#) \*\*pools, unsigned int flags)
- char \* [virConnectFindStoragePoolSources](#) ([virConnectPtr](#) conn, const char \*type, const char \*srcSpec, unsigned int flags)
- [virStoragePoolPtr virStoragePoolLookupByName](#) ([virConnectPtr](#) conn, const char \*name)
- [virStoragePoolPtr virStoragePoolLookupByUUID](#) ([virConnectPtr](#) conn, const unsigned char \*uuid)
- [virStoragePoolPtr virStoragePoolLookupByUUIDString](#) ([virConnectPtr](#) conn, const char \*uuid)
- [virStoragePoolPtr virStoragePoolLookupByVolume](#) ([virStorageVolPtr](#) vol)
- [virStoragePoolPtr virStoragePoolCreateXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- [virStoragePoolPtr virStoragePoolDefineXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)

- int [virStoragePoolBuild](#) ([virStoragePoolPtr](#) pool, unsigned int flags)
- int [virStoragePoolUndefine](#) ([virStoragePoolPtr](#) pool)
- int [virStoragePoolCreate](#) ([virStoragePoolPtr](#) pool, unsigned int flags)
- int [virStoragePoolDestroy](#) ([virStoragePoolPtr](#) pool)
- int [virStoragePoolDelete](#) ([virStoragePoolPtr](#) pool, unsigned int flags)
- int [virStoragePoolRef](#) ([virStoragePoolPtr](#) pool)
- int [virStoragePoolFree](#) ([virStoragePoolPtr](#) pool)
- int [virStoragePoolRefresh](#) ([virStoragePoolPtr](#) pool, unsigned int flags)
- const char \* [virStoragePoolGetName](#) ([virStoragePoolPtr](#) pool)
- int [virStoragePoolGetUUID](#) ([virStoragePoolPtr](#) pool, unsigned char \*uuid)
- int [virStoragePoolGetUUIDString](#) ([virStoragePoolPtr](#) pool, char \*buf)
- int [virStoragePoolGetInfo](#) ([virStoragePoolPtr](#) vol, [virStoragePoolInfoPtr](#) info)
- char \* [virStoragePoolGetXMLDesc](#) ([virStoragePoolPtr](#) pool, unsigned int flags)
- int [virStoragePoolGetAutostart](#) ([virStoragePoolPtr](#) pool, int \*autostart)
- int [virStoragePoolSetAutostart](#) ([virStoragePoolPtr](#) pool, int autostart)
- int [virStoragePoolNumOfVolumes](#) ([virStoragePoolPtr](#) pool)
- int [virStoragePoolListVolumes](#) ([virStoragePoolPtr](#) pool, char \*\*const names, int maxnames)
- int [virStoragePoolListAllVolumes](#) ([virStoragePoolPtr](#) pool, [virStorageVolPtr](#) \*\*vols, unsigned int flags)
- [virConnectPtr](#) [virStorageVolGetConnect](#) ([virStorageVolPtr](#) vol)
- [virStorageVolPtr](#) [virStorageVolLookupByName](#) ([virStoragePoolPtr](#) pool, const char \*name)
- [virStorageVolPtr](#) [virStorageVolLookupByKey](#) ([virConnectPtr](#) conn, const char \*key)
- [virStorageVolPtr](#) [virStorageVolLookupByPath](#) ([virConnectPtr](#) conn, const char \*path)
- const char \* [virStorageVolGetName](#) ([virStorageVolPtr](#) vol)
- const char \* [virStorageVolGetKey](#) ([virStorageVolPtr](#) vol)
- [virStorageVolPtr](#) [virStorageVolCreateXML](#) ([virStoragePoolPtr](#) pool, const char \*xmldesc, unsigned int flags)
- [virStorageVolPtr](#) [virStorageVolCreateXMLFrom](#) ([virStoragePoolPtr](#) pool, const char \*xmldesc, [virStorageVolPtr](#) clonevol, unsigned int flags)
- int [virStorageVolDownload](#) ([virStorageVolPtr](#) vol, [virStreamPtr](#) stream, unsigned long long offset, unsigned long long length, unsigned int flags)
- int [virStorageVolUpload](#) ([virStorageVolPtr](#) vol, [virStreamPtr](#) stream, unsigned long long offset, unsigned long long length, unsigned int flags)
- int [virStorageVolDelete](#) ([virStorageVolPtr](#) vol, unsigned int flags)
- int [virStorageVolWipe](#) ([virStorageVolPtr](#) vol, unsigned int flags)
- int [virStorageVolWipePattern](#) ([virStorageVolPtr](#) vol, unsigned int algorithm, unsigned int flags)
- int [virStorageVolRef](#) ([virStorageVolPtr](#) vol)
- int [virStorageVolFree](#) ([virStorageVolPtr](#) vol)
- int [virStorageVolGetInfo](#) ([virStorageVolPtr](#) vol, [virStorageVolInfoPtr](#) info)
- char \* [virStorageVolGetXMLDesc](#) ([virStorageVolPtr](#) pool, unsigned int flags)
- char \* [virStorageVolGetPath](#) ([virStorageVolPtr](#) vol)
- int [virStorageVolResize](#) ([virStorageVolPtr](#) vol, unsigned long long capacity, unsigned int flags)
- int [virDomainSendKey](#) ([virDomainPtr](#) domain, unsigned int codeset, unsigned int holdtime, unsigned int \*keycodes, int nkeycodes, unsigned int flags)
- [virDomainPtr](#) [virDomainCreateLinux](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- int [virNodeNumOfDevices](#) ([virConnectPtr](#) conn, const char \*cap, unsigned int flags)
- int [virNodeListDevices](#) ([virConnectPtr](#) conn, const char \*cap, char \*\*const names, int maxnames, unsigned int flags)
- int [virConnectListAllNodeDevices](#) ([virConnectPtr](#) conn, [virNodeDevicePtr](#) \*\*devices, unsigned int flags)
- [virNodeDevicePtr](#) [virNodeDeviceLookupByName](#) ([virConnectPtr](#) conn, const char \*name)
- const char \* [virNodeDeviceGetName](#) ([virNodeDevicePtr](#) dev)
- const char \* [virNodeDeviceGetParent](#) ([virNodeDevicePtr](#) dev)
- int [virNodeDeviceNumOfCaps](#) ([virNodeDevicePtr](#) dev)
- int [virNodeDeviceListCaps](#) ([virNodeDevicePtr](#) dev, char \*\*const names, int maxnames)
- char \* [virNodeDeviceGetXMLDesc](#) ([virNodeDevicePtr](#) dev, unsigned int flags)
- int [virNodeDeviceRef](#) ([virNodeDevicePtr](#) dev)
- int [virNodeDeviceFree](#) ([virNodeDevicePtr](#) dev)

- int [virNodeDeviceDetach](#) ([virNodeDevicePtr](#) dev)
- int [virNodeDeviceReAttach](#) ([virNodeDevicePtr](#) dev)
- int [virNodeDeviceReset](#) ([virNodeDevicePtr](#) dev)
- [virNodeDevicePtr](#) [virNodeDeviceCreateXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- int [virNodeDeviceDestroy](#) ([virNodeDevicePtr](#) dev)
- int [virConnectDomainEventRegister](#) ([virConnectPtr](#) conn, [virConnectDomainEventCallback](#) cb, void \*opaque, [virFreeCallback](#) freecb)
- int [virConnectDomainEventDeregister](#) ([virConnectPtr](#) conn, [virConnectDomainEventCallback](#) cb)
- void [virEventRegisterImpl](#) ([virEventAddHandleFunc](#) addHandle, [virEventUpdateHandleFunc](#) updateHandle, [virEventRemoveHandleFunc](#) removeHandle, [virEventAddTimeoutFunc](#) addTimeout, [virEventUpdateTimeoutFunc](#) updateTimeout, [virEventRemoveTimeoutFunc](#) removeTimeout)
- int [virEventRegisterDefaultImpl](#) (void)
- int [virEventRunDefaultImpl](#) (void)
- int [virEventAddHandle](#) (int fd, int events, [virEventHandleCallback](#) cb, void \*opaque, [virFreeCallback](#) ff)
- void [virEventUpdateHandle](#) (int watch, int events)
- int [virEventRemoveHandle](#) (int watch)
- int [virEventAddTimeout](#) (int frequency, [virEventTimeoutCallback](#) cb, void \*opaque, [virFreeCallback](#) ff)
- void [virEventUpdateTimeout](#) (int timer, int frequency)
- int [virEventRemoveTimeout](#) (int timer)
- [virConnectPtr](#) [virSecretGetConnect](#) ([virSecretPtr](#) secret)
- int [virConnectNumOfSecrets](#) ([virConnectPtr](#) conn)
- int [virConnectListSecrets](#) ([virConnectPtr](#) conn, char \*\*uuids, int maxuuids)
- int [virConnectListAllSecrets](#) ([virConnectPtr](#) conn, [virSecretPtr](#) \*\*secrets, unsigned int flags)
- [virSecretPtr](#) [virSecretLookupByUUID](#) ([virConnectPtr](#) conn, const unsigned char \*uuid)
- [virSecretPtr](#) [virSecretLookupByUUIDString](#) ([virConnectPtr](#) conn, const char \*uuid)
- [virSecretPtr](#) [virSecretLookupByUsage](#) ([virConnectPtr](#) conn, int usageType, const char \*usageID)
- [virSecretPtr](#) [virSecretDefineXML](#) ([virConnectPtr](#) conn, const char \*xml, unsigned int flags)
- int [virSecretGetUUID](#) ([virSecretPtr](#) secret, unsigned char \*buf)
- int [virSecretGetUUIDString](#) ([virSecretPtr](#) secret, char \*buf)
- int [virSecretGetUsageType](#) ([virSecretPtr](#) secret)
- const char \* [virSecretGetUsageID](#) ([virSecretPtr](#) secret)
- char \* [virSecretGetXMLDesc](#) ([virSecretPtr](#) secret, unsigned int flags)
- int [virSecretSetValue](#) ([virSecretPtr](#) secret, const unsigned char \*value, size\_t value\_size, unsigned int flags)
- unsigned char \* [virSecretGetValue](#) ([virSecretPtr](#) secret, size\_t \*value\_size, unsigned int flags)
- int [virSecretUndefine](#) ([virSecretPtr](#) secret)
- int [virSecretRef](#) ([virSecretPtr](#) secret)
- int [virSecretFree](#) ([virSecretPtr](#) secret)
- [virStreamPtr](#) [virStreamNew](#) ([virConnectPtr](#) conn, unsigned int flags)
- int [virStreamRef](#) ([virStreamPtr](#) st)
- int [virStreamSend](#) ([virStreamPtr](#) st, const char \*data, size\_t nbytes)
- int [virStreamRecv](#) ([virStreamPtr](#) st, char \*data, size\_t nbytes)
- int [virStreamSendAll](#) ([virStreamPtr](#) st, [virStreamSourceFunc](#) handler, void \*opaque)
- int [virStreamRecvAll](#) ([virStreamPtr](#) st, [virStreamSinkFunc](#) handler, void \*opaque)
- int [virStreamEventAddCallback](#) ([virStreamPtr](#) stream, int events, [virStreamEventCallback](#) cb, void \*opaque, [virFreeCallback](#) ff)
- int [virStreamEventUpdateCallback](#) ([virStreamPtr](#) stream, int events)
- int [virStreamEventRemoveCallback](#) ([virStreamPtr](#) stream)
- int [virStreamFinish](#) ([virStreamPtr](#) st)
- int [virStreamAbort](#) ([virStreamPtr](#) st)
- int [virStreamFree](#) ([virStreamPtr](#) st)
- int [virDomainsActive](#) ([virDomainPtr](#) dom)
- int [virDomainsPersistent](#) ([virDomainPtr](#) dom)
- int [virDomainsUpdated](#) ([virDomainPtr](#) dom)
- int [virNetworksActive](#) ([virNetworkPtr](#) net)
- int [virNetworksPersistent](#) ([virNetworkPtr](#) net)



- int [virStoragePoolsActive](#) ([virStoragePoolPtr](#) pool)
- int [virStoragePoolsPersistent](#) ([virStoragePoolPtr](#) pool)
- int [virInterfaceIsActive](#) ([virInterfacePtr](#) iface)
- int [virConnectIsEncrypted](#) ([virConnectPtr](#) conn)
- int [virConnectIsSecure](#) ([virConnectPtr](#) conn)
- int [virConnectIsAlive](#) ([virConnectPtr](#) conn)
- int [virConnectCompareCPU](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- char \* [virConnectBaselineCPU](#) ([virConnectPtr](#) conn, const char \*\*xmlCPUs, unsigned int ncpus, unsigned int flags)
- int [virDomainGetJobInfo](#) ([virDomainPtr](#) dom, [virDomainJobInfoPtr](#) info)
- int [virDomainAbortJob](#) ([virDomainPtr](#) dom)
- const char \* [virDomainSnapshotGetName](#) ([virDomainSnapshotPtr](#) snapshot)
- [virDomainPtr](#) [virDomainSnapshotGetDomain](#) ([virDomainSnapshotPtr](#) snapshot)
- [virConnectPtr](#) [virDomainSnapshotGetConnect](#) ([virDomainSnapshotPtr](#) snapshot)
- [virDomainSnapshotPtr](#) [virDomainSnapshotCreateXML](#) ([virDomainPtr](#) domain, const char \*xmlDesc, unsigned int flags)
- char \* [virDomainSnapshotGetXMLDesc](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotNum](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainSnapshotListNames](#) ([virDomainPtr](#) domain, char \*\*names, int nameslen, unsigned int flags)
- int [virDomainListAllSnapshots](#) ([virDomainPtr](#) domain, [virDomainSnapshotPtr](#) \*\*snaps, unsigned int flags)
- int [virDomainSnapshotNumChildren](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotListChildrenNames](#) ([virDomainSnapshotPtr](#) snapshot, char \*\*names, int nameslen, unsigned int flags)
- int [virDomainSnapshotListAllChildren](#) ([virDomainSnapshotPtr](#) snapshot, [virDomainSnapshotPtr](#) \*\*snaps, unsigned int flags)
- [virDomainSnapshotPtr](#) [virDomainSnapshotLookupByName](#) ([virDomainPtr](#) domain, const char \*name, unsigned int flags)
- int [virDomainHasCurrentSnapshot](#) ([virDomainPtr](#) domain, unsigned int flags)
- [virDomainSnapshotPtr](#) [virDomainSnapshotCurrent](#) ([virDomainPtr](#) domain, unsigned int flags)
- [virDomainSnapshotPtr](#) [virDomainSnapshotGetParent](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotIsCurrent](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotHasMetadata](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainRevertToSnapshot](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotDelete](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotRef](#) ([virDomainSnapshotPtr](#) snapshot)
- int [virDomainSnapshotFree](#) ([virDomainSnapshotPtr](#) snapshot)
- int [virConnectDomainEventRegisterAny](#) ([virConnectPtr](#) conn, [virDomainPtr](#) dom, int eventID, [virConnectDomainEventGenericCallback](#) cb, void \*opaque, [virFreeCallback](#) freecb)
- int [virConnectDomainEventDeregisterAny](#) ([virConnectPtr](#) conn, int callbackID)
- int [virConnectNumOfNWFilters](#) ([virConnectPtr](#) conn)
- int [virConnectListNWFilters](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)
- int [virConnectListAllNWFilters](#) ([virConnectPtr](#) conn, [virNWFilterPtr](#) \*\*filters, unsigned int flags)
- [virNWFilterPtr](#) [virNWFilterLookupByName](#) ([virConnectPtr](#) conn, const char \*name)
- [virNWFilterPtr](#) [virNWFilterLookupByUUID](#) ([virConnectPtr](#) conn, const unsigned char \*uuid)
- [virNWFilterPtr](#) [virNWFilterLookupByUUIDString](#) ([virConnectPtr](#) conn, const char \*uuid)
- [virNWFilterPtr](#) [virNWFilterDefineXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc)
- int [virNWFilterUndefine](#) ([virNWFilterPtr](#) nwfilter)
- int [virNWFilterRef](#) ([virNWFilterPtr](#) nwfilter)
- int [virNWFilterFree](#) ([virNWFilterPtr](#) nwfilter)
- const char \* [virNWFilterGetName](#) ([virNWFilterPtr](#) nwfilter)
- int [virNWFilterGetUUID](#) ([virNWFilterPtr](#) nwfilter, unsigned char \*uuid)
- int [virNWFilterGetUUIDString](#) ([virNWFilterPtr](#) nwfilter, char \*buf)
- char \* [virNWFilterGetXMLDesc](#) ([virNWFilterPtr](#) nwfilter, unsigned int flags)
- int [virDomainOpenConsole](#) ([virDomainPtr](#) dom, const char \*devname, [virStreamPtr](#) st, unsigned int flags)
- int [virDomainOpenGraphics](#) ([virDomainPtr](#) dom, unsigned int idx, int fd, unsigned int flags)

- int [virDomainInjectNMI](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virNodeGetMemoryParameters](#) ([virConnectPtr](#) conn, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- int [virNodeSetMemoryParameters](#) ([virConnectPtr](#) conn, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)

## Variables

- [VIR\\_EXPORT\\_VAR](#) [virConnectAuthPtr](#) [virConnectAuthPtrDefault](#)

### 4.1.1 Macro Definition Documentation

#### 4.1.1.1 #define \_virBlkioParameter \_virTypedParameter

[virBlkioParameter](#):

a [virBlkioParameter](#) is the set of blkio parameters. Provided for backwards compatibility; [virTypedParameter](#) is the preferred alias since 0.9.2.

#### 4.1.1.2 #define \_virMemoryParameter \_virTypedParameter

[virMemoryParameter](#):

a [virMemoryParameter](#) is the set of scheduler parameters. Provided for backwards compatibility; [virTypedParameter](#) is the preferred alias since 0.9.2.

#### 4.1.1.3 #define \_virSchedParameter \_virTypedParameter

[virSchedParameter](#):

a [virSchedParameter](#) is the set of scheduler parameters. Provided for backwards compatibility; [virTypedParameter](#) is the preferred alias since 0.9.2.

#### 4.1.1.4 #define LIBVIR\_VERSION\_NUMBER 10002

[LIBVIR\\_VERSION\\_NUMBER](#):

Macro providing the version of the library as version \* 1,000,000 + minor \* 1000 + micro

#### 4.1.1.5 #define VIR\_COPY\_CPUMAP( *cpumaps*, *maplen*, *vcpu*, *cpumap* ) memcpy(cpumap, &(cpumaps[(vcpu)\*(maplen)]), (maplen))

[VIR\\_COPY\\_CPUMAP](#): : pointer to an array of [cpumap](#) (in 8-bit bytes) (IN) : the length (in bytes) of one [cpumap](#) : the virtual CPU number : pointer to a [cpumap](#) (in 8-bit bytes) (OUT) This [cpumap](#) must be previously allocated by the caller (ie: malloc(maplen))

This macro is to be used in conjunction with [virDomainGetVcpus\(\)](#) and [virDomainPinVcpu\(\)](#) APIs. [VIR\\_COPY\\_CPUMAP](#) macro extract the [cpumap](#) of the specified [vcpu](#) from [cpumaps](#) array and copy it into [cpumap](#) to be used later by [virDomainPinVcpu\(\)](#) API.

#### 4.1.1.6 #define VIR\_CPU\_MAPLEN( *cpu* ) (((cpu)+7)/8)

[VIR\\_CPU\\_MAPLEN](#): : number of physical CPUs

This macro is to be used in conjunction with [virDomainPinVcpu\(\)](#) API. It returns the length (in bytes) required to store the complete CPU map between a single virtual & all physical CPUs of a domain.

4.1.1.7 **#define VIR\_CPU\_USABLE( *cpumaps, maplen, vcpu, cpu* )** (cpumaps[((vcpu)\*(maplen))+((cpu)/8)] & (1<<((cpu)%8)))

VIR\_CPU\_USABLE: : pointer to an array of cpumap (in 8-bit bytes) (IN) : the length (in bytes) of one cpumap : the virtual CPU number : the physical CPU number

This macro is to be used in conjunction with [virDomainGetVcpus\(\)](#) API. VIR\_CPU\_USABLE macro returns a non zero value (true) if the cpu is usable by the vcpu, and 0 otherwise.

4.1.1.8 **#define VIR\_DEPRECATED** /\* nothing \*/

4.1.1.9 **#define VIR\_DOMAIN\_BANDWIDTH\_IN\_AVERAGE** "inbound.average"

VIR\_DOMAIN\_BANDWIDTH\_IN\_AVERAGE:

Macro represents the inbound average of NIC bandwidth, as a uint.

4.1.1.10 **#define VIR\_DOMAIN\_BANDWIDTH\_IN\_BURST** "inbound.burst"

VIR\_DOMAIN\_BANDWIDTH\_IN\_BURST:

Macro represents the inbound burst of NIC bandwidth, as a uint.

4.1.1.11 **#define VIR\_DOMAIN\_BANDWIDTH\_IN\_PEAK** "inbound.peak"

VIR\_DOMAIN\_BANDWIDTH\_IN\_PEAK:

Macro represents the inbound peak of NIC bandwidth, as a uint.

4.1.1.12 **#define VIR\_DOMAIN\_BANDWIDTH\_OUT\_AVERAGE** "outbound.average"

VIR\_DOMAIN\_BANDWIDTH\_OUT\_AVERAGE:

Macro represents the outbound average of NIC bandwidth, as a uint.

4.1.1.13 **#define VIR\_DOMAIN\_BANDWIDTH\_OUT\_BURST** "outbound.burst"

VIR\_DOMAIN\_BANDWIDTH\_OUT\_BURST:

Macro represents the outbound burst of NIC bandwidth, as a uint.

4.1.1.14 **#define VIR\_DOMAIN\_BANDWIDTH\_OUT\_PEAK** "outbound.peak"

VIR\_DOMAIN\_BANDWIDTH\_OUT\_PEAK:

Macro represents the outbound peak of NIC bandwidth, as a uint.

4.1.1.15 **#define VIR\_DOMAIN\_BLKIO\_DEVICE\_WEIGHT** "device.weight"

VIR\_DOMAIN\_BLKIO\_DEVICE\_WEIGHT:

Macro for the blkio tunable weight\_device: it represents the per-device weight, as a string. The string is parsed as a series of /path/to/device,weight elements, separated by ','.



**4.1.1.16 #define VIR\_DOMAIN\_BLKIO\_FIELD\_LENGTH VIR\_TYPED\_PARAM\_FIELD\_LENGTH**

VIR\_DOMAIN\_BLKIO\_FIELD\_LENGTH:

Macro providing the field length of virBlkioParameter. Provided for backwards compatibility; VIR\_TYPED\_PARAM\_FIELD\_LENGTH is the preferred value since 0.9.2.

**4.1.1.17 #define VIR\_DOMAIN\_BLKIO\_WEIGHT "weight"**

VIR\_DOMAIN\_BLKIO\_WEIGHT:

Macro for the Blkio tunable weight: it represents the io weight the guest can use, as a uint.

**4.1.1.18 #define VIR\_DOMAIN\_BLOCK\_IOTUNE\_READ\_BYTES\_SEC "read\_bytes\_sec"**

VIR\_DOMAIN\_BLOCK\_IOTUNE\_READ\_BYTES\_SEC:

Macro for the BlockIoTune tunable weight: it represents the read bytes per second permitted through a block device, as a ullong.

**4.1.1.19 #define VIR\_DOMAIN\_BLOCK\_IOTUNE\_READ\_IOPS\_SEC "read\_iops\_sec"**

VIR\_DOMAIN\_BLOCK\_IOTUNE\_READ\_IOPS\_SEC:

Macro for the BlockIoTune tunable weight: it represents the read I/O operations per second permitted through a block device, as a ullong.

**4.1.1.20 #define VIR\_DOMAIN\_BLOCK\_IOTUNE\_TOTAL\_BYTES\_SEC "total\_bytes\_sec"**

VIR\_DOMAIN\_BLOCK\_IOTUNE\_TOTAL\_BYTES\_SEC:

Macro for the BlockIoTune tunable weight: it represents the total bytes per second permitted through a block device, as a ullong.

**4.1.1.21 #define VIR\_DOMAIN\_BLOCK\_IOTUNE\_TOTAL\_IOPS\_SEC "total\_iops\_sec"**

VIR\_DOMAIN\_BLOCK\_IOTUNE\_TOTAL\_IOPS\_SEC:

Macro for the BlockIoTune tunable weight: it represents the total I/O operations per second permitted through a block device, as a ullong.

**4.1.1.22 #define VIR\_DOMAIN\_BLOCK\_IOTUNE\_WRITE\_BYTES\_SEC "write\_bytes\_sec"**

VIR\_DOMAIN\_BLOCK\_IOTUNE\_WRITE\_BYTES\_SEC:

Macro for the BlockIoTune tunable weight: it represents the write bytes per second permitted through a block device, as a ullong.

**4.1.1.23 #define VIR\_DOMAIN\_BLOCK\_IOTUNE\_WRITE\_IOPS\_SEC "write\_iops\_sec"**

VIR\_DOMAIN\_BLOCK\_IOTUNE\_WRITE\_IOPS\_SEC: Macro for the BlockIoTune tunable weight: it represents the write I/O operations per second permitted through a block device, as a ullong.

**4.1.1.24 #define VIR\_DOMAIN\_BLOCK\_STATS\_ERRS "errs"**

VIR\_DOMAIN\_BLOCK\_STATS\_ERRS:

In Xen this returns the mysterious 'oo\_req', as an llong.

**4.1.1.25 #define VIR\_DOMAIN\_BLOCK\_STATS\_FIELD\_LENGTH VIR\_TYPED\_PARAM\_FIELD\_LENGTH**

VIR\_DOMAIN\_BLOCK\_STATS\_FIELD\_LENGTH:

Macro providing the field length of parameter names when using [virDomainBlockStatsFlags\(\)](#).

**4.1.1.26 #define VIR\_DOMAIN\_BLOCK\_STATS\_FLUSH\_REQ "flush\_operations"**

VIR\_DOMAIN\_BLOCK\_STATS\_FLUSH\_REQ:

Macro represents the total flush requests of the block device, as an llong.

**4.1.1.27 #define VIR\_DOMAIN\_BLOCK\_STATS\_FLUSH\_TOTAL\_TIMES "flush\_total\_times"**

VIR\_DOMAIN\_BLOCK\_STATS\_FLUSH\_TOTAL\_TIMES:

Macro represents the total time spend on cache flushing in nano-seconds of the block device, as an llong.

**4.1.1.28 #define VIR\_DOMAIN\_BLOCK\_STATS\_READ\_BYTES "rd\_bytes"**

VIR\_DOMAIN\_BLOCK\_STATS\_READ\_BYTES:

Macro represents the total number of read bytes of the block device, as an llong.

**4.1.1.29 #define VIR\_DOMAIN\_BLOCK\_STATS\_READ\_REQ "rd\_operations"**

VIR\_DOMAIN\_BLOCK\_STATS\_READ\_REQ:

Macro represents the total read requests of the block device, as an llong.

**4.1.1.30 #define VIR\_DOMAIN\_BLOCK\_STATS\_READ\_TOTAL\_TIMES "rd\_total\_times"**

VIR\_DOMAIN\_BLOCK\_STATS\_READ\_TOTAL\_TIMES:

Macro represents the total time spend on cache reads in nano-seconds of the block device, as an llong.

**4.1.1.31 #define VIR\_DOMAIN\_BLOCK\_STATS\_WRITE\_BYTES "wr\_bytes"**

VIR\_DOMAIN\_BLOCK\_STATS\_WRITE\_BYTES:

Macro represents the total number of write bytes of the block device, as an llong.

**4.1.1.32 #define VIR\_DOMAIN\_BLOCK\_STATS\_WRITE\_REQ "wr\_operations"**

VIR\_DOMAIN\_BLOCK\_STATS\_WRITE\_REQ:

Macro represents the total write requests of the block device, as an llong.

4.1.1.33 `#define VIR_DOMAIN_BLOCK_STATS_WRITE_TOTAL_TIMES "wr_total_times"`

`VIR_DOMAIN_BLOCK_STATS_WRITE_TOTAL_TIMES`:

Macro represents the total time spend on cache writes in nano-seconds of the block device, as an llong.

4.1.1.34 `#define VIR_DOMAIN_CPU_STATS_CPUTIME "cpu_time"`

`VIR_DOMAIN_CPU_STATS_CPUTIME`: cpu usage (sum of both vcpu and hypervisor usage) in nanoseconds, as a ullong

4.1.1.35 `#define VIR_DOMAIN_CPU_STATS_SYSTEMTIME "system_time"`

`VIR_DOMAIN_CPU_STATS_SYSTEMTIME`: cpu time charged to system instructions in nanoseconds, as a ullong

4.1.1.36 `#define VIR_DOMAIN_CPU_STATS_USERTIME "user_time"`

`VIR_DOMAIN_CPU_STATS_USERTIME`: cpu time charged to user instructions in nanoseconds, as a ullong

4.1.1.37 `#define VIR_DOMAIN_CPU_STATS_VCPU_TIME "vcpu_time"`

`VIR_DOMAIN_CPU_STATS_VCPU_TIME`: vcpu usage in nanoseconds (cpu\_time excluding hypervisor time), as a ullong

4.1.1.38 `#define VIR_DOMAIN_EVENT_CALLBACK( cb ) ((virConnectDomainEventGenericCallback)(cb))`

`VIR_DOMAIN_EVENT_CALLBACK`:

Used to cast the event specific callback into the generic one for use for virDomainEventRegister

4.1.1.39 `#define VIR_DOMAIN_MEMORY_FIELD_LENGTH VIR_TYPED_PARAM_FIELD_LENGTH`

`VIR_DOMAIN_MEMORY_FIELD_LENGTH`:

Macro providing the field length of virMemoryParameter. Provided for backwards compatibility; VIR\_TYPED\_PARAM\_FIELD\_LENGTH is the preferred value since 0.9.2.

4.1.1.40 `#define VIR_DOMAIN_MEMORY_HARD_LIMIT "hard_limit"`

`VIR_DOMAIN_MEMORY_HARD_LIMIT`:

Macro for the memory tunable hard\_limit: it represents the maximum memory the guest can use, as a ullong.

4.1.1.41 `#define VIR_DOMAIN_MEMORY_MIN_GUARANTEE "min_guarantee"`

`VIR_DOMAIN_MEMORY_MIN_GUARANTEE`:

Macro for the memory tunable min\_guarantee: it represents the minimum memory guaranteed to be reserved for the guest, as a ullong.

4.1.1.42 `#define VIR_DOMAIN_MEMORY_PARAM_UNLIMITED 9007199254740991LL /* = INT64_MAX >> 10 */`

`VIR_DOMAIN_MEMORY_PARAM_UNLIMITED`:

Macro providing the virMemoryParameter value that indicates "unlimited"

**4.1.1.43 #define VIR\_DOMAIN\_MEMORY\_SOFT\_LIMIT "soft\_limit"****VIR\_DOMAIN\_MEMORY\_SOFT\_LIMIT:**

Macro for the memory tunable `soft_limit`: it represents the memory upper limit enforced during memory contention, as a `ullong`.

**4.1.1.44 #define VIR\_DOMAIN\_MEMORY\_SWAP\_HARD\_LIMIT "swap\_hard\_limit"****VIR\_DOMAIN\_MEMORY\_SWAP\_HARD\_LIMIT:**

Macro for the swap tunable `swap_hard_limit`: it represents the maximum swap plus memory the guest can use, as a `ullong`. This limit has to be more than `VIR_DOMAIN_MEMORY_HARD_LIMIT`.

**4.1.1.45 #define VIR\_DOMAIN\_NUMA\_MODE "numa\_mode"****VIR\_DOMAIN\_NUMA\_MODE:**

Macro for typed parameter name that lists the numa mode of a domain, as an `int` containing a `virDomainNumatuneMemMode` value.

**4.1.1.46 #define VIR\_DOMAIN\_NUMA\_NODESET "numa\_nodeset"****VIR\_DOMAIN\_NUMA\_NODESET:**

Macro for typed parameter name that lists the numa nodeset of a domain, as a string.

**4.1.1.47 #define VIR\_DOMAIN\_SCHED\_FIELD\_LENGTH VIR\_TYPED\_PARAM\_FIELD\_LENGTH****VIR\_DOMAIN\_SCHED\_FIELD\_LENGTH:**

Macro providing the field length of `virSchedParameter`. Provided for backwards compatibility; `VIR_TYPED_PARAM_FIELD_LENGTH` is the preferred value since 0.9.2.

**4.1.1.48 #define VIR\_DOMAIN\_SCHEDULER\_CAP "cap"****VIR\_DOMAIN\_SCHEDULER\_CAP:**

Macro represents the maximum scheduler cap, when using the credit scheduler, as a `uint`.

**4.1.1.49 #define VIR\_DOMAIN\_SCHEDULER\_CPU\_SHARES "cpu\_shares"****VIR\_DOMAIN\_SCHEDULER\_CPU\_SHARES:**

Macro represents proportional weight of the scheduler used on the host cpu, when using the posix scheduler, as a `ullong`.

**4.1.1.50 #define VIR\_DOMAIN\_SCHEDULER\_EMULATOR\_PERIOD "emulator\_period"****VIR\_DOMAIN\_SCHEDULER\_EMULATOR\_PERIOD:**

Macro represents the enforcement period for a quota in microseconds, when using the posix scheduler, for all emulator activity not tied to vcpus, as a `ullong`.

**4.1.1.51 #define VIR\_DOMAIN\_SCHEDULER\_EMULATOR\_QUOTA "emulator\_quota"****VIR\_DOMAIN\_SCHEDULER\_EMULATOR\_QUOTA:**

Macro represents the maximum bandwidth to be used within a period for all emulator activity not tied to vcpus, when using the posix scheduler, as an llong.

**4.1.1.52 #define VIR\_DOMAIN\_SCHEDULER\_LIMIT "limit"****VIR\_DOMAIN\_SCHEDULER\_LIMIT:**

Macro represents the scheduler limit value, when using the allocation scheduler, as an llong.

**4.1.1.53 #define VIR\_DOMAIN\_SCHEDULER\_RESERVATION "reservation"****VIR\_DOMAIN\_SCHEDULER\_RESERVATION:**

Macro represents the scheduler reservation value, when using the allocation scheduler, as an llong.

**4.1.1.54 #define VIR\_DOMAIN\_SCHEDULER\_SHARES "shares"****VIR\_DOMAIN\_SCHEDULER\_SHARES:**

Macro represents the scheduler shares value, when using the allocation scheduler, as an int.

**4.1.1.55 #define VIR\_DOMAIN\_SCHEDULER\_VCPU\_PERIOD "vcpu\_period"****VIR\_DOMAIN\_SCHEDULER\_VCPU\_PERIOD:**

Macro represents the enforcement period for a quota, in microseconds, for vcpus only, when using the posix scheduler, as a ullong.

**4.1.1.56 #define VIR\_DOMAIN\_SCHEDULER\_VCPU\_QUOTA "vcpu\_quota"****VIR\_DOMAIN\_SCHEDULER\_VCPU\_QUOTA:**

Macro represents the maximum bandwidth to be used within a period for vcpus only, when using the posix scheduler, as an llong.

**4.1.1.57 #define VIR\_DOMAIN\_SCHEDULER\_WEIGHT "weight"****VIR\_DOMAIN\_SCHEDULER\_WEIGHT:**

Macro represents the relative weight, when using the credit scheduler, as a uint.

**4.1.1.58 #define VIR\_DOMAIN\_SEND\_KEY\_MAX\_KEYS 16****VIR\_DOMAIN\_SEND\_KEY\_MAX\_KEYS:**

Maximum number of keycodes that can be sent in one [virDomainSendKey\(\)](#) call.

**4.1.1.59 #define VIR\_EXPORT\_VAR extern**

4.1.1.60 **#define VIR\_GET\_CPUMAP( *cpumaps*, *maplen*, *vcpu* )** `&(cpumaps[(vcpu)*(maplen)])`

VIR\_GET\_CPUMAP: : pointer to an array of cpumap (in 8-bit bytes) (IN) : the length (in bytes) of one cpumap : the virtual CPU number

This macro is to be used in conjunction with [virDomainGetVcpus\(\)](#) and [virDomainPinVcpu\(\)](#) APIs. VIR\_GET\_CPU-MAP macro returns a pointer to the cpumap of the specified vcpu from cpumaps array.

4.1.1.61 **#define VIR\_NODE\_CPU\_STATS\_FIELD\_LENGTH** 80

VIR\_NODE\_CPU\_STATS\_FIELD\_LENGTH:

Macro providing the field length of virNodeCPUStats

4.1.1.62 **#define VIR\_NODE\_CPU\_STATS\_IDLE** "idle"

VIR\_NODE\_CPU\_STATS\_IDLE:

The cumulative idle CPU time, since the node booting up (in nanoseconds).

4.1.1.63 **#define VIR\_NODE\_CPU\_STATS\_IOWAIT** "iowait"

VIR\_NODE\_CPU\_STATS\_IOWAIT:

The cumulative I/O wait CPU time, since the node booting up (in nanoseconds).

4.1.1.64 **#define VIR\_NODE\_CPU\_STATS\_KERNEL** "kernel"

VIR\_NODE\_CPU\_STATS\_KERNEL:

Macro for the cumulative CPU time which was spent by the kernel, since the node booting up (in nanoseconds).

4.1.1.65 **#define VIR\_NODE\_CPU\_STATS\_USER** "user"

VIR\_NODE\_CPU\_STATS\_USER:

The cumulative CPU time which was spent by user processes, since the node booting up (in nanoseconds).

4.1.1.66 **#define VIR\_NODE\_CPU\_STATS\_UTILIZATION** "utilization"

VIR\_NODE\_CPU\_STATS\_UTILIZATION:

The CPU utilization of a node. The usage value is in percent and 100% represents all CPUs of the node.

4.1.1.67 **#define VIR\_NODE\_MEMORY\_SHARED\_FULL\_SCANS** "shm\_full\_scans"

4.1.1.68 **#define VIR\_NODE\_MEMORY\_SHARED\_PAGES\_SHARED** "shm\_pages\_shared"

4.1.1.69 **#define VIR\_NODE\_MEMORY\_SHARED\_PAGES\_SHARING** "shm\_pages\_sharing"

4.1.1.70 **#define VIR\_NODE\_MEMORY\_SHARED\_PAGES\_TO\_SCAN** "shm\_pages\_to\_scan"

4.1.1.71 **#define VIR\_NODE\_MEMORY\_SHARED\_PAGES\_UNSHARED** "shm\_pages\_unshared"

4.1.1.72 **#define VIR\_NODE\_MEMORY\_SHARED\_PAGES\_VOLATILE** "shm\_pages\_volatile"

4.1.1.73 `#define VIR_NODE_MEMORY_SHARED_SLEEP_MILLISECS "shm_sleep_millisecs"`

4.1.1.74 `#define VIR_NODE_MEMORY_STATS_BUFFERS "buffers"`

`VIR_NODE_MEMORY_STATS_BUFFERS:`

Macro for the buffer memory: On Linux, it is only returned in case of `VIR_NODE_MEMORY_STATS_ALL_CELLS`.

4.1.1.75 `#define VIR_NODE_MEMORY_STATS_CACHED "cached"`

`VIR_NODE_MEMORY_STATS_CACHED:`

Macro for the cached memory: On Linux, it is only returned in case of `VIR_NODE_MEMORY_STATS_ALL_CELLS`.

4.1.1.76 `#define VIR_NODE_MEMORY_STATS_FIELD_LENGTH 80`

`VIR_NODE_MEMORY_STATS_FIELD_LENGTH:`

Macro providing the field length of `virNodeMemoryStats`

4.1.1.77 `#define VIR_NODE_MEMORY_STATS_FREE "free"`

`VIR_NODE_MEMORY_STATS_FREE:`

Macro for the free memory of specified cell: On Linux, it includes buffer and cached memory, in case of `VIR_NODE_MEMORY_STATS_ALL_CELLS`.

4.1.1.78 `#define VIR_NODE_MEMORY_STATS_TOTAL "total"`

`VIR_NODE_MEMORY_STATS_TOTAL:`

Macro for the total memory of specified cell: it represents the maximum memory.

4.1.1.79 `#define VIR_NODEINFO_MAXCPUS( nodeinfo ) ((nodeinfo).nodes*(nodeinfo).sockets*(nodeinfo).cores*(nodeinfo).threads)`

`VIR_NODEINFO_MAXCPUS:` : `virNodeInfo` instance

This macro is to calculate the total number of CPUs supported but not necessary active in the host.

4.1.1.80 `#define VIR_SECURITY_DOI_BUFLen (256 + 1)`

`VIR_SECURITY_DOI_BUFLen:`

Macro providing the maximum length of the `virSecurityModel` doi string.

4.1.1.81 `#define VIR_SECURITY_LABEL_BUFLen (4096 + 1)`

`VIR_SECURITY_LABEL_BUFLen:`

Macro providing the maximum length of the `virSecurityLabel` label string. Note that this value is based on that used by Labeled NFS.

#### 4.1.1.82 `#define VIR_SECURITY_MODEL_BUFLen (256 + 1)`

`VIR_SECURITY_MODEL_BUFLen`:

Macro providing the maximum length of the `virSecurityModel` model string.

#### 4.1.1.83 `#define VIR_TYPED_PARAM_FIELD_LENGTH 80`

`VIR_TYPED_PARAM_FIELD_LENGTH`:

Macro providing the field length of `virTypedParameter` name

#### 4.1.1.84 `#define VIR_UNUSE_CPU( cpumap, cpu ) (cpumap[(cpu)/8] &= ~(1<<((cpu)%8)))`

`VIR_UNUSE_CPU`: : pointer to a bit map of real CPUs (in 8-bit bytes) (IN/OUT) : the physical CPU number

This macro is to be used in conjunction with `virDomainPinVcpu()` API. `USE_CPU` macro reset the bit (CPU not usable) of the related cpu in `cpumap`.

#### 4.1.1.85 `#define VIR_USE_CPU( cpumap, cpu ) (cpumap[(cpu)/8] |= (1<<((cpu)%8)))`

`VIR_USE_CPU`: : pointer to a bit map of real CPUs (in 8-bit bytes) (IN/OUT) : the physical CPU number

This macro is to be used in conjunction with `virDomainPinVcpu()` API. `USE_CPU` macro set the bit (CPU usable) of the related cpu in `cpumap`.

#### 4.1.1.86 `#define VIR_UUID_BUFLen (16)`

`VIR_UUID_BUFLen`:

This macro provides the length of the buffer required for `virDomainGetUUID()`

#### 4.1.1.87 `#define VIR_UUID_STRING_BUFLen (36+1)`

`VIR_UUID_STRING_BUFLen`:

This macro provides the length of the buffer required for `virDomainGetUUIDString()`

### 4.1.2 Typedef Documentation

#### 4.1.2.1 `typedef struct _virTypedParameter virBlkioParameter`

#### 4.1.2.2 `typedef virBlkioParameter* virBlkioParameterPtr`

`virBlkioParameterPtr`:

a `virBlkioParameterPtr` is a pointer to a `virBlkioParameter` structure. Provided for backwards compatibility; `virTypedParameterPtr` is the preferred alias since 0.9.2.

#### 4.1.2.3 `typedef struct _virConnect virConnect`

`virConnect`:

a `virConnect` is a private structure representing a connection to the Hypervisor.



## 4.1.2.4 typedef struct \_virConnectAuth virConnectAuth

## 4.1.2.5 typedef int(\* virConnectAuthCallbackPtr)(virConnectCredentialPtr cred, unsigned int ncred, void \*cbdata)

virConnectAuthCallbackPtr: : list of virConnectCredential object to fetch from user : size of cred list : opaque data passed to virConnectOpenAuth

When authentication requires one or more interactions, this callback is invoked. For each interaction supplied, data must be gathered from the user and filled in to the 'result' and 'resultlen' fields. If an interaction cannot be filled, fill in NULL and 0.

Returns 0 if all interactions were filled, or -1 upon error

## 4.1.2.6 typedef virConnectAuth\* virConnectAuthPtr

## 4.1.2.7 typedef void(\* virConnectCloseFunc)(virConnectPtr conn, int reason, void \*opaque)

## 4.1.2.8 typedef struct \_virConnectCredential virConnectCredential

## 4.1.2.9 typedef virConnectCredential\* virConnectCredentialPtr

## 4.1.2.10 typedef void(\* virConnectDomainEventBalloonChangeCallback)(virConnectPtr conn, virDomainPtr dom, unsigned long long actual, void \*opaque)

virConnectDomainEventBalloonChangeCallback: : connection object : domain on which the event occurred : the new balloon level measured in kibibytes(blocks of 1024 bytes) : application specified data

The callback signature to use when registering for an event of type VIR\_DOMAIN\_EVENT\_ID\_BALLOON\_CHANGE with [virConnectDomainEventRegisterAny\(\)](#)

## 4.1.2.11 typedef void(\* virConnectDomainEventBlockJobCallback)(virConnectPtr conn, virDomainPtr dom, const char \*disk, int type, int status, void \*opaque)

virConnectDomainEventBlockJobCallback: : connection object : domain on which the event occurred : fully-qualified filename of the affected disk : type of block job (virDomainBlockJobType) : final status of the operation (virConnectDomainEventBlockJobStatus)

The callback signature to use when registering for an event of type VIR\_DOMAIN\_EVENT\_ID\_BLOCK\_JOB with [virConnectDomainEventRegisterAny\(\)](#)

## 4.1.2.12 typedef int(\* virConnectDomainEventCallback)(virConnectPtr conn, virDomainPtr dom, int event, int detail, void \*opaque)

virConnectDomainEventCallback: : virConnect connection : The domain on which the event occurred : The specific virDomainEventType which occurred : event specific detail information : opaque user data

A callback function to be registered, and called when a domain event occurs

## 4.1.2.13 typedef void(\* virConnectDomainEventDiskChangeCallback)(virConnectPtr conn, virDomainPtr dom, const char \*oldSrcPath, const char \*newSrcPath, const char \*devAlias, int reason, void \*opaque)

virConnectDomainEventDiskChangeCallback: : connection object : domain on which the event occurred : old source path : new source path : device alias name : reason why this callback was called; any of virConnectDomainEventDiskChangeReason : application specified data

This callback occurs when disk gets changed. However, not all will cause both and to be non-NULL. Please see virConnectDomainEventDiskChangeReason for more details.

The callback signature to use when registering for an event of type `VIR_DOMAIN_EVENT_ID_DISK_CHANGE` with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.14** `typedef void(* virConnectDomainEventGenericCallback)(virConnectPtr conn, virDomainPtr dom, void *opaque)`

**4.1.2.15** `typedef void(* virConnectDomainEventGraphicsCallback)(virConnectPtr conn, virDomainPtr dom, int phase, virDomainEventGraphicsAddressPtr local, virDomainEventGraphicsAddressPtr remote, const char *authScheme, virDomainEventGraphicsSubjectPtr subject, void *opaque)`

`virConnectDomainEventGraphicsCallback`: : connection object : domain on which the event occurred : the phase of the connection : the local server address : the remote client address : the authentication scheme activated : the authenticated subject (user) : application specified data

The callback signature to use when registering for an event of type `VIR_DOMAIN_EVENT_ID_GRAPHICS` with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.16** `typedef void(* virConnectDomainEventIOErrorCallback)(virConnectPtr conn, virDomainPtr dom, const char *srcPath, const char *devAlias, int action, void *opaque)`

`virConnectDomainEventIOErrorCallback`: : connection object : domain on which the event occurred : The host file on which the IO error occurred : The guest device alias associated with the path : action that is to be taken due to the IO error : application specified data

The callback signature to use when registering for an event of type `VIR_DOMAIN_EVENT_ID_IO_ERROR` with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.17** `typedef void(* virConnectDomainEventIOErrorReasonCallback)(virConnectPtr conn, virDomainPtr dom, const char *srcPath, const char *devAlias, int action, const char *reason, void *opaque)`

`virConnectDomainEventIOErrorReasonCallback`: : connection object : domain on which the event occurred : The host file on which the IO error occurred : The guest device alias associated with the path : action that is to be taken due to the IO error : the cause of the IO error : application specified data

The callback signature to use when registering for an event of type `VIR_DOMAIN_EVENT_ID_IO_ERROR` with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.18** `typedef void(* virConnectDomainEventPMSuspendCallback)(virConnectPtr conn, virDomainPtr dom, int reason, void *opaque)`

`virConnectDomainEventPMSuspendCallback`: : connection object : domain on which the event occurred : reason why the callback was called, unused currently, always passes 0 : application specified data

This callback occurs when the guest is waken up.

The callback signature to use when registering for an event of type `VIR_DOMAIN_EVENT_ID_PMSuspend` with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.19** `typedef void(* virConnectDomainEventPMWakeupCallback)(virConnectPtr conn, virDomainPtr dom, int reason, void *opaque)`

`virConnectDomainEventPMWakeupCallback`: : connection object : domain on which the event occurred : reason why the callback was called, unused currently, always passes 0 : application specified data

This callback occurs when the guest is waken up.

The callback signature to use when registering for an event of type `VIR_DOMAIN_EVENT_ID_PMWAKEUP` with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.20** `typedef void(* virConnectDomainEventRTCChangeCallback)(virConnectPtr conn, virDomainPtr dom, long long utcoffset, void *opaque)`

virConnectDomainEventRTCChangeCallback: : connection object : domain on which the event occurred : the new RTC offset from UTC, measured in seconds : application specified data

The callback signature to use when registering for an event of type VIR\_DOMAIN\_EVENT\_ID\_RTC\_CHANGE with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.21** `typedef void(* virConnectDomainEventTrayChangeCallback)(virConnectPtr conn, virDomainPtr dom, const char *devAlias, int reason, void *opaque)`

virConnectDomainEventTrayChangeCallback: : connection object : domain on which the event occurred : device alias : why the tray status was changed? : application specified data

This callback occurs when the tray of a removable device is moved.

The callback signature to use when registering for an event of type VIR\_DOMAIN\_EVENT\_ID\_TRAY\_CHANGE with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.22** `typedef void(* virConnectDomainEventWatchdogCallback)(virConnectPtr conn, virDomainPtr dom, int action, void *opaque)`

virConnectDomainEventWatchdogCallback: : connection object : domain on which the event occurred : action that is to be taken due to watchdog firing : application specified data

The callback signature to use when registering for an event of type VIR\_DOMAIN\_EVENT\_ID\_WATCHDOG with [virConnectDomainEventRegisterAny\(\)](#)

**4.1.2.23** `typedef virConnect* virConnectPtr`

virConnectPtr:

a virConnectPtr is pointer to a virConnect private structure, this is the type used to reference a connection to the Hypervisor in the API.

**4.1.2.24** `typedef struct _virDomain virDomain`

virDomain:

a virDomain is a private structure representing a domain.

**4.1.2.25** `typedef struct _virDomainBlockInfo virDomainBlockInfo`

virDomainBlockInfo:

This struct provides information about the size of a block device backing store

Examples:

- Fully allocated raw file in filesystem:
  - capacity, allocation, physical: All the same
- Sparse raw file in filesystem:
  - capacity: logical size of the file
  - allocation, physical: number of blocks allocated to file

- qcow2 file in filesystem
  - capacity: logical size from qcow2 header
  - allocation, physical: logical size of the file / highest qcow extent (identical)
- qcow2 file in a block device
  - capacity: logical size from qcow2 header
  - allocation: highest qcow extent written
  - physical: size of the block device container

**4.1.2.26** `typedef virDomainBlockInfo* virDomainBlockInfoPtr`

**4.1.2.27** `typedef unsigned long long virDomainBlockJobCursor`

**4.1.2.28** `typedef struct _virDomainBlockJobInfo virDomainBlockJobInfo`

**4.1.2.29** `typedef virDomainBlockJobInfo* virDomainBlockJobInfoPtr`

**4.1.2.30** `typedef virDomainBlockStatsStruct* virDomainBlockStatsPtr`

virDomainBlockStatsPtr:

A pointer to a virDomainBlockStats structure

**4.1.2.31** `typedef struct _virDomainBlockStats virDomainBlockStatsStruct`

virDomainBlockStats:

Block device stats for virDomainBlockStats.

Hypervisors may return a field set to ((long long)-1) which indicates that the hypervisor does not support that statistic.

NB. Here 'long long' means 64 bit integer.

**4.1.2.32** `typedef struct _virDomainControllInfo virDomainControllInfo`

virDomainControllInfo:

Structure filled in by virDomainGetControllInfo and providing details about current state of control interface to a domain.

**4.1.2.33** `typedef virDomainControllInfo* virDomainControllInfoPtr`

virDomainControllInfoPtr:

Pointer to virDomainControllInfo structure.

**4.1.2.34** `typedef struct _virDomainDiskError virDomainDiskError`

virDomainDiskError:

4.1.2.35 `typedef virDomainDiskError* virDomainDiskErrorPtr`

4.1.2.36 `typedef struct _virDomainEventGraphicsAddress virDomainEventGraphicsAddress`

4.1.2.37 `typedef virDomainEventGraphicsAddress* virDomainEventGraphicsAddressPtr`

4.1.2.38 `typedef struct _virDomainEventGraphicsSubject virDomainEventGraphicsSubject`

4.1.2.39 `typedef struct _virDomainEventGraphicsSubjectIdentity virDomainEventGraphicsSubjectIdentity`

4.1.2.40 `typedef virDomainEventGraphicsSubjectIdentity* virDomainEventGraphicsSubjectIdentityPtr`

4.1.2.41 `typedef virDomainEventGraphicsSubject* virDomainEventGraphicsSubjectPtr`

4.1.2.42 `typedef struct _virDomainInfo virDomainInfo`

`virDomainInfoPtr`:

a `virDomainInfo` is a structure filled by [virDomainGetInfo\(\)](#) and extracting runtime information for a given active Domain

4.1.2.43 `typedef virDomainInfo* virDomainInfoPtr`

`virDomainInfoPtr`:

a `virDomainInfoPtr` is a pointer to a `virDomainInfo` structure.

4.1.2.44 `typedef virDomainInterfaceStatsStruct* virDomainInterfaceStatsPtr`

`virDomainInterfaceStatsPtr`:

A pointer to a `virDomainInterfaceStats` structure

4.1.2.45 `typedef struct _virDomainInterfaceStats virDomainInterfaceStatsStruct`

`virDomainInterfaceStats`:

Network interface stats for `virDomainInterfaceStats`.

Hypervisors may return a field set to `((long long)-1)` which indicates that the hypervisor does not support that statistic.

NB. Here 'long long' means 64 bit integer.

4.1.2.46 `typedef struct _virDomainJobInfo virDomainJobInfo`

4.1.2.47 `typedef virDomainJobInfo* virDomainJobInfoPtr`

4.1.2.48 `typedef virDomainMemoryStatStruct* virDomainMemoryStatPtr`

4.1.2.49 `typedef struct _virDomainMemoryStat virDomainMemoryStatStruct`

4.1.2.50 `typedef virDomain* virDomainPtr`

`virDomainPtr`:

a `virDomainPtr` is pointer to a `virDomain` private structure, this is the type used to reference a domain in the API.

#### 4.1.2.51 `typedef struct _virDomainSnapshot virDomainSnapshot`

`virDomainSnapshot`:

a `virDomainSnapshot` is a private structure representing a snapshot of a domain.

#### 4.1.2.52 `typedef virDomainSnapshot* virDomainSnapshotPtr`

`virDomainSnapshotPtr`:

a `virDomainSnapshotPtr` is pointer to a `virDomainSnapshot` private structure, and is the type used to reference a domain snapshot in the API.

#### 4.1.2.53 `typedef int(* virEventAddHandleFunc)(int fd, int event, virEventHandleCallback cb, void *opaque, virFreeCallback ff)`

`virEventAddHandleFunc`: : file descriptor to listen on : bitset of events on which to fire the callback : the callback to be called when an event occurs : user data to pass to the callback : the callback invoked to free opaque data blob

Part of the `EventImpl`, this callback adds a file handle callback to listen for specific events. The same file handle can be registered multiple times provided the requested event sets are non-overlapping

If the opaque user data requires free'ing when the handle is unregistered, then a 2nd callback can be supplied for this purpose. This callback needs to be invoked from a clean stack. If 'ff' callbacks are invoked directly from the `virEventRemoveHandleFunc` they will likely deadlock in `libvirt`.

Returns a handle watch number to be used for updating and unregistering for events

#### 4.1.2.54 `typedef int(* virEventAddTimeoutFunc)(int timeout, virEventTimeoutCallback cb, void *opaque, virFreeCallback ff)`

`virEventAddTimeoutFunc`: : The timeout to monitor : the callback to call when timeout has expired : user data to pass to the callback : the callback invoked to free opaque data blob

Part of the `EventImpl`, this user-defined callback handles adding an event timeout.

If the opaque user data requires free'ing when the handle is unregistered, then a 2nd callback can be supplied for this purpose.

Returns a timer value

#### 4.1.2.55 `typedef void(* virEventHandleCallback)(int watch, int fd, int events, void *opaque)`

`virEventHandleCallback`:

: watch on which the event occurred : file handle on which the event occurred : bitset of events from `virEventHandle-Type` constants : user data registered with handle

Callback for receiving file handle events. The callback will be invoked once for each event which is pending.

#### 4.1.2.56 `typedef int(* virEventRemoveHandleFunc)(int watch)`

`virEventRemoveHandleFunc`: : file descriptor watch to stop listening on

Part of the `EventImpl`, this user-provided callback is notified when an fd is no longer being listened on.

If a `virEventHandleFreeFunc` was supplied when the handle was registered, it will be invoked some time during, or after this function call, when it is safe to release the user data.

**4.1.2.57 typedef int(\* virEventRemoveTimeoutFunc)(int timer)**

virEventRemoveTimeoutFunc: : the timer to remove

Part of the EventImpl, this user-defined callback removes a timer

If a virEventTimeoutFreeFunc was supplied when the handle was registered, it will be invoked some time during, or after this function call, when it is safe to release the user data.

Returns 0 on success, -1 on failure

**4.1.2.58 typedef void(\* virEventTimeoutCallback)(int timer, void \*opaque)**

virEventTimeoutCallback:

: timer id emitting the event : user data registered with handle

callback for receiving timer events

**4.1.2.59 typedef void(\* virEventUpdateHandleFunc)(int watch, int event)**

virEventUpdateHandleFunc: : file descriptor watch to modify : new events to listen on

Part of the EventImpl, this user-provided callback is notified when events to listen on change

**4.1.2.60 typedef void(\* virEventUpdateTimeoutFunc)(int timer, int timeout)**

virEventUpdateTimeoutFunc: : the timer to modify : the new timeout value

Part of the EventImpl, this user-defined callback updates an event timeout.

**4.1.2.61 typedef void(\* virFreeCallback)(void \*opaque)****4.1.2.62 typedef struct \_virInterface virInterface**

virInterface:

a virInterface is a private structure representing a virtual interface.

**4.1.2.63 typedef virInterface\* virInterfacePtr**

virInterfacePtr:

a virInterfacePtr is pointer to a virInterface private structure, this is the type used to reference a virtual interface in the API.

**4.1.2.64 typedef struct \_virTypedParameter virMemoryParameter****4.1.2.65 typedef virMemoryParameter\* virMemoryParameterPtr**

virMemoryParameterPtr:

a virMemoryParameterPtr is a pointer to a virMemoryParameter structure. Provided for backwards compatibility; virTypedParameterPtr is the preferred alias since 0.9.2.

**4.1.2.66 typedef struct \_virNetwork virNetwork**

virNetwork:

a virNetwork is a private structure representing a virtual network.

#### 4.1.2.67 `typedef virNetwork* virNetworkPtr`

virNetworkPtr:

a virNetworkPtr is pointer to a virNetwork private structure, this is the type used to reference a virtual network in the API.

#### 4.1.2.68 `typedef struct _virNodeCPUStats virNodeCPUStats`

virNodeCPUStats:

a virNodeCPUStats is a structure filled by [virNodeGetCPUStats\(\)](#) providing information about the CPU stats of the node.

#### 4.1.2.69 `typedef virNodeCPUStats* virNodeCPUStatsPtr`

virNodeCPUStatsPtr:

a virNodeCPUStatsPtr is a pointer to a virNodeCPUStats structure.

#### 4.1.2.70 `typedef struct _virNodeDevice virNodeDevice`

virNodeDevice:

A virNodeDevice contains a node (host) device details.

#### 4.1.2.71 `typedef virNodeDevice* virNodeDevicePtr`

virNodeDevicePtr:

A virNodeDevicePtr is a pointer to a virNodeDevice structure. Get one via [virNodeDeviceLookupByKey](#), [virNodeDeviceLookupByName](#), or [virNodeDeviceCreate](#). Be sure to Call [virNodeDeviceFree](#) when done using a virNodeDevicePtr obtained from any of the above functions to avoid leaking memory.

#### 4.1.2.72 `typedef struct _virNodeInfo virNodeInfo`

virNodeInfoPtr:

a virNodeInfo is a structure filled by [virNodeGetInfo\(\)](#) and providing the information for the Node.

#### 4.1.2.73 `typedef virNodeInfo* virNodeInfoPtr`

virNodeInfoPtr:

a virNodeInfoPtr is a pointer to a virNodeInfo structure.

#### 4.1.2.74 `typedef struct _virNodeMemoryStats virNodeMemoryStats`

virNodeMemoryStats:

a virNodeMemoryStats is a structure filled by [virNodeGetMemoryStats\(\)](#) providing information about the memory of the node.



**4.1.2.75 typedef virNodeMemoryStats\* virNodeMemoryStatsPtr**

virNodeMemoryStatsPtr:

a virNodeMemoryStatsPtr is a pointer to a virNodeMemoryStats structure.

**4.1.2.76 typedef struct \_virNWFilter virNWFilter**

virNWFilter:

a virNWFilter is a private structure representing a network filter

**4.1.2.77 typedef virNWFilter\* virNWFilterPtr**

virNWFilterPtr:

a virNWFilterPtr is pointer to a virNWFilter private structure, this is the type used to reference a network filter in the API.

**4.1.2.78 typedef struct \_virTypedParameter virSchedParameter****4.1.2.79 typedef virSchedParameter\* virSchedParameterPtr**

virSchedParameterPtr:

a virSchedParameterPtr is a pointer to a virSchedParameter structure. Provided for backwards compatibility; virTypedParameterPtr is the preferred alias since 0.9.2.

**4.1.2.80 typedef struct \_virSecret virSecret**

virSecret:

A virSecret stores a secret value (e.g. a passphrase or encryption key) and associated metadata.

**4.1.2.81 typedef virSecret\* virSecretPtr****4.1.2.82 typedef struct \_virSecurityLabel virSecurityLabel**

virSecurityLabel:

a virSecurityLabel is a structure filled by [virDomainGetSecurityLabel\(\)](#), providing the security label and associated attributes for the specified domain.

**4.1.2.83 typedef virSecurityLabel\* virSecurityLabelPtr**

virSecurityLabelPtr:

a virSecurityLabelPtr is a pointer to a virSecurityLabel.

**4.1.2.84 typedef struct \_virSecurityModel virSecurityModel**

virSecurityModel:

a virSecurityModel is a structure filled by [virNodeGetSecurityModel\(\)](#), providing the per-hypervisor security model and DOI attributes for the specified domain.

**4.1.2.85 typedef virSecurityModel\* virSecurityModelPtr**

virSecurityModelPtr:

a virSecurityModelPtr is a pointer to a virSecurityModel.

**4.1.2.86 typedef struct \_virStoragePool virStoragePool**

virStoragePool:

a virStoragePool is a private structure representing a storage pool

**4.1.2.87 typedef struct \_virStoragePoolInfo virStoragePoolInfo****4.1.2.88 typedef virStoragePoolInfo\* virStoragePoolInfoPtr****4.1.2.89 typedef virStoragePool\* virStoragePoolPtr**

virStoragePoolPtr:

a virStoragePoolPtr is pointer to a virStoragePool private structure, this is the type used to reference a storage pool in the API.

**4.1.2.90 typedef struct \_virStorageVol virStorageVol**

virStorageVol:

a virStorageVol is a private structure representing a storage volume

**4.1.2.91 typedef struct \_virStorageVolInfo virStorageVolInfo****4.1.2.92 typedef virStorageVolInfo\* virStorageVolInfoPtr****4.1.2.93 typedef virStorageVol\* virStorageVolPtr**

virStorageVolPtr:

a virStorageVolPtr is pointer to a virStorageVol private structure, this is the type used to reference a storage volume in the API.

**4.1.2.94 typedef struct \_virStream virStream**

virStream:

a virStream is a private structure representing a data stream.

**4.1.2.95 typedef void(\* virStreamEventCallback)(virStreamPtr stream, int events, void \*opaque)**

virStreamEventCallback:

: stream on which the event occurred : bitset of events from virEventHandleType constants : user data registered with handle

Callback for receiving stream events. The callback will be invoked once for each event which is pending.

**4.1.2.96 typedef virStream\* virStreamPtr**

virStreamPtr:

a virStreamPtr is pointer to a virStream private structure, this is the type used to reference a data stream in the API.

**4.1.2.97 typedef int(\* virStreamSinkFunc)(virStreamPtr st, const char \*data, size\_t nbytes, void \*opaque)**

virStreamSinkFunc:

: the stream object : preallocated array to be filled with data : size of the data array : optional application provided data

The virStreamSinkFunc callback is used together with the virStreamRecvAll function for libvirt to provide the data that has been received.

The callback will be invoked multiple times, providing data in small chunks. The application should consume up 'nbytes' from the 'data' array of data and then return the number actual number of bytes consumed. The callback will continue to be invoked until it indicates the end of the stream has been reached. A return value of -1 at any time will abort the receive operation

Returns the number of bytes consumed or -1 upon error

**4.1.2.98 typedef int(\* virStreamSourceFunc)(virStreamPtr st, char \*data, size\_t nbytes, void \*opaque)**

virStreamSourceFunc:

: the stream object : preallocated array to be filled with data : size of the data array : optional application provided data

The virStreamSourceFunc callback is used together with the virStreamSendAll function for libvirt to obtain the data that is to be sent.

The callback will be invoked multiple times, fetching data in small chunks. The application should fill the 'data' array with upto 'nbytes' of data and then return the number actual number of bytes. The callback will continue to be invoked until it indicates the end of the source has been reached by returning 0. A return value of -1 at any time will abort the send operation

Returns the number of bytes filled, 0 upon end of file, or -1 upon error

**4.1.2.99 typedef struct \_virTypedParameter virTypedParameter**

virTypedParameter:

A named parameter, including a type and value.

The types virSchedParameter, virBlkioParameter, and virMemoryParameter are aliases of this type, for use when targetting libvirt earlier than 0.9.2.

**4.1.2.100 typedef virTypedParameter\* virTypedParameterPtr**

virTypedParameterPtr:

a pointer to a virTypedParameter structure.

**4.1.2.101 typedef struct \_virVcpulInfo virVcpulInfo****4.1.2.102 typedef virVcpulInfo\* virVcpulInfoPtr**

### 4.1.3 Enumeration Type Documentation

#### 4.1.3.1 enum virBlkioParameterType

virBlkioParameterType:

A blkio parameter field type. Provided for backwards compatibility; virTypedParameterType is the preferred enum since 0.9.2.

Enumerator

```
VIR_DOMAIN_BLKIO_PARAM_INT  
VIR_DOMAIN_BLKIO_PARAM_UINT  
VIR_DOMAIN_BLKIO_PARAM_LLONG  
VIR_DOMAIN_BLKIO_PARAM_ULLONG  
VIR_DOMAIN_BLKIO_PARAM_DOUBLE  
VIR_DOMAIN_BLKIO_PARAM_BOOLEAN
```

#### 4.1.3.2 enum virConnectCloseReason

Enumerator

```
VIR_CONNECT_CLOSE_REASON_ERROR  
VIR_CONNECT_CLOSE_REASON_EOF  
VIR_CONNECT_CLOSE_REASON_KEEPAIVE  
VIR_CONNECT_CLOSE_REASON_CLIENT
```

#### 4.1.3.3 enum virConnectCredentialType

Enumerator

```
VIR_CRED_USERNAME  
VIR_CRED_AUTHNAME  
VIR_CRED_LANGUAGE  
VIR_CRED_CNONCE  
VIR_CRED_PASSPHRASE  
VIR_CRED_ECHOPROMPT  
VIR_CRED_NOECHOPROMPT  
VIR_CRED_REALM  
VIR_CRED_EXTERNAL
```

#### 4.1.3.4 enum virConnectDomainEventBlockJobStatus

virConnectDomainEventBlockJobStatus:

The final status of a [virDomainBlockPull\(\)](#) or [virDomainBlockRebase\(\)](#) operation

Enumerator

```
VIR_DOMAIN_BLOCK_JOB_COMPLETED  
VIR_DOMAIN_BLOCK_JOB_FAILED  
VIR_DOMAIN_BLOCK_JOB_CANCELED
```

#### 4.1.3.5 enum virConnectDomainEventDiskChangeReason

virConnectDomainEventDiskChangeReason:

The reason describing why this callback is called

Enumerator

***VIR\_DOMAIN\_EVENT\_DISK\_CHANGE\_MISSING\_ON\_START***

#### 4.1.3.6 enum virConnectFlags

virConnectFlags

Flags when opening a connection to a hypervisor

Enumerator

***VIR\_CONNECT\_RO***

***VIR\_CONNECT\_NO\_ALIASES***

#### 4.1.3.7 enum virConnectListAllDomainsFlags

virConnectListAllDomainsFlags:

Flags used to tune which domains are listed by [virConnectListAllDomains\(\)](#). Note that these flags come in groups; if all bits from a group are 0, then that group is not used to filter results.

Enumerator

***VIR\_CONNECT\_LIST\_DOMAINS\_ACTIVE***

***VIR\_CONNECT\_LIST\_DOMAINS\_INACTIVE***

***VIR\_CONNECT\_LIST\_DOMAINS\_PERSISTENT***

***VIR\_CONNECT\_LIST\_DOMAINS\_TRANSIENT***

***VIR\_CONNECT\_LIST\_DOMAINS\_RUNNING***

***VIR\_CONNECT\_LIST\_DOMAINS\_PAUSED***

***VIR\_CONNECT\_LIST\_DOMAINS\_SHUTOFF***

***VIR\_CONNECT\_LIST\_DOMAINS\_OTHER***

***VIR\_CONNECT\_LIST\_DOMAINS\_MANAGEDSAVE***

***VIR\_CONNECT\_LIST\_DOMAINS\_NO\_MANAGEDSAVE***

***VIR\_CONNECT\_LIST\_DOMAINS\_AUTOSTART***

***VIR\_CONNECT\_LIST\_DOMAINS\_NO\_AUTOSTART***

***VIR\_CONNECT\_LIST\_DOMAINS\_HAS\_SNAPSHOT***

***VIR\_CONNECT\_LIST\_DOMAINS\_NO\_SNAPSHOT***

#### 4.1.3.8 enum virConnectListAllInterfacesFlags

Enumerator

***VIR\_CONNECT\_LIST\_INTERFACES\_INACTIVE***

***VIR\_CONNECT\_LIST\_INTERFACES\_ACTIVE***

#### 4.1.3.9 enum virConnectListAllNetworksFlags

Enumerator

*VIR\_CONNECT\_LIST\_NETWORKS\_INACTIVE*  
*VIR\_CONNECT\_LIST\_NETWORKS\_ACTIVE*  
*VIR\_CONNECT\_LIST\_NETWORKS\_PERSISTENT*  
*VIR\_CONNECT\_LIST\_NETWORKS\_TRANSIENT*  
*VIR\_CONNECT\_LIST\_NETWORKS\_AUTOSTART*  
*VIR\_CONNECT\_LIST\_NETWORKS\_NO\_AUTOSTART*

#### 4.1.3.10 enum virConnectListAllNodeDeviceFlags

Enumerator

*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_SYSTEM*  
*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_PCI\_DEV*  
*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_USB\_DEV*  
*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_USB\_INTERFACE*  
*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_NET*  
*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_SCSI\_HOST*  
*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_SCSI\_TARGET*  
*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_SCSI*  
*VIR\_CONNECT\_LIST\_NODE\_DEVICES\_CAP\_STORAGE*

#### 4.1.3.11 enum virConnectListAllSecretsFlags

Enumerator

*VIR\_CONNECT\_LIST\_SECRETS\_EPHEMERAL*  
*VIR\_CONNECT\_LIST\_SECRETS\_NO\_EPHEMERAL*  
*VIR\_CONNECT\_LIST\_SECRETS\_PRIVATE*  
*VIR\_CONNECT\_LIST\_SECRETS\_NO\_PRIVATE*

#### 4.1.3.12 enum virConnectListAllStoragePoolsFlags

Enumerator

*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_INACTIVE*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_ACTIVE*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_PERSISTENT*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_TRANSIENT*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_AUTOSTART*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_NO\_AUTOSTART*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_DIR*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_FS*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_NETFS*  
*VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_LOGICAL*

***VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_DISK***  
***VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_ISCSI***  
***VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_SCSI***  
***VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_MPATH***  
***VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_RBD***  
***VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_SHEEPDOG***

#### 4.1.3.13 enum virCPUCompareResult

Enumerator

***VIR\_CPU\_COMPARE\_ERROR***  
***VIR\_CPU\_COMPARE\_INCOMPATIBLE***  
***VIR\_CPU\_COMPARE\_IDENTICAL***  
***VIR\_CPU\_COMPARE\_SUPERSET***

#### 4.1.3.14 enum virDomainBlockCommitFlags

virDomainBlockCommitFlags:

Flags available for [virDomainBlockCommit\(\)](#).

Enumerator

***VIR\_DOMAIN\_BLOCK\_COMMIT\_SHALLOW***  
***VIR\_DOMAIN\_BLOCK\_COMMIT\_DELETE***

#### 4.1.3.15 enum virDomainBlockedReason

Enumerator

***VIR\_DOMAIN\_BLOCKED\_UNKNOWN***

#### 4.1.3.16 enum virDomainBlockJobAbortFlags

virDomainBlockJobAbortFlags:

VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_ASYNC: Request only, do not wait for completion  
VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_PIVOT: Pivot to mirror when ending a copy job

Enumerator

***VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_ASYNC***  
***VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_PIVOT***

#### 4.1.3.17 enum virDomainBlockJobType

virDomainBlockJobType:

VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_PULL: Block Pull (virDomainBlockPull, or virDomainBlockRebase without flags), job ends on completion  
VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_COPY: Block Copy (virDomainBlockRebase with flags), job exists as long as mirroring is active  
VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_COMMIT: Block Commit (virDomainBlockCommit), job ends on completion

Enumerator

***VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_UNKNOWN***  
***VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_PULL***  
***VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_COPY***  
***VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_COMMIT***

#### 4.1.3.18 enum virDomainBlockRebaseFlags

virDomainBlockRebaseFlags:

Flags available for [virDomainBlockRebase\(\)](#).

Enumerator

***VIR\_DOMAIN\_BLOCK\_REBASE\_SHALLOW***  
***VIR\_DOMAIN\_BLOCK\_REBASE\_REUSE\_EXT***  
***VIR\_DOMAIN\_BLOCK\_REBASE\_COPY\_RAW***  
***VIR\_DOMAIN\_BLOCK\_REBASE\_COPY***

#### 4.1.3.19 enum virDomainBlockResizeFlags

virDomainBlockResizeFlags:

Flags available for [virDomainBlockResize\(\)](#).

Enumerator

***VIR\_DOMAIN\_BLOCK\_RESIZE\_BYTES***

#### 4.1.3.20 enum virDomainConsoleFlags

virDomainConsoleFlags

Since 0.9.10

Enumerator

***VIR\_DOMAIN\_CONSOLE\_FORCE***  
***VIR\_DOMAIN\_CONSOLE\_SAFE***

#### 4.1.3.21 enum virDomainControlState

virDomainControlState:

Current state of a control interface to the domain.

Enumerator

***VIR\_DOMAIN\_CONTROL\_OK***  
***VIR\_DOMAIN\_CONTROL\_JOB***  
***VIR\_DOMAIN\_CONTROL\_OCCUPIED***  
***VIR\_DOMAIN\_CONTROL\_ERROR***



## 4.1.3.22 enum virDomainCoreDumpFlags

Enumerator

***VIR\_DUMP\_CRASH***  
***VIR\_DUMP\_LIVE***  
***VIR\_DUMP\_BYPASS\_CACHE***  
***VIR\_DUMP\_RESET***  
***VIR\_DUMP\_MEMORY\_ONLY***

## 4.1.3.23 enum virDomainCrashedReason

Enumerator

***VIR\_DOMAIN\_CRASHED\_UNKNOWN***

## 4.1.3.24 enum virDomainCreateFlags

virDomainCreateFlags:

Flags OR'ed together to provide specific behaviour when creating a Domain.

Enumerator

***VIR\_DOMAIN\_NONE***  
***VIR\_DOMAIN\_START\_PAUSED***  
***VIR\_DOMAIN\_START\_AUTODESTROY***  
***VIR\_DOMAIN\_START\_BYPASS\_CACHE***  
***VIR\_DOMAIN\_START\_FORCE\_BOOT***

## 4.1.3.25 enum virDomainDestroyFlagsValues

virDomainDestroyFlagsValues:

Flags used to provide specific behaviour to the [virDomainDestroyFlags\(\)](#) function

Enumerator

***VIR\_DOMAIN\_DESTROY\_DEFAULT***  
***VIR\_DOMAIN\_DESTROY\_GRACEFUL***

## 4.1.3.26 enum virDomainDeviceModifyFlags

Enumerator

***VIR\_DOMAIN\_DEVICE\_MODIFY\_CURRENT***  
***VIR\_DOMAIN\_DEVICE\_MODIFY\_LIVE***  
***VIR\_DOMAIN\_DEVICE\_MODIFY\_CONFIG***  
***VIR\_DOMAIN\_DEVICE\_MODIFY\_FORCE***

#### 4.1.3.27 enum virDomainDiskErrorCode

virDomainDiskErrorCode:

Disk I/O error.

Enumerator

***VIR\_DOMAIN\_DISK\_ERROR\_NONE***  
***VIR\_DOMAIN\_DISK\_ERROR\_UNSPEC***  
***VIR\_DOMAIN\_DISK\_ERROR\_NO\_SPACE***

#### 4.1.3.28 enum virDomainEventDefinedDetailType

virDomainEventDefinedDetailType:

Details on the caused of the 'defined' lifecycle event

Enumerator

***VIR\_DOMAIN\_EVENT\_DEFINED\_ADDED***  
***VIR\_DOMAIN\_EVENT\_DEFINED\_UPDATED***

#### 4.1.3.29 enum virDomainEventGraphicsAddressType

virDomainEventGraphicsAddressType:

The type of address for the connection

Enumerator

***VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_IPV4***  
***VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_IPV6***  
***VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_UNIX***

#### 4.1.3.30 enum virDomainEventGraphicsPhase

virDomainEventGraphicsPhase:

The phase of the graphics client connection

Enumerator

***VIR\_DOMAIN\_EVENT\_GRAPHICS\_CONNECT***  
***VIR\_DOMAIN\_EVENT\_GRAPHICS\_INITIALIZE***  
***VIR\_DOMAIN\_EVENT\_GRAPHICS\_DISCONNECT***

#### 4.1.3.31 enum virDomainEventID

Enumerator

***VIR\_DOMAIN\_EVENT\_ID\_LIFECYCLE***  
***VIR\_DOMAIN\_EVENT\_ID\_REBOOT***  
***VIR\_DOMAIN\_EVENT\_ID\_RTC\_CHANGE***  
***VIR\_DOMAIN\_EVENT\_ID\_WATCHDOG***

***VIR\_DOMAIN\_EVENT\_ID\_IO\_ERROR***  
***VIR\_DOMAIN\_EVENT\_ID\_GRAPHICS***  
***VIR\_DOMAIN\_EVENT\_ID\_IO\_ERROR\_REASON***  
***VIR\_DOMAIN\_EVENT\_ID\_CONTROL\_ERROR***  
***VIR\_DOMAIN\_EVENT\_ID\_BLOCK\_JOB***  
***VIR\_DOMAIN\_EVENT\_ID\_DISK\_CHANGE***  
***VIR\_DOMAIN\_EVENT\_ID\_TRAY\_CHANGE***  
***VIR\_DOMAIN\_EVENT\_ID\_PMWAKEUP***  
***VIR\_DOMAIN\_EVENT\_ID\_PMSUSPEND***  
***VIR\_DOMAIN\_EVENT\_ID\_BALLOON\_CHANGE***

#### 4.1.3.32 enum virDomainEventIOErrorAction

virDomainEventIOErrorAction:

The action that is to be taken due to an IO error occurring

Enumerator

***VIR\_DOMAIN\_EVENT\_IO\_ERROR\_NONE***  
***VIR\_DOMAIN\_EVENT\_IO\_ERROR\_PAUSE***  
***VIR\_DOMAIN\_EVENT\_IO\_ERROR\_REPORT***

#### 4.1.3.33 enum virDomainEventPMSuspendedDetailType

virDomainEventPMSuspendedDetailType:

Details about the 'pmsuspended' lifecycle event

Enumerator

***VIR\_DOMAIN\_EVENT\_PMSUSPENDED\_MEMORY***

#### 4.1.3.34 enum virDomainEventResumedDetailType

virDomainEventResumedDetailType:

Details on the caused of the 'resumed' lifecycle event

Enumerator

***VIR\_DOMAIN\_EVENT\_RESUMED\_UNPAUSED***  
***VIR\_DOMAIN\_EVENT\_RESUMED\_MIGRATED***  
***VIR\_DOMAIN\_EVENT\_RESUMED\_FROM\_SNAPSHOT***

#### 4.1.3.35 enum virDomainEventShutdownDetailType

virDomainEventShutdownDetailType:

Details about the 'shutdown' lifecycle event

Enumerator

***VIR\_DOMAIN\_EVENT\_SHUTDOWN\_FINISHED***

#### 4.1.3.36 enum virDomainEventStartedDetailType

virDomainEventStartedDetailType:

Details on the caused of the 'started' lifecycle event

Enumerator

***VIR\_DOMAIN\_EVENT\_STARTED\_BOOTED***  
***VIR\_DOMAIN\_EVENT\_STARTED\_MIGRATED***  
***VIR\_DOMAIN\_EVENT\_STARTED\_RESTORED***  
***VIR\_DOMAIN\_EVENT\_STARTED\_FROM\_SNAPSHOT***  
***VIR\_DOMAIN\_EVENT\_STARTED\_WAKEUP***

#### 4.1.3.37 enum virDomainEventStoppedDetailType

virDomainEventStoppedDetailType:

Details on the caused of the 'stopped' lifecycle event

Enumerator

***VIR\_DOMAIN\_EVENT\_STOPPED\_SHUTDOWN***  
***VIR\_DOMAIN\_EVENT\_STOPPED\_DESTROYED***  
***VIR\_DOMAIN\_EVENT\_STOPPED\_CRASHED***  
***VIR\_DOMAIN\_EVENT\_STOPPED\_MIGRATED***  
***VIR\_DOMAIN\_EVENT\_STOPPED\_SAVED***  
***VIR\_DOMAIN\_EVENT\_STOPPED\_FAILED***  
***VIR\_DOMAIN\_EVENT\_STOPPED\_FROM\_SNAPSHOT***

#### 4.1.3.38 enum virDomainEventSuspendedDetailType

virDomainEventSuspendedDetailType:

Details on the caused of the 'suspended' lifecycle event

Enumerator

***VIR\_DOMAIN\_EVENT\_SUSPENDED\_PAUSED***  
***VIR\_DOMAIN\_EVENT\_SUSPENDED\_MIGRATED***  
***VIR\_DOMAIN\_EVENT\_SUSPENDED\_IOERROR***  
***VIR\_DOMAIN\_EVENT\_SUSPENDED\_WATCHDOG***  
***VIR\_DOMAIN\_EVENT\_SUSPENDED\_RESTORED***  
***VIR\_DOMAIN\_EVENT\_SUSPENDED\_FROM\_SNAPSHOT***

#### 4.1.3.39 enum virDomainEventTrayChangeReason

virConnectDomainEventTrayChangeReason:

The reason describing why the callback was called

Enumerator

***VIR\_DOMAIN\_EVENT\_TRAY\_CHANGE\_OPEN***  
***VIR\_DOMAIN\_EVENT\_TRAY\_CHANGE\_CLOSE***

#### 4.1.3.40 enum virDomainEventType

virDomainEventType:

a virDomainEventType is emitted during domain lifecycle events

Enumerator

***VIR\_DOMAIN\_EVENT\_DEFINED***  
***VIR\_DOMAIN\_EVENT\_UNDEFINED***  
***VIR\_DOMAIN\_EVENT\_STARTED***  
***VIR\_DOMAIN\_EVENT\_SUSPENDED***  
***VIR\_DOMAIN\_EVENT\_RESUMED***  
***VIR\_DOMAIN\_EVENT\_STOPPED***  
***VIR\_DOMAIN\_EVENT\_SHUTDOWN***  
***VIR\_DOMAIN\_EVENT\_PMSUSPENDED***

#### 4.1.3.41 enum virDomainEventUndefinedDetailType

virDomainEventUndefinedDetailType:

Details on the caused of the 'undefined' lifecycle event

Enumerator

***VIR\_DOMAIN\_EVENT\_UNDEFINED\_REMOVED***

#### 4.1.3.42 enum virDomainEventWatchdogAction

virDomainEventWatchdogAction:

The action that is to be taken due to the watchdog device firing

Enumerator

***VIR\_DOMAIN\_EVENT\_WATCHDOG\_NONE***  
***VIR\_DOMAIN\_EVENT\_WATCHDOG\_PAUSE***  
***VIR\_DOMAIN\_EVENT\_WATCHDOG\_RESET***  
***VIR\_DOMAIN\_EVENT\_WATCHDOG\_POWEROFF***  
***VIR\_DOMAIN\_EVENT\_WATCHDOG\_SHUTDOWN***  
***VIR\_DOMAIN\_EVENT\_WATCHDOG\_DEBUG***

#### 4.1.3.43 enum virDomainJobType

Enumerator

***VIR\_DOMAIN\_JOB\_NONE***  
***VIR\_DOMAIN\_JOB\_BOUNDED***  
***VIR\_DOMAIN\_JOB\_UNBOUNDED***  
***VIR\_DOMAIN\_JOB\_COMPLETED***  
***VIR\_DOMAIN\_JOB\_FAILED***  
***VIR\_DOMAIN\_JOB\_CANCELLED***

## 4.1.3.44 enum virDomainMemoryFlags

Enumerator

*VIR\_MEMORY\_VIRTUAL*  
*VIR\_MEMORY\_PHYSICAL*

## 4.1.3.45 enum virDomainMemoryModFlags

Enumerator

*VIR\_DOMAIN\_MEM\_CURRENT*  
*VIR\_DOMAIN\_MEM\_LIVE*  
*VIR\_DOMAIN\_MEM\_CONFIG*  
*VIR\_DOMAIN\_MEM\_MAXIMUM*

## 4.1.3.46 enum virDomainMemoryStatTags

Memory Statistics Tags:

Enumerator

*VIR\_DOMAIN\_MEMORY\_STAT\_SWAP\_IN*  
*VIR\_DOMAIN\_MEMORY\_STAT\_SWAP\_OUT*  
*VIR\_DOMAIN\_MEMORY\_STAT\_MAJOR\_FAULT*  
*VIR\_DOMAIN\_MEMORY\_STAT\_MINOR\_FAULT*  
*VIR\_DOMAIN\_MEMORY\_STAT\_UNUSED*  
*VIR\_DOMAIN\_MEMORY\_STAT\_AVAILABLE*  
*VIR\_DOMAIN\_MEMORY\_STAT\_ACTUAL\_BALLOON*  
*VIR\_DOMAIN\_MEMORY\_STAT\_RSS*  
*VIR\_DOMAIN\_MEMORY\_STAT\_NR*

## 4.1.3.47 enum virDomainMetadataType

Enumerator

*VIR\_DOMAIN\_METADATA\_DESCRIPTION*  
*VIR\_DOMAIN\_METADATA\_TITLE*  
*VIR\_DOMAIN\_METADATA\_ELEMENT*

## 4.1.3.48 enum virDomainMigrateFlags

Enumerator

*VIR\_MIGRATE\_LIVE*  
*VIR\_MIGRATE\_PEER2PEER*  
*VIR\_MIGRATE\_TUNNELLED*  
*VIR\_MIGRATE\_PERSIST\_DEST*  
*VIR\_MIGRATE\_UNDEFINE\_SOURCE*  
*VIR\_MIGRATE\_PAUSED*

***VIR\_MIGRATE\_NON\_SHARED\_DISK***  
***VIR\_MIGRATE\_NON\_SHARED\_INC***  
***VIR\_MIGRATE\_CHANGE\_PROTECTION***  
***VIR\_MIGRATE\_UNSAFE***

#### 4.1.3.49 enum virDomainModificationImpact

virDomainModificationImpact:

Several modification APIs take flags to determine whether a change to the domain affects just the running instance, just the persistent definition, or both at the same time. The counterpart query APIs also take the same flags to determine whether to query the running instance or persistent definition, although both cannot be queried at once.

The use of VIR\_DOMAIN\_AFFECT\_CURRENT will resolve to either VIR\_DOMAIN\_AFFECT\_LIVE or VIR\_DOMAIN\_AFFECT\_CONFIG according to current domain state. VIR\_DOMAIN\_AFFECT\_LIVE requires a running domain, and VIR\_DOMAIN\_AFFECT\_CONFIG requires a persistent domain (whether or not it is running).

These enums should not conflict with those of virTypedParameterFlags.

Enumerator

***VIR\_DOMAIN\_AFFECT\_CURRENT***  
***VIR\_DOMAIN\_AFFECT\_LIVE***  
***VIR\_DOMAIN\_AFFECT\_CONFIG***

#### 4.1.3.50 enum virDomainNostateReason

Enumerator

***VIR\_DOMAIN\_NOSTATE\_UNKNOWN***

#### 4.1.3.51 enum virDomainNumatuneMemMode

virDomainNumatuneMemMode: Representation of the various modes in the <numatune> element of a domain.

Enumerator

***VIR\_DOMAIN\_NUMATUNE\_MEM\_STRICT***  
***VIR\_DOMAIN\_NUMATUNE\_MEM\_PREFERRED***  
***VIR\_DOMAIN\_NUMATUNE\_MEM\_INTERLEAVE***

#### 4.1.3.52 enum virDomainOpenGraphicsFlags

Enumerator

***VIR\_DOMAIN\_OPEN\_GRAPHICS\_SKIPAUTH***

#### 4.1.3.53 enum virDomainPausedReason

Enumerator

***VIR\_DOMAIN\_PAUSED\_UNKNOWN***  
***VIR\_DOMAIN\_PAUSED\_USER***

***VIR\_DOMAIN\_PAUSED\_MIGRATION***  
***VIR\_DOMAIN\_PAUSED\_SAVE***  
***VIR\_DOMAIN\_PAUSED\_DUMP***  
***VIR\_DOMAIN\_PAUSED\_IOERROR***  
***VIR\_DOMAIN\_PAUSED\_WATCHDOG***  
***VIR\_DOMAIN\_PAUSED\_FROM\_SNAPSHOT***  
***VIR\_DOMAIN\_PAUSED\_SHUTTING\_DOWN***

#### 4.1.3.54 enum virDomainPMSuspendedReason

Enumerator

***VIR\_DOMAIN\_PMSUSPENDED\_UNKNOWN***

#### 4.1.3.55 enum virDomainRebootFlagValues

Enumerator

***VIR\_DOMAIN\_REBOOT\_DEFAULT***  
***VIR\_DOMAIN\_REBOOT\_ACPI\_POWER\_BTN***  
***VIR\_DOMAIN\_REBOOT\_GUEST\_AGENT***

#### 4.1.3.56 enum virDomainRunningReason

Enumerator

***VIR\_DOMAIN\_RUNNING\_UNKNOWN***  
***VIR\_DOMAIN\_RUNNING\_BOOTED***  
***VIR\_DOMAIN\_RUNNING\_MIGRATED***  
***VIR\_DOMAIN\_RUNNING\_RESTORED***  
***VIR\_DOMAIN\_RUNNING\_FROM\_SNAPSHOT***  
***VIR\_DOMAIN\_RUNNING\_UNPAUSED***  
***VIR\_DOMAIN\_RUNNING\_MIGRATION\_CANCELED***  
***VIR\_DOMAIN\_RUNNING\_SAVE\_CANCELED***  
***VIR\_DOMAIN\_RUNNING\_WAKEUP***

#### 4.1.3.57 enum virDomainSaveRestoreFlags

virDomainSaveRestoreFlags: Flags for use in [virDomainSaveFlags\(\)](#), [virDomainManagedSave\(\)](#), [virDomainRestoreFlags\(\)](#), and [virDomainSaveImageDefineXML\(\)](#). Not all flags apply to all these functions.

Enumerator

***VIR\_DOMAIN\_SAVE\_BYPASS\_CACHE***  
***VIR\_DOMAIN\_SAVE\_RUNNING***  
***VIR\_DOMAIN\_SAVE\_PAUSED***



## 4.1.3.58 enum virDomainShutdownFlagValues

Enumerator

*VIR\_DOMAIN\_SHUTDOWN\_DEFAULT*  
*VIR\_DOMAIN\_SHUTDOWN\_ACPI\_POWER\_BTN*  
*VIR\_DOMAIN\_SHUTDOWN\_GUEST\_AGENT*

## 4.1.3.59 enum virDomainShutdownReason

Enumerator

*VIR\_DOMAIN\_SHUTDOWN\_UNKNOWN*  
*VIR\_DOMAIN\_SHUTDOWN\_USER*

## 4.1.3.60 enum virDomainShutoffReason

Enumerator

*VIR\_DOMAIN\_SHUTOFF\_UNKNOWN*  
*VIR\_DOMAIN\_SHUTOFF\_SHUTDOWN*  
*VIR\_DOMAIN\_SHUTOFF\_DESTROYED*  
*VIR\_DOMAIN\_SHUTOFF\_CRASHED*  
*VIR\_DOMAIN\_SHUTOFF\_MIGRATED*  
*VIR\_DOMAIN\_SHUTOFF\_SAVED*  
*VIR\_DOMAIN\_SHUTOFF\_FAILED*  
*VIR\_DOMAIN\_SHUTOFF\_FROM\_SNAPSHOT*

## 4.1.3.61 enum virDomainSnapshotCreateFlags

Enumerator

*VIR\_DOMAIN\_SNAPSHOT\_CREATE\_REDEFINE*  
*VIR\_DOMAIN\_SNAPSHOT\_CREATE\_CURRENT*  
*VIR\_DOMAIN\_SNAPSHOT\_CREATE\_NO\_METADATA*  
*VIR\_DOMAIN\_SNAPSHOT\_CREATE\_HALT*  
*VIR\_DOMAIN\_SNAPSHOT\_CREATE\_DISK\_ONLY*  
*VIR\_DOMAIN\_SNAPSHOT\_CREATE\_REUSE\_EXT*  
*VIR\_DOMAIN\_SNAPSHOT\_CREATE\_QUIESCE*  
*VIR\_DOMAIN\_SNAPSHOT\_CREATE\_ATOMIC*

## 4.1.3.62 enum virDomainSnapshotDeleteFlags

Enumerator

*VIR\_DOMAIN\_SNAPSHOT\_DELETE\_CHILDREN*  
*VIR\_DOMAIN\_SNAPSHOT\_DELETE\_METADATA\_ONLY*  
*VIR\_DOMAIN\_SNAPSHOT\_DELETE\_CHILDREN\_ONLY*

#### 4.1.3.63 enum virDomainSnapshotListFlags

virDomainSnapshotListFlags:

Flags valid for [virDomainSnapshotNum\(\)](#), [virDomainSnapshotListNames\(\)](#), [virDomainSnapshotNumChildren\(\)](#), and [virDomainSnapshotListChildrenNames\(\)](#), [virDomainListAllSnapshots\(\)](#), and [virDomainSnapshotListAllChildren\(\)](#). Note that the interpretation of flag (1<<0) depends on which function it is passed to; but serves to toggle the per-call default of whether the listing is shallow or recursive. Remaining bits come in groups; if all bits from a group are 0, then that group is not used to filter results.

Enumerator

***VIR\_DOMAIN\_SNAPSHOT\_LIST\_ROOTS***  
***VIR\_DOMAIN\_SNAPSHOT\_LIST\_DESCENDANTS***  
***VIR\_DOMAIN\_SNAPSHOT\_LIST\_LEAVES***  
***VIR\_DOMAIN\_SNAPSHOT\_LIST\_NO\_LEAVES***  
***VIR\_DOMAIN\_SNAPSHOT\_LIST\_METADATA***  
***VIR\_DOMAIN\_SNAPSHOT\_LIST\_NO\_METADATA***

#### 4.1.3.64 enum virDomainSnapshotRevertFlags

Enumerator

***VIR\_DOMAIN\_SNAPSHOT\_REVERT\_RUNNING***  
***VIR\_DOMAIN\_SNAPSHOT\_REVERT\_PAUSED***  
***VIR\_DOMAIN\_SNAPSHOT\_REVERT\_FORCE***

#### 4.1.3.65 enum virDomainState

virDomainState:

A domain may be in different states at a given point in time

Enumerator

***VIR\_DOMAIN\_NOSTATE***  
***VIR\_DOMAIN\_RUNNING***  
***VIR\_DOMAIN\_BLOCKED***  
***VIR\_DOMAIN\_PAUSED***  
***VIR\_DOMAIN\_SHUTDOWN***  
***VIR\_DOMAIN\_SHUTOFF***  
***VIR\_DOMAIN\_CRASHED***  
***VIR\_DOMAIN\_PMSUSPENDED***

#### 4.1.3.66 enum virDomainUndefineFlagsValues

Enumerator

***VIR\_DOMAIN\_UNDEFINE\_MANAGED\_SAVE***  
***VIR\_DOMAIN\_UNDEFINE\_SNAPSHOTS\_METADATA***

## 4.1.3.67 enum virDomainVcpuFlags

Enumerator

***VIR\_DOMAIN\_VCPU\_CURRENT***  
***VIR\_DOMAIN\_VCPU\_LIVE***  
***VIR\_DOMAIN\_VCPU\_CONFIG***  
***VIR\_DOMAIN\_VCPU\_MAXIMUM***

## 4.1.3.68 enum virDomainXMLFlags

virDomainXMLFlags:

Flags available for virDomainGetXMLDesc

Enumerator

***VIR\_DOMAIN\_XML\_SECURE***  
***VIR\_DOMAIN\_XML\_INACTIVE***  
***VIR\_DOMAIN\_XML\_UPDATE\_CPU***

## 4.1.3.69 enum virEventHandleType

virEventHandleType:

a virEventHandleType is used similar to POLLxxx FD events, but is specific to libvirt. A client app must translate to, and from POLL events when using this construct.

Enumerator

***VIR\_EVENT\_HANDLE\_READABLE***  
***VIR\_EVENT\_HANDLE\_WRITABLE***  
***VIR\_EVENT\_HANDLE\_ERROR***  
***VIR\_EVENT\_HANDLE\_HANGUP***

## 4.1.3.70 enum virInterfaceXMLFlags

Enumerator

***VIR\_INTERFACE\_XML\_INACTIVE***

## 4.1.3.71 enum virKeycodeSet

virKeycodeSet:

Enum to specify which keycode mapping is in use for [virDomainSendKey\(\)](#).

Enumerator

***VIR\_KEYCODE\_SET\_LINUX***  
***VIR\_KEYCODE\_SET\_XT***  
***VIR\_KEYCODE\_SET\_ATSET1***  
***VIR\_KEYCODE\_SET\_ATSET2***

***VIR\_KEYCODE\_SET\_ATSET3***  
***VIR\_KEYCODE\_SET\_OSX***  
***VIR\_KEYCODE\_SET\_XT\_KBD***  
***VIR\_KEYCODE\_SET\_USB***  
***VIR\_KEYCODE\_SET\_WIN32***  
***VIR\_KEYCODE\_SET\_RFB***

#### 4.1.3.72 enum virMemoryParameterType

virMemoryParameterType:

A memory parameter field type. Provided for backwards compatibility; virTypedParameterType is the preferred enum since 0.9.2.

Enumerator

***VIR\_DOMAIN\_MEMORY\_PARAM\_INT***  
***VIR\_DOMAIN\_MEMORY\_PARAM\_UINT***  
***VIR\_DOMAIN\_MEMORY\_PARAM\_LLONG***  
***VIR\_DOMAIN\_MEMORY\_PARAM\_ULLONG***  
***VIR\_DOMAIN\_MEMORY\_PARAM\_DOUBLE***  
***VIR\_DOMAIN\_MEMORY\_PARAM\_BOOLEAN***

#### 4.1.3.73 enum virNetworkUpdateCommand

virNetworkUpdateCommand:

describes which type of update to perform on a <network> definition.

Enumerator

***VIR\_NETWORK\_UPDATE\_COMMAND\_NONE***  
***VIR\_NETWORK\_UPDATE\_COMMAND\_MODIFY***  
***VIR\_NETWORK\_UPDATE\_COMMAND\_DELETE***  
***VIR\_NETWORK\_UPDATE\_COMMAND\_ADD\_LAST***  
***VIR\_NETWORK\_UPDATE\_COMMAND\_ADD\_FIRST***

#### 4.1.3.74 enum virNetworkUpdateFlags

virNetworkUpdateFlags:

Flags to control options for [virNetworkUpdate\(\)](#)

Enumerator

***VIR\_NETWORK\_UPDATE\_AFFECT\_CURRENT***  
***VIR\_NETWORK\_UPDATE\_AFFECT\_LIVE***  
***VIR\_NETWORK\_UPDATE\_AFFECT\_CONFIG***

## 4.1.3.75 enum virNetworkUpdateSection

virNetworkUpdateSection:

describes which section of a <network> definition the provided xml should be applied to.

Enumerator

***VIR\_NETWORK\_SECTION\_NONE***  
***VIR\_NETWORK\_SECTION\_BRIDGE***  
***VIR\_NETWORK\_SECTION\_DOMAIN***  
***VIR\_NETWORK\_SECTION\_IP***  
***VIR\_NETWORK\_SECTION\_IP\_DHCP\_HOST***  
***VIR\_NETWORK\_SECTION\_IP\_DHCP\_RANGE***  
***VIR\_NETWORK\_SECTION\_FORWARD***  
***VIR\_NETWORK\_SECTION\_FORWARD\_INTERFACE***  
***VIR\_NETWORK\_SECTION\_FORWARD\_PF***  
***VIR\_NETWORK\_SECTION\_PORTGROUP***  
***VIR\_NETWORK\_SECTION\_DNS\_HOST***  
***VIR\_NETWORK\_SECTION\_DNS\_TXT***  
***VIR\_NETWORK\_SECTION\_DNS\_SRV***  
***VIR\_NETWORK\_SECTION\_TUNNEL*** **POL New**

## 4.1.3.76 enum virNetworkXMLFlags

Enumerator

***VIR\_NETWORK\_XML\_INACTIVE***

## 4.1.3.77 enum virNodeGetCPUStatsAllCPUs

VIR\_NODE\_CPU\_STATS\_ALL\_CPUS:

Value for specifying request for the total CPU time/utilization

Enumerator

***VIR\_NODE\_CPU\_STATS\_ALL\_CPUS***

## 4.1.3.78 enum virNodeGetMemoryStatsAllCells

VIR\_NODE\_MEMORY\_STATS\_ALL\_CELLS:

Value for specifying request for the total memory of all cells.

Enumerator

***VIR\_NODE\_MEMORY\_STATS\_ALL\_CELLS***

#### 4.1.3.79 enum virNodeSuspendTarget

virNodeSuspendTarget:

Flags to indicate which system-wide sleep state the host must be transitioned to.

Enumerator

***VIR\_NODE\_SUSPEND\_TARGET\_MEM***  
***VIR\_NODE\_SUSPEND\_TARGET\_DISK***  
***VIR\_NODE\_SUSPEND\_TARGET\_HYBRID***

#### 4.1.3.80 enum virSchedParameterType

virSchedParameterType:

A scheduler parameter field type. Provided for backwards compatibility; virTypedParameterType is the preferred enum since 0.9.2.

Enumerator

***VIR\_DOMAIN\_SCHED\_FIELD\_INT***  
***VIR\_DOMAIN\_SCHED\_FIELD\_UINT***  
***VIR\_DOMAIN\_SCHED\_FIELD\_LLONG***  
***VIR\_DOMAIN\_SCHED\_FIELD\_ULLONG***  
***VIR\_DOMAIN\_SCHED\_FIELD\_DOUBLE***  
***VIR\_DOMAIN\_SCHED\_FIELD\_BOOLEAN***

#### 4.1.3.81 enum virSecretUsageType

Enumerator

***VIR\_SECRET\_USAGE\_TYPE\_NONE***  
***VIR\_SECRET\_USAGE\_TYPE\_VOLUME***  
***VIR\_SECRET\_USAGE\_TYPE\_CEPH***

#### 4.1.3.82 enum virStoragePoolBuildFlags

Enumerator

***VIR\_STORAGE\_POOL\_BUILD\_NEW***  
***VIR\_STORAGE\_POOL\_BUILD\_REPAIR***  
***VIR\_STORAGE\_POOL\_BUILD\_RESIZE***  
***VIR\_STORAGE\_POOL\_BUILD\_NO\_OVERWRITE***  
***VIR\_STORAGE\_POOL\_BUILD\_OVERWRITE***

#### 4.1.3.83 enum virStoragePoolDeleteFlags

Enumerator

***VIR\_STORAGE\_POOL\_DELETE\_NORMAL***  
***VIR\_STORAGE\_POOL\_DELETE\_ZEROED***

## 4.1.3.84 enum virStoragePoolState

Enumerator

***VIR\_STORAGE\_POOL\_INACTIVE***  
***VIR\_STORAGE\_POOL\_BUILDING***  
***VIR\_STORAGE\_POOL\_RUNNING***  
***VIR\_STORAGE\_POOL\_DEGRADED***  
***VIR\_STORAGE\_POOL\_INACCESSIBLE***

## 4.1.3.85 enum virStorageVolDeleteFlags

Enumerator

***VIR\_STORAGE\_VOL\_DELETE\_NORMAL***  
***VIR\_STORAGE\_VOL\_DELETE\_ZEROED***

## 4.1.3.86 enum virStorageVolResizeFlags

Enumerator

***VIR\_STORAGE\_VOL\_RESIZE\_ALLOCATE***  
***VIR\_STORAGE\_VOL\_RESIZE\_DELTA***  
***VIR\_STORAGE\_VOL\_RESIZE\_SHRINK***

## 4.1.3.87 enum virStorageVolType

Enumerator

***VIR\_STORAGE\_VOL\_FILE***  
***VIR\_STORAGE\_VOL\_BLOCK***  
***VIR\_STORAGE\_VOL\_DIR***  
***VIR\_STORAGE\_VOL\_NETWORK***

## 4.1.3.88 enum virStorageVolWipeAlgorithm

Enumerator

***VIR\_STORAGE\_VOL\_WIPE\_ALG\_ZERO***  
***VIR\_STORAGE\_VOL\_WIPE\_ALG\_NNSA***  
***VIR\_STORAGE\_VOL\_WIPE\_ALG\_DOD***  
***VIR\_STORAGE\_VOL\_WIPE\_ALG\_BSI***  
***VIR\_STORAGE\_VOL\_WIPE\_ALG\_GUTMANN***  
***VIR\_STORAGE\_VOL\_WIPE\_ALG\_SCHNEIER***  
***VIR\_STORAGE\_VOL\_WIPE\_ALG\_PFITZNER7***  
***VIR\_STORAGE\_VOL\_WIPE\_ALG\_PFITZNER33***  
***VIR\_STORAGE\_VOL\_WIPE\_ALG\_RANDOM***

#### 4.1.3.89 enum virStorageXMLFlags

Enumerator

***VIR\_STORAGE\_XML\_INACTIVE***

#### 4.1.3.90 enum virStreamEventType

Enumerator

***VIR\_STREAM\_EVENT\_READABLE***

***VIR\_STREAM\_EVENT\_WRITABLE***

***VIR\_STREAM\_EVENT\_ERROR***

***VIR\_STREAM\_EVENT\_HANGUP***

#### 4.1.3.91 enum virStreamFlags

Enumerator

***VIR\_STREAM\_NONBLOCK***

#### 4.1.3.92 enum virTypedParameterFlags

virTypedParameterFlags:

Flags related to libvirt APIs that use virTypedParameter.

These enums should not conflict with those of virDomainModificationImpact.

Enumerator

***VIR\_TYPED\_PARAM\_STRING\_OKAY***

#### 4.1.3.93 enum virTypedParameterType

virTypedParameterType:

Express the type of a virTypedParameter

Enumerator

***VIR\_TYPED\_PARAM\_INT***

***VIR\_TYPED\_PARAM\_UINT***

***VIR\_TYPED\_PARAM\_LLONG***

***VIR\_TYPED\_PARAM\_ULLONG***

***VIR\_TYPED\_PARAM\_DOUBLE***

***VIR\_TYPED\_PARAM\_BOOLEAN***

***VIR\_TYPED\_PARAM\_STRING***



## 4.1.3.94 enum virVcpuState

virVcpuInfo: structure for information about a virtual CPU in a domain.

Enumerator

***VIR\_VCPU\_OFFLINE***  
***VIR\_VCPU\_RUNNING***  
***VIR\_VCPU\_BLOCKED***

## 4.1.4 Function Documentation

## 4.1.4.1 char\* virConnectBaselineCPU ( virConnectPtr conn, const char \*\* xmlCPUs, unsigned int ncpus, unsigned int flags )

virConnectBaselineCPU:

: virConnect connection : number of CPUs in xmlCPUs : array of XML descriptions of host CPUs : fine-tuning flags

Computes the most feature-rich CPU which is compatible with all given host CPUs.

Returns XML description of the computed CPU or NULL on error.

virConnectBaselineCPU:

: virConnect connection : array of XML descriptions of host CPUs : number of CPUs in xmlCPUs : extra flags; not used yet, so callers should always pass 0

Computes the most feature-rich CPU which is compatible with all given host CPUs.

Returns XML description of the computed CPU or NULL on error.

## 4.1.4.2 int virConnectClose ( virConnectPtr conn )

virConnectClose: : pointer to the hypervisor connection

This function closes the connection to the Hypervisor. This should not be called if further interaction with the Hypervisor are needed especially if there is running domain which need further monitoring by the application.

Connections are reference counted; the count is explicitly increased by the initial open (virConnectOpen, virConnectOpenAuth, and the like) as well as virConnectRef; it is also temporarily increased by other API that depend on the connection remaining alive. The open and every virConnectRef call should have a matching virConnectClose, and all other references will be released after the corresponding operation completes.

Returns a positive number if at least 1 reference remains on success. The returned value should not be assumed to be the total reference count. A return of 0 implies no references remain and the connection is closed and memory has been freed. A return of -1 implies a failure.

It is possible for the last virConnectClose to return a positive value if some other object still has a temporary reference to the connection, but the application should not try to further use a connection after the virConnectClose that matches the initial open.

## 4.1.4.3 int virConnectCompareCPU ( virConnectPtr conn, const char \* xmlDesc, unsigned int flags )

virConnectCompareCPU:

: virConnect connection : XML description of either guest or host cpu with <cpu> root tag : comparison flags

Compares given CPU with host cpu.

Returns virCPUCompareResult.

virConnectCompareCPU: : virConnect connection : XML describing the CPU to compare with host CPU : extra flags; not used yet, so callers should always pass 0

Compares the given CPU description with the host CPU

Returns comparison result according to enum `virCPUCompareResult`

#### 4.1.4.4 `int virConnectDomainEventDeregister ( virConnectPtr conn, virConnectDomainEventCallback cb )`

`virConnectDomainEventDeregister`: : pointer to the connection : callback to the function handling domain events

Removes a callback previously registered with the `virConnectDomainEventRegister` function.

Use of this method is no longer recommended. Instead applications should try `virConnectDomainEventUnregisterAny` which has a more flexible API contract

Returns 0 on success, -1 on failure

#### 4.1.4.5 `int virConnectDomainEventDeregisterAny ( virConnectPtr conn, int callbackID )`

`virConnectDomainEventDeregisterAny`: : pointer to the connection : the callback identifier

Removes an event callback. The `callbackID` parameter should be the value obtained from a previous `virDomainEventRegisterAny` method.

Returns 0 on success, -1 on failure

#### 4.1.4.6 `int virConnectDomainEventRegister ( virConnectPtr conn, virConnectDomainEventCallback cb, void * opaque, virFreeCallback freecb )`

`virConnectDomainEventRegister`: : pointer to the connection : callback to the function handling domain events : opaque data to pass on to the callback : optional function to deallocate opaque when not used anymore

Adds a callback to receive notifications of domain lifecycle events occurring on a connection

Use of this method is no longer recommended. Instead applications should try `virConnectDomainEventRegisterAny` which has a more flexible API contract

The `virDomainPtr` object handle passed into the callback upon delivery of an event is only valid for the duration of execution of the callback. If the callback wishes to keep the domain object after the callback returns, it shall take a reference to it, by calling `virDomainRef`. The reference can be released once the object is no longer required by calling `virDomainFree`.

Returns 0 on success, -1 on failure

#### 4.1.4.7 `int virConnectDomainEventRegisterAny ( virConnectPtr conn, virDomainPtr dom, int eventID, virConnectDomainEventGenericCallback cb, void * opaque, virFreeCallback freecb )`

`virConnectDomainEventRegisterAny`: : pointer to the connection : pointer to the domain : the event type to receive : callback to the function handling domain events : opaque data to pass on to the callback : optional function to deallocate opaque when not used anymore

Adds a callback to receive notifications of arbitrary domain events occurring on a domain.

If `dom` is `NULL`, then events will be monitored for any domain. If `dom` is non-`NULL`, then only the specific domain will be monitored

Most types of event have a callback providing a custom set of parameters for the event. When registering an event, it is thus necessary to use the `VIR_DOMAIN_EVENT_CALLBACK()` macro to cast the supplied function pointer to match the signature of this method.

The `virDomainPtr` object handle passed into the callback upon delivery of an event is only valid for the duration of execution of the callback. If the callback wishes to keep the domain object after the callback returns, it shall take a reference to it, by calling `virDomainRef`. The reference can be released once the object is no longer required by calling `virDomainFree`.

The return value from this method is a positive integer identifier for the callback. To unregister a callback, this callback ID should be passed to the `virDomainEventUnregisterAny` method

Returns a callback identifier on success, -1 on failure

**4.1.4.8** `char* virConnectDomainXMLFromNative ( virConnectPtr conn, const char * nativeFormat, const char * nativeConfig, unsigned int flags )`

`virConnectDomainXMLFromNative`: : a connection object : configuration format importing from : the configuration data to import : extra flags; not used yet, so callers should always pass 0

Reads native configuration data describing a domain, and generates libvirt domain XML. The format of the native data is hypervisor dependant.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must `free()` the returned value.

**4.1.4.9** `char* virConnectDomainXMLToNative ( virConnectPtr conn, const char * nativeFormat, const char * domainXml, unsigned int flags )`

`virConnectDomainXMLToNative`: : a connection object : configuration format exporting to : the domain configuration to export : extra flags; not used yet, so callers should always pass 0

Reads a domain XML configuration document, and generates a native configuration file describing the domain. The format of the native data is hypervisor dependant.

Returns a 0 terminated UTF-8 encoded native config datafile, or NULL in case of error. the caller must `free()` the returned value.

**4.1.4.10** `char* virConnectFindStoragePoolSources ( virConnectPtr conn, const char * type, const char * srcSpec, unsigned int flags )`

`virConnectFindStoragePoolSources`: : pointer to hypervisor connection : type of storage pool sources to discover : XML document specifying discovery source : extra flags; not used yet, so callers should always pass 0

Talks to a storage backend and attempts to auto-discover the set of available storage pool sources. e.g. For iSCSI this would be a set of iSCSI targets. For NFS this would be a list of exported paths. The `srcSpec` (optional for some storage pool types, e.g. local ones) is an instance of the storage pool's source element specifying where to look for the pools.

`srcSpec` is not required for some types (e.g., those querying local storage resources only)

Returns an xml document consisting of a `SourceList` element containing a source document appropriate to the given pool type for each discovered source.

**4.1.4.11** `char* virConnectGetCapabilities ( virConnectPtr conn )`

`virConnectGetCapabilities`: : pointer to the hypervisor connection

Provides capabilities of the hypervisor / driver.

Returns NULL in case of error, or an XML string defining the capabilities. The client must `free` the returned string after use.

**4.1.4.12** `char* virConnectGetHostname ( virConnectPtr conn )`

`virConnectGetHostname`: : pointer to a hypervisor connection

This returns the system hostname on which the hypervisor is running (the result of the `gethostname` system call). If we are connected to a remote system, then this returns the hostname of the remote system.

Returns the hostname which must be freed by the caller, or NULL if there was an error.

#### 4.1.4.13 `int virConnectGetLibVersion ( virConnectPtr conn, unsigned long * libVer )`

`virConnectGetLibVersion`: : pointer to the hypervisor connection : returns the libvirt library version used on the connection (OUT)

Provides , which is the version of libvirt used by the daemon running on the host

Returns -1 in case of failure, 0 otherwise, and values for have the format major \* 1,000,000 + minor \* 1,000 + release.

#### 4.1.4.14 `int virConnectGetMaxVcpus ( virConnectPtr conn, const char * type )`

`virConnectGetMaxVcpus`: : pointer to the hypervisor connection : value of the 'type' attribute in the <domain> element

Provides the maximum number of virtual CPUs supported for a guest VM of a specific type. The 'type' parameter here corresponds to the 'type' attribute in the <domain> element of the XML.

Returns the maximum of virtual CPU or -1 in case of error.

#### 4.1.4.15 `char* virConnectGetSysinfo ( virConnectPtr conn, unsigned int flags )`

`virConnectGetSysinfo`: : pointer to a hypervisor connection : extra flags; not used yet, so callers should always pass 0

This returns the XML description of the sysinfo details for the host on which the hypervisor is running, in the same format as the <sysinfo> element of a domain XML. This information is generally available only for hypervisors running with root privileges.

Returns the XML string which must be freed by the caller, or NULL if there was an error.

#### 4.1.4.16 `const char* virConnectGetType ( virConnectPtr conn )`

`virConnectGetType`: : pointer to the hypervisor connection

Get the name of the Hypervisor software used.

Returns NULL in case of error, a static zero terminated string otherwise.

See also: <http://www.redhat.com/archives/libvir-list/2007-February/msg00096.-html>

#### 4.1.4.17 `char* virConnectGetURI ( virConnectPtr conn )`

`virConnectGetURI`: : pointer to a hypervisor connection

This returns the URI (name) of the hypervisor connection. Normally this is the same as or similar to the string passed to the `virConnectOpen/virConnectOpenReadOnly` call, but the driver may make the URI canonical. If name == NULL was passed to `virConnectOpen`, then the driver will return a non-NULL URI which can be used to connect to the same hypervisor later.

Returns the URI string which must be freed by the caller, or NULL if there was an error.

#### 4.1.4.18 `int virConnectGetVersion ( virConnectPtr conn, unsigned long * hvVer )`

`virConnectGetVersion`: : pointer to the hypervisor connection : return value for the version of the running hypervisor (OUT)

Get the version level of the Hypervisor running. This may work only with hypervisor call, i.e. with privileged access to the hypervisor, not with a Read-Only connection.

Returns -1 in case of error, 0 otherwise. if the version can't be extracted by lack of capacities returns 0 and is 0, otherwise value is  $\text{major} * 1,000,000 + \text{minor} * 1,000 + \text{release}$

#### 4.1.4.19 int virConnectIsAlive ( virConnectPtr conn )

virConnectIsAlive: : pointer to the connection object

Determine if the connection to the hypervisor is still alive

A connection will be classed as alive if it is either local, or running over a channel (TCP or UNIX socket) which is not closed.

Returns 1 if alive, 0 if dead, -1 on error

#### 4.1.4.20 int virConnectIsEncrypted ( virConnectPtr conn )

virConnectIsEncrypted: : pointer to the connection object

Determine if the connection to the hypervisor is encrypted

Returns 1 if encrypted, 0 if not encrypted, -1 on error

#### 4.1.4.21 int virConnectIsSecure ( virConnectPtr conn )

virConnectIsSecure: : pointer to the connection object

Determine if the connection to the hypervisor is secure

A connection will be classed as secure if it is either encrypted, or running over a channel which is not exposed to eavesdropping (eg a UNIX domain socket, or pipe)

Returns 1 if secure, 0 if not secure, -1 on error

#### 4.1.4.22 int virConnectListAllDomains ( virConnectPtr conn, virDomainPtr \*\* domains, unsigned int flags )

virConnectListAllDomains: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing domain objects or NULL if the list is not required (just returns number of guests). : bitwise-OR of virConnectListAllDomainsFlags

Collect a possibly-filtered list of all domains, and return an allocated array of information for each. This API solves the race inherent in [virConnectListDomains\(\)](#) and [virConnectListDefinedDomains\(\)](#).

Normally, all domains are returned; however, can be used to filter the results for a smaller list of targeted domains. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a domain, and where all bits within a group describe all possible domains. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction (for example, not all hypervisors can tell whether domains have snapshots). For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination (such as an inactive transient domain), in that case a hypervisor may return either 0 or an error.

The first group of is VIR\_CONNECT\_LIST\_DOMAINS\_ACTIVE (online domains) and VIR\_CONNECT\_LIST\_DOMAINS\_INACTIVE (offline domains).

The next group of is VIR\_CONNECT\_LIST\_DOMAINS\_PERSISTENT (defined domains) and VIR\_CONNECT\_LIST\_DOMAINS\_TRANSIENT (running but not defined).

The next group of covers various domain states: VIR\_CONNECT\_LIST\_DOMAINS\_RUNNING, VIR\_CONNECT\_LIST\_DOMAINS\_PAUSED, VIR\_CONNECT\_LIST\_DOMAINS\_SHUTOFF, and a catch-all for all other states (such as crashed, this catch-all covers the possibility of adding new states).

The remaining groups cover boolean attributes commonly asked about domains; they include `VIR_CONNECT_LIST_DOMAINS_MANAGEDSAVE` and `VIR_CONNECT_LIST_DOMAINS_NO_MANAGEDSAVE`, for filtering based on whether a managed save image exists; `VIR_CONNECT_LIST_DOMAINS_AUTOSTART` and `VIR_CONNECT_LIST_DOMAINS_NO_AUTOSTART`, for filtering based on autostart; `VIR_CONNECT_LIST_DOMAINS_HAS_SNAPSHOT` and `VIR_CONNECT_LIST_DOMAINS_NO_SNAPSHOT`, for filtering based on whether a domain has snapshots.

Returns the number of domains found or -1 and sets domains to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling `virDomainFree()` on each array element, then calling `free()` on .

Example of usage: `virDomainPtr *domains; virDomainPtr dom; int i; int ret; unsigned int flags = VIR_CONNECT_LIST_RUNNING | VIR_CONNECT_LIST_PERSISTENT;`

```
ret = virConnectListAllDomains(conn, &domains, flags); if (ret < 0) error();
```

```
for (i = 0; i < ret; i++) { do_something_with_domain(domains[i]);
```

```
//here or in a separate loop if needed virDomainFree(domains[i]); }
```

```
free(domains);
```

#### 4.1.4.23 `int virConnectListAllInterfaces ( virConnectPtr conn, virInterfacePtr ** ifaces, unsigned int flags )`

`virConnectListAllInterfaces`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the interface objects or NULL if the list is not required (just returns number of interfaces). : bitwise-OR of `virConnectListAllInterfacesFlags`.

Collect the list of interfaces, and allocate an array to store those objects. This API solves the race inherent between `virConnectListInterfaces` and `virConnectListDefinedInterfaces`.

Normally, all interfaces are returned; however, can be used to filter the results for a smaller list of targeted interfaces. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a interface, and where all bits within a group describe all possible interfaces.

The only group of is `VIR_CONNECT_LIST_INTERFACES_ACTIVE` (up) and `VIR_CONNECT_LIST_INTERFACES_INACTIVE` (down) to filter the interfaces by state.

Returns the number of interfaces found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling `virStorageInterfaceFree()` on each array element, then calling `free()` on .

#### 4.1.4.24 `int virConnectListAllNetworks ( virConnectPtr conn, virNetworkPtr ** nets, unsigned int flags )`

`virConnectListAllNetworks`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the network objects or NULL if the list is not required (just returns number of networks). : bitwise-OR of `virConnectListAllNetworksFlags`.

Collect the list of networks, and allocate an array to store those objects. This API solves the race inherent between `virConnectListNetworks` and `virConnectListDefinedNetworks`.

Normally, all networks are returned; however, can be used to filter the results for a smaller list of targeted networks. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a network, and where all bits within a group describe all possible networks.

The first group of is `VIR_CONNECT_LIST_NETWORKS_ACTIVE` (up) and `VIR_CONNECT_LIST_NETWORKS_INACTIVE` (down) to filter the networks by state.

The second group of is `VIR_CONNECT_LIST_NETWORKS_PERSISTENT` (defined) and `VIR_CONNECT_LIST_NETWORKS_TRANSIENT` (running but not defined), to filter the networks by whether they have persistent config or not.

The third group of is `VIR_CONNECT_LIST_NETWORKS_AUTOSTART` and `VIR_CONNECT_LIST_NETWORKS_NO_AUTOSTART`, to filter the networks by whether they are marked as autostart or not.

Returns the number of networks found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virNetworkFree\(\)](#) on each array element, then calling `free()` on .

#### 4.1.4.25 `int virConnectListAllNodeDevices ( virConnectPtr conn, virNodeDevicePtr ** devices, unsigned int flags )`

`virConnectListAllNodeDevices`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the node device objects or NULL if the list is not required (just returns number of node devices). : bitwise-OR of `virConnectListAllNodeDevices`.

Collect the list of node devices, and allocate an array to store those objects.

Normally, all node devices are returned; however, can be used to filter the results for a smaller list of targeted node devices. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a node device, and where all bits within a group describe all possible node devices.

Only one group of the is provided to filter the node devices by capability type, flags include: `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SYSTEM` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_PCI_DEV` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_USB_DEV` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_USB_INTERFACE` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_NET` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI_HOST` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI_TARGET` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_STORAGE`

Returns the number of node devices found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virNodeDeviceFree\(\)](#) on each array element, then calling `free()` on .

#### 4.1.4.26 `int virConnectListAllNWFilters ( virConnectPtr conn, virNWFilterPtr ** filters, unsigned int flags )`

`virConnectListAllNWFilters`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the network filter objects or NULL if the list is not required (just returns number of network filters). : extra flags; not used yet, so callers should always pass 0

Collect the list of network filters, and allocate an array to store those objects.

Returns the number of network filters found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virNWFilterFree\(\)](#) on each array element, then calling `free()` on .

#### 4.1.4.27 `int virConnectListAllSecrets ( virConnectPtr conn, virSecretPtr ** secrets, unsigned int flags )`

`virConnectListAllSecrets`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the secret objects or NULL if the list is not required (just returns the number of secrets). : extra flags; not used yet, so callers should always pass 0

Collect the list of secrets, and allocate an array to store those objects.

Normally, all secrets are returned; however, can be used to filter the results for a smaller list of targeted secrets. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a secret, and where all bits within a group describe all possible secrets.

The first group of is used to filter secrets by its storage location. Flag `VIR_CONNECT_LIST_SECRETS_EPHEMERAL` selects secrets that are kept only in memory. Flag `VIR_CONNECT_LIST_SECRETS_NO_EPHEMERAL` selects secrets that are kept in persistent storage.

The second group of is used to filter secrets by privacy. Flag `VIR_CONNECT_LIST_SECRETS_PRIVATE` selects secrets that are never revealed to any caller of libvirt nor to any other node. Flag `VIR_CONNECT_LIST_SECRET_S_NO_PRIVATE` selects non-private secrets.

Returns the number of secrets found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration

easier. The caller is responsible for calling [virSecretFree\(\)](#) on each array element, then calling `free()` on .

#### 4.1.4.28 `int virConnectListAllStoragePools ( virConnectPtr conn, virStoragePoolPtr ** pools, unsigned int flags )`

`virConnectListAllStoragePools`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing storage pool objects or NULL if the list is not required (just returns number of pools). : bitwise-OR of `virConnectListAllStoragePoolsFlags`.

Collect the list of storage pools, and allocate an array to store those objects. This API solves the race inherent between `virConnectListStoragePools` and `virConnectListDefinedStoragePools`.

Normally, all storage pools are returned; however, can be used to filter the results for a smaller list of targeted pools. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a pool, and where all bits within a group describe all possible pools.

The first group of is `VIR_CONNECT_LIST_STORAGE_POOLS_ACTIVE` (online) and `VIR_CONNECT_LIST_STORAGE_POOLS_INACTIVE` (offline) to filter the pools by state.

The second group of is `VIR_CONNECT_LIST_STORAGE_POOLS_PERSISTENT` (defined) and `VIR_CONNECT_LIST_STORAGE_POOLS_TRANSIENT` (running but not defined), to filter the pools by whether they have persistent config or not.

The third group of is `VIR_CONNECT_LIST_STORAGE_POOLS_AUTOSTART` and `VIR_CONNECT_LIST_STORAGE_POOLS_NO_AUTOSTART`, to filter the pools by whether they are marked as autostart or not.

The last group of is provided to filter the pools by the types, the flags include: `VIR_CONNECT_LIST_STORAGE_POOLS_DIR` `VIR_CONNECT_LIST_STORAGE_POOLS_FS` `VIR_CONNECT_LIST_STORAGE_POOLS_NETFS` `VIR_CONNECT_LIST_STORAGE_POOLS_LOGICAL` `VIR_CONNECT_LIST_STORAGE_POOLS_DISK` `VIR_CONNECT_LIST_STORAGE_POOLS_ISCSI` `VIR_CONNECT_LIST_STORAGE_POOLS_SCSI` `VIR_CONNECT_LIST_STORAGE_POOLS_MPATH` `VIR_CONNECT_LIST_STORAGE_POOLS_RBD` `VIR_CONNECT_LIST_STORAGE_POOLS_SHEEPDOG`

Returns the number of storage pools found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virStoragePoolFree\(\)](#) on each array element, then calling `free()` on .

#### 4.1.4.29 `int virConnectListDefinedDomains ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListDefinedDomains`: : pointer to the hypervisor connection : pointer to an array to store the names : size of the array

list the defined but inactive domains, stores the pointers to the names in

For active domains, see [virConnectListDomains\(\)](#). For more control over the results, see [virConnectListAllDomains\(\)](#).

Returns the number of names provided in the array or -1 in case of error. Note that this command is inherently racy; a domain can be defined between a call to [virConnectNumOfDefinedDomains\(\)](#) and this call; you are only guaranteed that all currently defined domains were listed if the return is less than . The client must call `free()` on each returned name.

#### 4.1.4.30 `int virConnectListDefinedInterfaces ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListDefinedInterfaces`: : pointer to the hypervisor connection : array to collect the list of names of interfaces : size of

Collect the list of defined (inactive) physical host interfaces, and store their names in .

For more control over the results, see [virConnectListAllInterfaces\(\)](#).

Returns the number of names provided in the array or -1 in case of error. Note that this command is inherently racy; a interface can be defined between a call to [virConnectNumOfDefinedInterfaces\(\)](#) and this call; you are only



guaranteed that all currently defined interfaces were listed if the return is less than . The client must call free() on each returned name.

#### 4.1.4.31 int virConnectListDefinedNetworks ( virConnectPtr conn, char \*\*const names, int maxnames )

virConnectListDefinedNetworks: : pointer to the hypervisor connection : pointer to an array to store the names : size of the array

list the inactive networks, stores the pointers to the names in

For more control over the results, see [virConnectListAllNetworks\(\)](#).

Returns the number of names provided in the array or -1 in case of error. Note that this command is inherently racy; a network can be defined between a call to [virConnectNumOfDefinedNetworks\(\)](#) and this call; you are only guaranteed that all currently defined networks were listed if the return is less than . The client must call free() on each returned name.

#### 4.1.4.32 int virConnectListDefinedStoragePools ( virConnectPtr conn, char \*\*const names, int maxnames )

virConnectListDefinedStoragePools: : pointer to hypervisor connection : array of char \* to fill with pool names (allocated by caller) : size of the names array

Provides the list of names of inactive storage pools up to maxnames. If there are more than maxnames, the remaining names will be silently ignored.

For more control over the results, see [virConnectListAllStoragePools\(\)](#).

Returns the number of names provided in the array or -1 in case of error. Note that this command is inherently racy; a pool can be defined between a call to [virConnectNumOfDefinedStoragePools\(\)](#) and this call; you are only guaranteed that all currently defined pools were listed if the return is less than . The client must call free() on each returned name.

#### 4.1.4.33 int virConnectListDomains ( virConnectPtr conn, int \* ids, int maxids )

virConnectListDomains: : pointer to the hypervisor connection : array to collect the list of IDs of active domains : size of

Collect the list of active domains, and store their IDs in array

For inactive domains, see [virConnectListDefinedDomains\(\)](#). For more control over the results, see [virConnectListAllDomains\(\)](#).

Returns the number of domains found or -1 in case of error. Note that this command is inherently racy; a domain can be started between a call to [virConnectNumOfDomains\(\)](#) and this call; you are only guaranteed that all currently active domains were listed if the return is less than .

#### 4.1.4.34 int virConnectListInterfaces ( virConnectPtr conn, char \*\*const names, int maxnames )

virConnectListInterfaces: : pointer to the hypervisor connection : array to collect the list of names of interfaces : size of

Collect the list of active physical host interfaces, and store their names in

For more control over the results, see [virConnectListAllInterfaces\(\)](#).

Returns the number of interfaces found or -1 in case of error. Note that this command is inherently racy; a interface can be started between a call to [virConnectNumOfInterfaces\(\)](#) and this call; you are only guaranteed that all currently active interfaces were listed if the return is less than .

#### 4.1.4.35 int virConnectListNetworks ( virConnectPtr conn, char \*\*const names, int maxnames )

virConnectListNetworks: : pointer to the hypervisor connection : array to collect the list of names of active networks  
: size of

Collect the list of active networks, and store their names in

For more control over the results, see [virConnectListAllNetworks\(\)](#).

Returns the number of networks found or -1 in case of error. Note that this command is inherently racy; a network can be started between a call to [virConnectNumOfNetworks\(\)](#) and this call; you are only guaranteed that all currently active networks were listed if the return is less than .

#### 4.1.4.36 int virConnectListNWFilters ( virConnectPtr conn, char \*\*const names, int maxnames )

virConnectListNWFilters: : pointer to the hypervisor connection : array to collect the list of names of network filters  
: size of

Collect the list of network filters, and store their names in

Returns the number of network filters found or -1 in case of error

#### 4.1.4.37 int virConnectListSecrets ( virConnectPtr conn, char \*\* uuids, int maxuuids )

virConnectListSecrets: : virConnect connection : Pointer to an array to store the UUIDs : size of the array.

List UUIDs of defined secrets, store pointers to names in uuids.

Returns the number of UUIDs provided in the array, or -1 on failure.

#### 4.1.4.38 int virConnectListStoragePools ( virConnectPtr conn, char \*\*const names, int maxnames )

virConnectListStoragePools: : pointer to hypervisor connection : array of char \* to fill with pool names (allocated by caller) : size of the names array

Provides the list of names of active storage pools up to maxnames. If there are more than maxnames, the remaining names will be silently ignored.

For more control over the results, see [virConnectListAllStoragePools\(\)](#).

Returns the number of pools found or -1 in case of error. Note that this command is inherently racy; a pool can be started between a call to [virConnectNumOfStoragePools\(\)](#) and this call; you are only guaranteed that all currently active pools were listed if the return is less than .

#### 4.1.4.39 int virConnectNumOfDefinedDomains ( virConnectPtr conn )

virConnectNumOfDefinedDomains: : pointer to the hypervisor connection

Provides the number of defined but inactive domains.

Returns the number of domain found or -1 in case of error

#### 4.1.4.40 int virConnectNumOfDefinedInterfaces ( virConnectPtr conn )

virConnectNumOfDefinedInterfaces: : pointer to the hypervisor connection

Provides the number of defined (inactive) interfaces on the physical host.

Returns the number of defined interface found or -1 in case of error

**4.1.4.41 int virConnectNumOfDefinedNetworks ( virConnectPtr conn )**

virConnectNumOfDefinedNetworks: : pointer to the hypervisor connection

Provides the number of inactive networks.

Returns the number of networks found or -1 in case of error

**4.1.4.42 int virConnectNumOfDefinedStoragePools ( virConnectPtr conn )**

virConnectNumOfDefinedStoragePools: : pointer to hypervisor connection

Provides the number of inactive storage pools

Returns the number of pools found, or -1 on error

**4.1.4.43 int virConnectNumOfDomains ( virConnectPtr conn )**

virConnectNumOfDomains: : pointer to the hypervisor connection

Provides the number of active domains.

Returns the number of domain found or -1 in case of error

**4.1.4.44 int virConnectNumOfInterfaces ( virConnectPtr conn )**

virConnectNumOfInterfaces: : pointer to the hypervisor connection

Provides the number of active interfaces on the physical host.

Returns the number of active interfaces found or -1 in case of error

**4.1.4.45 int virConnectNumOfNetworks ( virConnectPtr conn )**

virConnectNumOfNetworks: : pointer to the hypervisor connection

Provides the number of active networks.

Returns the number of network found or -1 in case of error

**4.1.4.46 int virConnectNumOfNWFilters ( virConnectPtr conn )**

virConnectNumOfNWFilters: : pointer to the hypervisor connection

Provides the number of nwfilters.

Returns the number of nwfilters found or -1 in case of error

**4.1.4.47 int virConnectNumOfSecrets ( virConnectPtr conn )**

virConnectNumOfSecrets: : virConnect connection

Fetch number of currently defined secrets.

Returns the number currently defined secrets.

**4.1.4.48 int virConnectNumOfStoragePools ( virConnectPtr conn )**

virConnectNumOfStoragePools: : pointer to hypervisor connection

Provides the number of active storage pools

Returns the number of pools found, or -1 on error

**4.1.4.49** `virConnectPtr virConnectOpen ( const char * name )`

**4.1.4.50** `virConnectPtr virConnectOpenAuth ( const char * name, virConnectAuthPtr auth, unsigned int flags )`

**4.1.4.51** `virConnectPtr virConnectOpenReadOnly ( const char * name )`

**4.1.4.52** `int virConnectRef ( virConnectPtr conn )`

`virConnectRef`: : the connection to hold a reference on

Increment the reference count on the connection. For each additional call to this method, there shall be a corresponding call to `virConnectClose` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a connection would increment the reference count.

Returns 0 in case of success, -1 in case of failure

**4.1.4.53** `int virConnectRegisterCloseCallback ( virConnectPtr conn, virConnectCloseFunc cb, void * opaque, virFreeCallback freecb )`

`virConnectRegisterCloseCallback`: : pointer to connection object : callback to invoke upon close : user data to pass to : callback to free

Registers a callback to be invoked when the connection is closed. This callback is invoked when there is any condition that causes the socket connection to the hypervisor to be closed.

This function is only applicable to hypervisor drivers which maintain a persistent open connection. Drivers which open a new connection for every operation will not invoke this.

The must not invoke any other libvirt public APIs, since it is not called from a re-entrant safe context.

Returns 0 on success, -1 on error

**4.1.4.54** `int virConnectSetKeepAlive ( virConnectPtr conn, int interval, unsigned int count )`

`virConnectSetKeepAlive`: : pointer to a hypervisor connection : number of seconds of inactivity before a keepalive message is sent : number of messages that can be sent in a row

Start sending keepalive messages after interval second of inactivity and consider the connection to be broken when no response is received after count keepalive messages sent in a row. In other words, sending count + 1 keepalive message results in closing the connection. When interval is  $\leq 0$ , no keepalive messages will be sent. When count is 0, the connection will be automatically closed after interval seconds of inactivity without sending any keepalive messages.

Note: client has to implement and run event loop to be able to use keepalive messages. Failure to do so may result in connections being closed unexpectedly.

Note: This API function controls only keepalive messages sent by the client. If the server is configured to use keepalive you still need to run the event loop to respond to them, even if you disable keepalives by this function.

Returns -1 on error, 0 on success, 1 when remote party doesn't support keepalive messages.

**4.1.4.55** `int virConnectUnregisterCloseCallback ( virConnectPtr conn, virConnectCloseFunc cb )`

`virConnectUnregisterCloseCallback`: : pointer to connection object : pointer to the current registered callback

Unregisters the callback previously set with the `virConnectRegisterCloseCallback` method. The callback will no longer receive notifications when the connection closes. If a `virFreeCallback` was provided at time of registration, it will be invoked

Returns 0 on success, -1 on error

#### 4.1.4.56 `int virDomainAbortJob ( virDomainPtr domain )`

`virDomainAbortJob`: : a domain object

Requests that the current background job be aborted at the soonest opportunity. This will block until the job has either completed, or aborted.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.57 `int virDomainAttachDevice ( virDomainPtr domain, const char * xml )`

`virDomainAttachDevice`: : pointer to domain object : pointer to XML description of one device

Create a virtual device attachment to backend. This function, having hotplug semantics, is only allowed on an active domain.

For compatibility, this method can also be used to change the media in an existing CDROM/Floppy device, however, applications are recommended to use the `virDomainUpdateDeviceFlag` method instead.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.58 `int virDomainAttachDeviceFlags ( virDomainPtr domain, const char * xml, unsigned int flags )`

`virDomainAttachDeviceFlags`: : pointer to domain object : pointer to XML description of one device : bitwise-OR of `virDomainDeviceModifyFlags`

Attach a virtual device to a domain, using the flags parameter to control how the device is attached. `VIR_DOMAIN_AFFECT_CURRENT` specifies that the device allocation is made based on current domain state. `VIR_DOMAIN_AFFECT_LIVE` specifies that the device shall be allocated to the active domain instance only and is not added to the persisted domain configuration. `VIR_DOMAIN_AFFECT_CONFIG` specifies that the device shall be allocated to the persisted domain configuration only. Note that the target hypervisor must return an error if unable to satisfy flags. E.g. the hypervisor driver will return failure if LIVE is specified but it only supports modifying the persisted device allocation.

For compatibility, this method can also be used to change the media in an existing CDROM/Floppy device, however, applications are recommended to use the `virDomainUpdateDeviceFlag` method instead.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.59 `int virDomainBlockCommit ( virDomainPtr dom, const char * disk, const char * base, const char * top, unsigned long bandwidth, unsigned int flags )`

`virDomainBlockCommit`: : pointer to domain object : path to the block device, or device shorthand : path to backing file to merge into, or NULL for default : path to file within backing chain that contains data to be merged, or NULL to merge all possible data : (optional) specify commit bandwidth limit in MiB/s : bitwise-OR of `virDomainBlockCommitFlags`

Commit changes that were made to temporary top-level files within a disk image backing file chain into a lower-level base file. In other words, take all the difference between and , and update to contain that difference; after the commit, any portion of the chain that previously depended on will now depend on , and all files after up to and including will now be invalidated. A typical use of this command is to reduce the length of a backing file chain after taking an external disk snapshot. To move data in the opposite direction, see [virDomainBlockPull\(\)](#).

This command starts a long-running commit block job, whose status may be tracked by [virDomainBlockJobInfo\(\)](#) with a job type of `VIR_DOMAIN_BLOCK_JOB_TYPE_COMMIT`, and the operation can be aborted with [virDomain-](#)

[BlockJobAbort\(\)](#). When finished, an asynchronous event is raised to indicate the final status, and the job no longer exists. If the job is aborted, it is up to the hypervisor whether starting a new job will resume from the same point, or start over.

Be aware that this command may invalidate files even if it is aborted; the user is cautioned against relying on the contents of invalidated intermediate files such as without manually rebasing those files to use a backing file of a read-only copy of prior to the point where the commit operation was started (although such a rebase cannot be safely done until the commit has successfully completed). However, the domain itself will not have any issues; the active layer remains valid throughout the entire commit operation. As a convenience, if contains `VIR_DOMAIN_BLOCK_COMMIT_DELETE`, this command will unlink all files that were invalidated, after the commit successfully completes.

By default, if is `NULL`, the commit target will be the bottom of the backing chain; if contains `VIR_DOMAIN_BLOCK_COMMIT_SHALLOW`, then the immediate backing file of will be used instead. If is `NULL`, the active image at the top of the chain will be used. Some hypervisors place restrictions on how much can be committed, and might fail if is not the immediate backing file of , or if is the active layer in use by a running domain, or if is not the top-most file; restrictions may differ for online vs. offline domains.

The parameter is either an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `/path/to/image`), or the device target shorthand (the `<target dev=...">` sub-element, such as `xvda`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

The maximum bandwidth (in MiB/s) that will be used to do the commit can be specified with the bandwidth parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0; in this case, it might still be possible for a later call to [virDomainBlockJobSetSpeed\(\)](#) to succeed. The actual speed can be determined with [virDomainGetBlockJobInfo\(\)](#).

Returns 0 if the operation has started, -1 on failure.

#### 4.1.4.60 `int virDomainBlockJobAbort ( virDomainPtr dom, const char * disk, unsigned int flags )`

`virDomainBlockJobAbort`: : pointer to domain object : path to the block device, or device shorthand : bitwise-OR of `virDomainBlockJobAbortFlags`

Cancel the active block job on the given disk.

The parameter is either an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `/path/to/image`), or (since 0.9.5) the device target shorthand (the `<target dev=...">` sub-element, such as `xvda`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

If the current block job for is `VIR_DOMAIN_BLOCK_JOB_TYPE_PULL`, then by default, this function performs a synchronous operation and the caller may assume that the operation has completed when 0 is returned. However, BlockJob operations may take a long time to cancel, and during this time further domain interactions may be unresponsive. To avoid this problem, pass `VIR_DOMAIN_BLOCK_JOB_ABORT_ASYNC` in the argument to enable asynchronous behavior, returning as soon as possible. When the job has been canceled, a BlockJob event will be emitted, with status `VIR_DOMAIN_BLOCK_JOB_CANCELED` (even if the `ABORT_ASYNC` flag was not used); it is also possible to poll [virDomainBlockJobInfo\(\)](#) to see if the job cancellation is still pending. This type of job can be restarted to pick up from where it left off.

If the current block job for is `VIR_DOMAIN_BLOCK_JOB_TYPE_COPY`, then the default is to abort the mirroring and revert to the source disk; adding of `VIR_DOMAIN_BLOCK_JOB_ABORT_PIVOT` causes this call to fail with `VIR_ERR_BLOCK_COPY_ACTIVE` if the copy is not fully populated, otherwise it will swap the disk over to the copy to end the mirroring. An event will be issued when the job is ended, and it is possible to use `VIR_DOMAIN_BLOCK_JOB_ABORT_ASYNC` to control whether this command waits for the completion of the job. Restarting this job requires starting over from the beginning of the first phase.

Returns -1 in case of failure, 0 when successful.

**4.1.4.61** `int virDomainBlockJobSetSpeed ( virDomainPtr dom, const char * disk, unsigned long bandwidth, unsigned int flags )`

virDomainBlockJobSetSpeed: : pointer to domain object : path to the block device, or device shorthand : specify bandwidth limit in MiB/s : extra flags; not used yet, so callers should always pass 0

Set the maximum allowable bandwidth that a block job may consume. If bandwidth is 0, the limit will revert to the hypervisor default.

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or (since 0.9.5) the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Returns -1 in case of failure, 0 when successful.

**4.1.4.62** `int virDomainBlockPeek ( virDomainPtr dom, const char * disk, unsigned long long offset, size_t size, void * buffer, unsigned int flags )`

virDomainBlockPeek: : pointer to the domain object : path to the block device, or device shorthand : offset within block device : size to read : return buffer (must be at least size bytes) : extra flags; not used yet, so callers should always pass 0

This function allows you to read the contents of a domain's disk device.

Typical uses for this are to determine if the domain has written a Master Boot Record (indicating that the domain has completed installation), or to try to work out the state of the domain's filesystems.

(Note that in the local case you might try to open the block device or file directly, but that won't work in the remote case, nor if you don't have sufficient permission. Hence the need for this call).

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or (since 0.9.5) the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

'offset' and 'size' represent an area which must lie entirely within the device or file. 'size' may be 0 to test if the call would succeed.

'buffer' is the return buffer and must be at least 'size' bytes.

NB. The remote driver imposes a 64K byte limit on 'size'. For your program to be able to work reliably over a remote connection you should split large requests to  $\leq 65536$  bytes. However, with 0.9.13 this RPC limit has been raised to 1M byte.

Returns: 0 in case of success or -1 in case of failure.

**4.1.4.63** `int virDomainBlockPull ( virDomainPtr dom, const char * disk, unsigned long bandwidth, unsigned int flags )`

virDomainBlockPull: : pointer to domain object : path to the block device, or device shorthand : (optional) specify copy bandwidth limit in MiB/s : extra flags; not used yet, so callers should always pass 0

Populate a disk image with data from its backing image. Once all data from its backing image has been pulled, the disk no longer depends on a backing image. This function pulls data for the entire device in the background. Progress of the operation can be checked with [virDomainGetBlockJobInfo\(\)](#) and the operation can be aborted with [virDomainBlockJobAbort\(\)](#). When finished, an asynchronous event is raised to indicate the final status. To move data in the opposite direction, see [virDomainBlockCommit\(\)](#).

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or (since 0.9.5) the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

The maximum bandwidth (in MiB/s) that will be used to do the copy can be specified with the bandwidth parameter.

If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0; in this case, it might still be possible for a later call to [virDomainBlockJobSetSpeed\(\)](#) to succeed. The actual speed can be determined with [virDomainGetBlockJobInfo\(\)](#).

This is shorthand for [virDomainBlockRebase\(\)](#) with a NULL base.

Returns 0 if the operation has started, -1 on failure.

**4.1.4.64** `int virDomainBlockRebase ( virDomainPtr dom, const char * disk, const char * base, unsigned long bandwidth, unsigned int flags )`

`virDomainBlockRebase`: : pointer to domain object : path to the block device, or device shorthand : path to backing file to keep, or NULL for no backing file : (optional) specify copy bandwidth limit in MiB/s : bitwise-OR of `virDomainBlockRebaseFlags`

Populate a disk image with data from its backing image chain, and setting the backing image to , or alternatively copy an entire backing chain to a new file .

When is 0, this starts a pull, where must be the absolute path of one of the backing images further up the chain, or NULL to convert the disk image so that it has no backing image. Once all data from its backing image chain has been pulled, the disk no longer depends on those intermediate backing images. This function pulls data for the entire device in the background. Progress of the operation can be checked with [virDomainGetBlockJobInfo\(\)](#) with a job type of `VIR_DOMAIN_BLOCK_JOB_TYPE_PULL`, and the operation can be aborted with [virDomainBlockJobAbort\(\)](#). When finished, an asynchronous event is raised to indicate the final status, and the job no longer exists. If the job is aborted, a new one can be started later to resume from the same point.

When includes `VIR_DOMAIN_BLOCK_REBASE_COPY`, this starts a copy, where must be the name of a new file to copy the chain to. By default, the copy will pull the entire source chain into the destination file, but if also contains `VIR_DOMAIN_BLOCK_REBASE_SHALLOW`, then only the top of the source chain will be copied (the source and destination have a common backing file). By default, will be created with the same file format as the source, but this can be altered by adding `VIR_DOMAIN_BLOCK_REBASE_COPY_RAW` to force the copy to be raw (does not make sense with the shallow flag unless the source is also raw), or by using `VIR_DOMAIN_BLOCK_REBASE_REUSE_EXT` to reuse an existing file with initial contents identical to the backing file of the source (this allows a management app to pre-create files with relative backing file names, rather than the default of absolute backing file names; as a security precaution, you should generally only use `reuse_ext` with the shallow flag and a non-raw destination file).

A copy job has two parts; in the first phase, the parameter affects how fast the source is pulled into the destination, and the job can only be canceled by reverting to the source file; progress in this phase can be tracked via the [virDomainBlockJobInfo\(\)](#) command, with a job type of `VIR_DOMAIN_BLOCK_JOB_TYPE_COPY`. The job transitions to the second phase when the job info states `cur == end`, and remains alive to mirror all further changes to both source and destination. The user must call [virDomainBlockJobAbort\(\)](#) to end the mirroring while choosing whether to revert to source or pivot to the destination. An event is issued when the job ends, and in the future, an event may be added when the job transitions from pulling to mirroring. If the job is aborted, a new job will have to start over from the beginning of the first phase.

Some hypervisors will restrict certain actions, such as [virDomainSave\(\)](#) or [virDomainDetachDevice\(\)](#), while a copy job is active; they may also restrict a copy job to transient domains.

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

The maximum bandwidth (in MiB/s) that will be used to do the copy can be specified with the bandwidth parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0; in this case, it might still be possible for a later call to [virDomainBlockJobSetSpeed\(\)](#) to succeed. The actual speed can be determined with [virDomainGetBlockJobInfo\(\)](#).

When is NULL and is 0, this is identical to [virDomainBlockPull\(\)](#).

Returns 0 if the operation has started, -1 on failure.



#### 4.1.4.65 `int virDomainBlockResize ( virDomainPtr dom, const char * disk, unsigned long long size, unsigned int flags )`

`virDomainBlockResize`: : pointer to the domain object : path to the block image, or shorthand : new size of the block image, see below for unit : bitwise-OR of `virDomainBlockResizeFlags`

Resize a block device of domain while the domain is running. If is 0, then is in kibibytes (blocks of 1024 bytes); since 0.9.11, if includes `VIR_DOMAIN_BLOCK_RESIZE_BYTES`, is in bytes instead. is taken directly as the new size. Depending on the file format, the hypervisor may round up to the next alignment boundary.

The parameter is either an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `"/path/to/image"`), or (since 0.9.5) the device target shorthand (the `<target dev=...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Note that this call may fail if the underlying virtualization hypervisor does not support it; this call requires privileged access to the hypervisor.

Returns: 0 in case of success or -1 in case of failure.

#### 4.1.4.66 `int virDomainBlockStats ( virDomainPtr dom, const char * disk, virDomainBlockStatsPtr stats, size_t size )`

`virDomainBlockStats`: : pointer to the domain object : path to the block device, or device shorthand : block device stats (returned) : size of stats structure

This function returns block device (disk) stats for block devices attached to the domain.

The parameter is either the device target shorthand (the `<target dev=...">` sub-element, such as `"xvda"`), or (since 0.9.8) an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `"/path/to/image"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Domains may have more than one block device. To get stats for each you should make multiple calls to this function.

Individual fields within the stats structure may be returned as -1, which indicates that the hypervisor does not support that particular statistic.

Returns: 0 in case of success or -1 in case of failure.

#### 4.1.4.67 `int virDomainBlockStatsFlags ( virDomainPtr dom, const char * disk, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainBlockStatsFlags`: : pointer to domain object : path to the block device, or device shorthand : pointer to block stats parameter object (return value) : pointer to number of block stats; input and output : bitwise-OR of `virTypedParameterFlags`

This function is to get block stats parameters for block devices attached to the domain.

The parameter is either the device target shorthand (the `<target dev=...">` sub-element, such as `"xvda"`), or (since 0.9.8) an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `"/path/to/image"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Domains may have more than one block device. To get stats for each you should make multiple calls to this function.

On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. (Note that block devices of different types might support different parameters, so it might be necessary to compute for each block device). The caller should then allocate array, i.e. `(sizeof() * )` bytes and call the API again. See [virDomainGetMemoryParameters\(\)](#) for more details.

Returns -1 in case of error, 0 in case of success.

#### 4.1.4.68 `int virDomainCoreDump ( virDomainPtr domain, const char * to, unsigned int flags )`

`virDomainCoreDump`: : a domain object : path for the core file : bitwise-OR of `virDomainCoreDumpFlags`

This method will dump the core of a domain on a given file for analysis. Note that for remote Xen Daemon the file path will be interpreted in the remote host. Hypervisors may require the user to manually ensure proper permissions on the file named by .

If includes `VIR_DUMP_CRASH`, then leave the guest shut off with a crashed state after the dump completes. If includes `VIR_DUMP_LIVE`, then make the core dump while continuing to allow the guest to run; otherwise, the guest is suspended during the dump. `VIR_DUMP_RESET` flag forces reset of the quest after dump. The above three flags are mutually exclusive.

Additionally, if includes `VIR_DUMP_BYPASS_CACHE`, then libvirt will attempt to bypass the file system cache while creating the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing saves to NFS.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.69 `int virDomainCreate ( virDomainPtr domain )`

`virDomainCreate`: : pointer to a defined domain

Launch a defined domain. If the call succeeds the domain moves from the defined to the running domains pools. The domain will be paused only if restoring from managed state created from a paused domain. For more control, see [virDomainCreateWithFlags\(\)](#).

Returns 0 in case of success, -1 in case of error

#### 4.1.4.70 `virDomainPtr virDomainCreateLinux ( virConnectPtr conn, const char * xmlDesc, unsigned int flags )`

`virDomainCreateLinux`: : pointer to the hypervisor connection : string containing an XML description of the domain : extra flags; not used yet, so callers should always pass 0

Deprecated after 0.4.6. Renamed to [virDomainCreateXML\(\)](#) providing identical functionality. This existing name will left indefinitely for API compatibility.

Returns a new domain object or NULL in case of failure

#### 4.1.4.71 `int virDomainCreateWithFlags ( virDomainPtr domain, unsigned int flags )`

`virDomainCreateWithFlags`: : pointer to a defined domain : bitwise-OR of supported `virDomainCreateFlags`

Launch a defined domain. If the call succeeds the domain moves from the defined to the running domains pools.

If the `VIR_DOMAIN_START_PAUSED` flag is set, or if the guest domain has a managed save image that requested paused state (see [virDomainManagedSave\(\)](#)) the guest domain will be started, but its CPUs will remain paused. The CPUs can later be manually started using [virDomainResume\(\)](#). In all other cases, the guest domain will be running.

If the `VIR_DOMAIN_START_AUTODESTROY` flag is set, the guest domain will be automatically destroyed when the `virConnectPtr` object is finally released. This will also happen if the client application crashes / loses its connection to the libvirtd daemon. Any domains marked for auto destroy will block attempts at migration, save-to-file, or snapshots.

If the `VIR_DOMAIN_START_BYPASS_CACHE` flag is set, and there is a managed save file for this domain (created by [virDomainManagedSave\(\)](#)), then libvirt will attempt to bypass the file system cache while restoring the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing loads from NFS.

If the `VIR_DOMAIN_START_FORCE_BOOT` flag is set, then any managed save file for this domain is discarded, and the domain boots from scratch.

Returns 0 in case of success, -1 in case of error

#### 4.1.4.72 **virDomainPtr virDomainCreateXML ( virConnectPtr conn, const char \* xmlDesc, unsigned int flags )**

virDomainCreateXML: : pointer to the hypervisor connection : string containing an XML description of the domain : bitwise-OR of supported virDomainCreateFlags

Launch a new guest domain, based on an XML description similar to the one returned by [virDomainGetXMLDesc\(\)](#). This function may require privileged access to the hypervisor. The domain is not persistent, so its definition will disappear when it is destroyed, or if the host is restarted (see [virDomainDefineXML\(\)](#) to define persistent domains).

If the VIR\_DOMAIN\_START\_PAUSED flag is set, the guest domain will be started, but its CPUs will remain paused. The CPUs can later be manually started using [virDomainResume](#).

If the VIR\_DOMAIN\_START\_AUTODESTROY flag is set, the guest domain will be automatically destroyed when the virConnectPtr object is finally released. This will also happen if the client application crashes / loses its connection to the libvirtd daemon. Any domains marked for auto destroy will block attempts at migration, save-to-file, or snapshots.

Returns a new domain object or NULL in case of failure

#### 4.1.4.73 **virDomainPtr virDomainDefineXML ( virConnectPtr conn, const char \* xml )**

virDomainDefineXML: : pointer to the hypervisor connection : the XML description for the domain, preferably in UTF-8

Define a domain, but does not start it. This definition is persistent, until explicitly undefined with [virDomainUndefine\(\)](#). A previous definition for this domain would be overridden if it already exists.

Some hypervisors may prevent this operation if there is a current block copy operation on a transient domain with the same id as the domain being defined; in that case, use [virDomainBlockJobAbort\(\)](#) to stop the block copy first.

Returns NULL in case of error, a pointer to the domain otherwise

#### 4.1.4.74 **int virDomainDestroy ( virDomainPtr domain )**

virDomainDestroy: : a domain object

Destroy the domain object. The running instance is shutdown if not down already and all resources used by it are given back to the hypervisor. This does not free the associated virDomainPtr object. This function may require privileged access.

virDomainDestroy first requests that a guest terminate (e.g. SIGTERM), then waits for it to comply. After a reasonable timeout, if the guest still exists, virDomainDestroy will forcefully terminate the guest (e.g. SIGKILL) if necessary (which may produce undesirable results, for example unflushed disk cache in the guest). To avoid this possibility, it's recommended to instead call [virDomainDestroyFlags](#), sending the VIR\_DOMAIN\_DESTROY\_GRACEFUL flag.

If the domain is transient and has any snapshot metadata (see [virDomainSnapshotNum\(\)](#)), then that metadata will automatically be deleted when the domain quits.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.75 **int virDomainDestroyFlags ( virDomainPtr domain, unsigned int flags )**

virDomainDestroyFlags: : a domain object : bitwise-OR of virDomainDestroyFlagsValues

Destroy the domain object. The running instance is shutdown if not down already and all resources used by it are given back to the hypervisor. This does not free the associated virDomainPtr object. This function may require privileged access.

Calling this function with no set (equal to zero) is equivalent to calling [virDomainDestroy](#), and after a reasonable timeout will forcefully terminate the guest (e.g. SIGKILL) if necessary (which may produce undesirable results,

for example unflushed disk cache in the guest). Including `VIR_DOMAIN_DESTROY_GRACEFUL` in the flags will prevent the forceful termination of the guest, and `virDomainDestroyFlags` will instead return an error if the guest doesn't terminate by the end of the timeout; at that time, the management application can decide if calling again without `VIR_DOMAIN_DESTROY_GRACEFUL` is appropriate.

Another alternative which may produce cleaner results for the guest's disks is to use `virDomainShutdown()` instead, but that depends on guest support (some hypervisor/guest combinations may ignore the shutdown request).

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.76 `int virDomainDetachDevice ( virDomainPtr domain, const char * xml )`

`virDomainDetachDevice`: : pointer to domain object : pointer to XML description of one device

Destroy a virtual device attachment to backend. This function, having hot-unplug semantics, is only allowed on an active domain.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.77 `int virDomainDetachDeviceFlags ( virDomainPtr domain, const char * xml, unsigned int flags )`

`virDomainDetachDeviceFlags`: : pointer to domain object : pointer to XML description of one device : bitwise-OR of `virDomainDeviceModifyFlags`

Detach a virtual device from a domain, using the flags parameter to control how the device is detached. `VIR_DOMAIN_AFFECT_CURRENT` specifies that the device allocation is removed based on current domain state. `VIR_DOMAIN_AFFECT_LIVE` specifies that the device shall be deallocated from the active domain instance only and is not from the persisted domain configuration. `VIR_DOMAIN_AFFECT_CONFIG` specifies that the device shall be deallocated from the persisted domain configuration only. Note that the target hypervisor must return an error if unable to satisfy flags. E.g. the hypervisor driver will return failure if `LIVE` is specified but it only supports removing the persisted device allocation.

Some hypervisors may prevent this operation if there is a current block copy operation on the device being detached; in that case, use `virDomainBlockJobAbort()` to stop the block copy first.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.78 `int virDomainFree ( virDomainPtr domain )`

`virDomainFree`: : a domain object

Free the domain object. The running instance is kept alive. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.79 `int virDomainGetAutostart ( virDomainPtr domain, int * autostart )`

`virDomainGetAutostart`: : a domain object : the value returned

Provides a boolean value indicating whether the domain configured to be automatically started when the host machine boots.

Returns -1 in case of error, 0 in case of success

#### 4.1.4.80 `int virDomainGetBlkioParameters ( virDomainPtr domain, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainGetBlkioParameters`: : pointer to domain object : pointer to blkio parameter object (return value, allocated by the caller) : pointer to number of blkio parameters; input and output : bitwise-OR of `virDomainModificationImpact`

and `virTypedParameterFlags`

Get all blkio parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. The caller should then allocate array, i.e. `(sizeof() * )` bytes and call the API again.

See [virDomainGetMemoryParameters\(\)](#) for an equivalent usage example.

This function may require privileged access to the hypervisor. This function expects the caller to allocate the .

Returns -1 in case of error, 0 in case of success.

**4.1.4.81** `int virDomainGetBlockInfo ( virDomainPtr domain, const char * disk, virDomainBlockInfoPtr info, unsigned int flags )`

`virDomainGetBlockInfo`: : a domain object : path to the block device, or device shorthand : pointer to a `virDomainBlockInfo` structure allocated by the user : extra flags; not used yet, so callers should always pass 0

Extract information about a domain's block device.

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or (since 0.9.5) the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.82** `int virDomainGetBlockioTune ( virDomainPtr dom, const char * disk, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainGetBlockioTune`: : pointer to domain object : path to the block device, or device shorthand : Pointer to blkio parameter object (return value, allocated by the caller) : Pointer to number of blkio parameters : bitwise-OR of `virDomainModificationImpact` and `virTypedParameterFlags`

Get all block IO tunable parameters for a given device. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor, either for the given (note that block devices of different types might support different parameters), or if is NULL, for all possible disks. The caller should then allocate array, i.e. `(sizeof() * )` bytes and call the API again. See [virDomainGetMemoryParameters\(\)](#) for more details.

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`. This parameter cannot be NULL unless is 0 on input.

Returns -1 in case of error, 0 in case of success.

**4.1.4.83** `int virDomainGetBlockJobInfo ( virDomainPtr dom, const char * disk, virDomainBlockJobInfoPtr info, unsigned int flags )`

`virDomainGetBlockJobInfo`: : pointer to domain object : path to the block device, or device shorthand : pointer to a `virDomainBlockJobInfo` structure : extra flags; not used yet, so callers should always pass 0

Request block job information for the given disk. If an operation is active will be updated with the current progress.

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or (since 0.9.5) the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Returns -1 in case of failure, 0 when nothing found, 1 when info was found.

#### 4.1.4.84 **virConnectPtr** virDomainGetConnect ( **virDomainPtr** dom )

virDomainGetConnect: : pointer to a domain

Provides the connection pointer associated with a domain. The reference counter on the connection is not increased by this call.

WARNING: When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the domain object together.

Returns the virConnectPtr or NULL in case of failure.

#### 4.1.4.85 **int** virDomainGetControllInfo ( **virDomainPtr** domain, **virDomainControllInfoPtr** info, unsigned int flags )

virDomainGetControllInfo: : a domain object : pointer to a virDomainControllInfo structure allocated by the user : extra flags; not used yet, so callers should always pass 0

Extract details about current state of control interface to a domain.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.86 **int** virDomainGetCPUStats ( **virDomainPtr** domain, **virTypedParameterPtr** params, unsigned int nparams, int start\_cpu, unsigned int ncpus, unsigned int flags )

virDomainGetCPUStats: : domain to query : array to populate on output : number of parameters per cpu : which cpu to start with, or -1 for summary : how many cpus to query : bitwise-OR of virTypedParameterFlags

Get statistics relating to CPU usage attributable to a single domain (in contrast to the statistics returned by [virNodeGetCPUStats\(\)](#) for all processes on the host). must be running (an inactive domain has no attributable cpu usage). On input, must contain at least \* entries, allocated by the caller.

If is -1, then must be 1, and the returned results reflect the statistics attributable to the entire domain (such as user and system time for the process as a whole). Otherwise, represents which cpu to start with, and represents how many consecutive processors to query, with statistics attributable per processor (such as per-cpu usage). If is larger than the number of cpus available to query, then the trailing part of the array will be unpopulated.

The remote driver imposes a limit of 128 and 16 ; the number of parameters per cpu should not exceed 16, but if you have a host with more than 128 CPUs, your program should split the request into multiple calls.

As special cases, if is NULL and is 0 and is 1, and the return value will be how many statistics are available for the given . This number may be different for of -1 than for any non-negative value, but will be the same for all non-negative . Likewise, if is NULL and is 0 and is 0, the number of cpus available to query is returned. From the host perspective, this would typically match the cpus member of [virNodeGetInfo\(\)](#), but might be less due to host cpu hotplug.

For now, is unused, and the statistics all relate to the usage from the host perspective. It is possible that a future version will support a flag that queries the cpu usage from the guest's perspective, where the maximum cpu to query would be related to [virDomainGetVcpusFlags\(\)](#) rather than [virNodeGetInfo\(\)](#). An individual guest vcpu cannot be reliably mapped back to a specific host cpu unless a single-processor vcpu pinning was used, but when is -1, any difference in usage between a host and guest perspective would serve as a measure of hypervisor overhead.

Typical use sequence is below.

```
getting total stats: set start_cpu as -1, ncpus 1 virDomainGetCPUStats(dom, NULL, 0, -1, 1, 0) => nparams
params = calloc(nparams, sizeof(virTypedParameter)) virDomainGetCPUStats(dom, params, nparams, -1, 1, 0) =>
total stats.
```

```
getting per-cpu stats: virDomainGetCPUStats(dom, NULL, 0, 0, 0, 0) => ncpus virDomainGetCPUStats(dom, N-
ULL, 0, 0, 1, 0) => nparams params = calloc(ncpus * nparams, sizeof(virTypedParameter)) virDomainGetCPU-
Stats(dom, params, nparams, 0, ncpus, 0) => per-cpu stats
```

Returns -1 on failure, or the number of statistics that were populated per cpu on success (this will be less than the total number of populated , unless was 1; and may be less than ). The populated parameters start at each stride of , which means the results may be discontinuous; any unpopulated parameters will be zeroed on success (this includes skipped elements if is too large, and tail elements if is too large). The caller is responsible for freeing any returned string parameters.

**4.1.4.87** `int virDomainGetDiskErrors ( virDomainPtr dom, virDomainDiskErrorPtr errors, unsigned int maxerrors, unsigned int flags )`

virDomainGetDiskErrors: : a domain object : array to populate on output : size of array : extra flags; not used yet, so callers should always pass 0

The function populates array with all disks that encountered an I/O error. Disks with no error will not be returned in the array. Each disk is identified by its target (the dev attribute of target subelement in domain XML), such as "vda", and accompanied with the error that was seen on it. The caller is also responsible for calling free() on each disk name returned.

In a special case when is NULL and is 0, the function returns preferred size of that the caller should use to get all disk errors.

Since calling virDomainGetDiskErrors(dom, NULL, 0, 0) to get preferred size of array and getting the errors are two separate operations, new disks may be hotplugged to the domain and new errors may be encountered between the two calls. Thus, this function may not return all disk errors because the supplied array is not large enough. Such errors may, however, be detected by listening to domain events.

Returns number of disks with errors filled in the array or -1 on error.

**4.1.4.88** `int virDomainGetEmulatorPinInfo ( virDomainPtr domain, unsigned char * cpumap, int maplen, unsigned int flags )`

virDomainGetEmulatorPinInfo: : pointer to domain object, or NULL for Domain0 : pointer to a bit map of real CPUs for all emulator threads of this domain (in 8-bit bytes) (OUT) There is only one cpumap for all emulator threads. Must not be NULL. : the number of bytes in one cpumap, from 1 up to size of CPU map. Must be positive. : bitwise-OR of virDomainModificationImpact Must not be VIR\_DOMAIN\_AFFECT\_LIVE and VIR\_DOMAIN\_AFFECT\_CONFIG concurrently.

Query the CPU affinity setting of all emulator threads of domain, store it in cpumap.

Returns 1 in case of success, 0 in case of no emulator threads are pinned to pcpus, -1 in case of failure.

**4.1.4.89** `char* virDomainGetHostname ( virDomainPtr domain, unsigned int flags )`

virDomainGetHostname: : a domain object : extra flags; not used yet, so callers should always pass 0

Get the hostname for that domain.

Dependent on hypervisor used, this may require a guest agent to be available.

Returns the hostname which must be freed by the caller, or NULL if there was an error.

**4.1.4.90** `unsigned int virDomainGetID ( virDomainPtr domain )`

virDomainGetID: : a domain object

Get the hypervisor ID number for the domain

Returns the domain ID number or (unsigned int) -1 in case of error

**4.1.4.91** `int virDomainGetInfo ( virDomainPtr domain, virDomainInfoPtr info )`

virDomainGetInfo: : a domain object : pointer to a virDomainInfo structure allocated by the user



Extract information about a domain. Note that if the connection used to get the domain is limited only a partial set of the information can be extracted.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.92** `int virDomainGetInterfaceParameters ( virDomainPtr domain, const char * device, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainGetInterfaceParameters`: : pointer to domain object : the interface name or mac address : pointer to interface parameter objects (return value, allocated by the caller) : pointer to number of interface parameter; input and output : bitwise-OR of `virDomainModificationImpact` and `virTypedParameterFlags`

Get all interface parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. The caller should then allocate array, i.e. `(sizeof() * )` bytes and call the API again. See [virDomainGetMemoryParameters\(\)](#) for an equivalent usage example.

This function may require privileged access to the hypervisor. This function expects the caller to allocate the .

Returns -1 in case of error, 0 in case of success.

**4.1.4.93** `int virDomainGetJobInfo ( virDomainPtr domain, virDomainJobInfoPtr info )`

`virDomainGetJobInfo`: : a domain object : pointer to a `virDomainJobInfo` structure allocated by the user

Extract information about progress of a background job on a domain. Will return an error if the domain is not active.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.94** `unsigned long virDomainGetMaxMemory ( virDomainPtr domain )`

`virDomainGetMaxMemory`: : a domain object or NULL

Retrieve the maximum amount of physical memory allocated to a domain. If domain is NULL, then this get the amount of memory reserved to Domain0 i.e. the domain where the application runs.

Returns the memory size in kibibytes (blocks of 1024 bytes), or 0 in case of error.

**4.1.4.95** `int virDomainGetMaxVcpus ( virDomainPtr domain )`

`virDomainGetMaxVcpus`: : pointer to domain object

Provides the maximum number of virtual CPUs supported for the guest VM. If the guest is inactive, this is basically the same as [virConnectGetMaxVcpus\(\)](#). If the guest is running this will reflect the maximum number of virtual CPUs the guest was booted with. For more details, see [virDomainGetVcpusFlags\(\)](#).

Returns the maximum of virtual CPU or -1 in case of error.

**4.1.4.96** `int virDomainGetMemoryParameters ( virDomainPtr domain, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainGetMemoryParameters`: : pointer to domain object : pointer to memory parameter object (return value, allocated by the caller) : pointer to number of memory parameters; input and output : bitwise-OR of `virDomainModificationImpact` and `virTypedParameterFlags`

Get all memory parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.



As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. The caller should then allocate array, i.e. (sizeof( \* ) bytes and call the API again.

Here is a sample code snippet:

```
if ((virDomainGetMemoryParameters(dom, NULL, &nparams, 0) == 0) && (nparams != 0)) { if ((params = malloc(sizeof(*params) * nparams)) == NULL) goto error; memset(params, 0, sizeof(*params) * nparams); if (virDomainGetMemoryParameters(dom, params, &nparams, 0)) goto error; }
```

This function may require privileged access to the hypervisor. This function expects the caller to allocate the .

Returns -1 in case of error, 0 in case of success.

#### 4.1.4.97 `char* virDomainGetMetadata ( virDomainPtr domain, int type, const char * uri, unsigned int flags )`

`virDomainGetMetadata`: : a domain object : type of description, from `virDomainMetadataType` : XML namespace identifier : bitwise-OR of `virDomainModificationImpact`

Retrieves the appropriate domain element given by . If `VIR_DOMAIN_METADATA_ELEMENT` is requested parameter must be set to the name of the namespace the requested elements belong to, otherwise must be NULL.

If an element of the domain XML is not present, the resulting error will be `VIR_ERR_NO_DOMAIN_METADATA`. This method forms a shortcut for seeing information from [virDomainSetMetadata\(\)](#) without having to go through [virDomainGetXMLDesc\(\)](#).

controls whether the live domain or persistent configuration will be queried.

Returns the metadata string on success (caller must free), or NULL in case of failure.

#### 4.1.4.98 `const char* virDomainGetName ( virDomainPtr domain )`

`virDomainGetName`: : a domain object

Get the public name for that domain

Returns a pointer to the name or NULL, the string need not be deallocated its lifetime will be the same as the domain object.

#### 4.1.4.99 `int virDomainGetNumaParameters ( virDomainPtr domain, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainGetNumaParameters`: : pointer to domain object : pointer to numa parameter object (return value, allocated by the caller) : pointer to number of numa parameters : bitwise-OR of `virDomainModificationImpact` and `virTypedParameterFlags`

Get all numa parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. The caller should then allocate array, i.e. (sizeof( \* ) bytes and call the API again.

See [virDomainGetMemoryParameters\(\)](#) for an equivalent usage example.

This function may require privileged access to the hypervisor. This function expects the caller to allocate the .

Returns -1 in case of error, 0 in case of success.

#### 4.1.4.100 `char* virDomainGetOSType ( virDomainPtr domain )`

`virDomainGetOSType`: : a domain object

Get the type of domain operation system.

Returns the new string or NULL in case of error, the string must be freed by the caller.

#### 4.1.4.101 `int virDomainGetSchedulerParameters ( virDomainPtr domain, virTypedParameterPtr params, int * nparams )`

`virDomainGetSchedulerParameters`: : pointer to domain object : pointer to scheduler parameter objects (return value) : pointer to number of scheduler parameter objects (this value should generally be as large as the returned value `nparams` of `virDomainGetSchedulerType()`); input and output

Get all scheduler parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value. cannot be 0.

It is hypervisor specific whether this returns the live or persistent state; for more control, use `virDomainGetSchedulerParametersFlags()`.

Returns -1 in case of error, 0 in case of success.

#### 4.1.4.102 `int virDomainGetSchedulerParametersFlags ( virDomainPtr domain, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainGetSchedulerParametersFlags`: : pointer to domain object : pointer to scheduler parameter object (return value) : pointer to number of scheduler parameter (this value should be same than the returned value `nparams` of `virDomainGetSchedulerType()`); input and output : bitwise-OR of `virDomainModificationImpact` and `virTypedParameterFlags`

Get all scheduler parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value. cannot be 0.

The value of can be exactly `VIR_DOMAIN_AFFECT_CURRENT`, `VIR_DOMAIN_AFFECT_LIVE`, or `VIR_DOMAIN_AFFECT_CONFIG`.

Here is a sample code snippet:

```
char *ret = virDomainGetSchedulerType(dom, &nparams); if (ret && nparams != 0) { if ((params = malloc(sizeof(*params) * nparams)) == NULL) goto error; memset(params, 0, sizeof(*params) * nparams); if (virDomainGetSchedulerParametersFlags(dom, params, &nparams, 0)) goto error; }
```

Returns -1 in case of error, 0 in case of success.

#### 4.1.4.103 `char* virDomainGetSchedulerType ( virDomainPtr domain, int * nparams )`

`virDomainGetSchedulerType`: : pointer to domain object : pointer to number of scheduler parameters, can be NULL (return value)

Get the scheduler type and the number of scheduler parameters.

Returns NULL in case of error. The caller must free the returned string.

#### 4.1.4.104 `int virDomainGetSecurityLabel ( virDomainPtr domain, virSecurityLabelPtr seclabel )`

`virDomainGetSecurityLabel`: : a domain object : pointer to a `virSecurityLabel` structure

Extract security label of an active domain. The 'label' field in the argument will be initialized to the empty string if the domain is not running under a security model.

Returns 0 in case of success, -1 in case of failure

#### 4.1.4.105 `int virDomainGetSecurityLabelList ( virDomainPtr domain, virSecurityLabelPtr * seclabels )`

`virDomainGetSecurityLabelList`: : a domain object : will be auto-allocated and filled with domains' security labels. Caller must free memory on return.

Extract the security labels of an active domain. The 'label' field in the argument will be initialized to the empty string if the domain is not running under a security model.

Returns number of elements in on success, -1 in case of failure.

#### 4.1.4.106 int virDomainGetState ( virDomainPtr domain, int \* state, int \* reason, unsigned int flags )

virDomainGetState: : a domain object : returned state of the domain (one of virDomainState) : returned reason which led to (one of virDomainReason corresponding to the current state); it is allowed to be NULL : extra flags; not used yet, so callers should always pass 0

Extract domain state. Each state can be accompanied with a reason (if known) which led to the state.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.107 int virDomainGetUUID ( virDomainPtr domain, unsigned char \* uuid )

virDomainGetUUID: : a domain object : pointer to a VIR\_UUID\_BUFLen bytes array

Get the UUID for a domain

Returns -1 in case of error, 0 in case of success

#### 4.1.4.108 int virDomainGetUUIDString ( virDomainPtr domain, char \* buf )

virDomainGetUUIDString: : a domain object : pointer to a VIR\_UUID\_STRING\_BUFLen bytes array

Get the UUID for a domain as string. For more information about UUID see RFC4122.

Returns -1 in case of error, 0 in case of success

#### 4.1.4.109 int virDomainGetVcpuPinInfo ( virDomainPtr domain, int ncpumaps, unsigned char \* cpumaps, int maplen, unsigned int flags )

virDomainGetVcpuPinInfo: : pointer to domain object, or NULL for Domain0 : the number of cpumap (listed first to match virDomainGetVcpus) : pointer to a bit map of real CPUs for all vcpus of this domain (in 8-bit bytes) (OUT) It's assumed there is <ncpumaps> cpumap in cpumaps array. The memory allocated to cpumaps must be (ncpumaps \* maplen) bytes (ie: calloc(ncpumaps, maplen)). One cpumap inside cpumaps has the format described in [virDomainPinVcpu\(\)](#) API. Must not be NULL. : the number of bytes in one cpumap, from 1 up to size of CPU map. Must be positive. : bitwise-OR of virDomainModificationImpact Must not be VIR\_DOMAIN\_AFFECT\_LIVE and VIR\_DOMAIN\_AFFECT\_CONFIG concurrently.

Query the CPU affinity setting of all virtual CPUs of domain, store it in cpumaps.

Returns the number of virtual CPUs in case of success, -1 in case of failure.

#### 4.1.4.110 int virDomainGetVcpus ( virDomainPtr domain, virVcpuInfoPtr info, int maxinfo, unsigned char \* cpumaps, int maplen )

virDomainGetVcpus: : pointer to domain object, or NULL for Domain0 : pointer to an array of virVcpuInfo structures (OUT) : number of structures in info array : pointer to a bit map of real CPUs for all vcpus of this domain (in 8-bit bytes) (OUT) If cpumaps is NULL, then no cpumap information is returned by the API. It's assumed there is <maxinfo> cpumap in cpumaps array. The memory allocated to cpumaps must be (maxinfo \* maplen) bytes (ie: calloc(maxinfo, maplen)). One cpumap inside cpumaps has the format described in [virDomainPinVcpu\(\)](#) API. : number of bytes in one cpumap, from 1 up to size of CPU map in underlying virtualization system (Xen...). Must be zero when cpumaps is NULL and positive when it is non-NULL.

Extract information about virtual CPUs of domain, store it in info array and also in cpumaps if this pointer isn't NULL. This call may fail on an inactive domain.

See also virDomainGetVcpuPinInfo for querying just cpumaps, including on an inactive domain.

Returns the number of info filled in case of success, -1 in case of failure.

#### 4.1.4.111 `int virDomainGetVcpuFlags ( virDomainPtr domain, unsigned int flags )`

`virDomainGetVcpuFlags`: : pointer to domain object, or NULL for Domain0 : bitwise-OR of `virDomainVcpuFlags`

Query the number of virtual CPUs used by the domain. Note that this call may fail if the underlying virtualization hypervisor does not support it. This function may require privileged access to the hypervisor.

If includes `VIR_DOMAIN_AFFECT_LIVE`, this will query a running domain (which will fail if domain is not active); if it includes `VIR_DOMAIN_AFFECT_CONFIG`, this will query the XML description of the domain. It is an error to set both flags. If neither flag is set (that is, `VIR_DOMAIN_AFFECT_CURRENT`), then the configuration queried depends on whether the domain is currently running.

If includes `VIR_DOMAIN_VCPU_MAXIMUM`, then the maximum virtual CPU limit is queried. Otherwise, this call queries the current virtual CPU limit.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.112 `char* virDomainGetXMLDesc ( virDomainPtr domain, unsigned int flags )`

`virDomainGetXMLDesc`: : a domain object : bitwise-OR of `virDomainXMLFlags`

Provide an XML description of the domain. The description may be reused later to relaunch the domain with [virDomainCreateXML\(\)](#).

No security-sensitive data will be included unless contains `VIR_DOMAIN_XML_SECURE`; this flag is rejected on read-only connections. If includes `VIR_DOMAIN_XML_INACTIVE`, then the XML represents the configuration that will be used on the next boot of a persistent domain; otherwise, the configuration represents the currently running domain. If contains `VIR_DOMAIN_XML_UPDATE_CPU`, then the portion of the domain XML describing CPU capabilities is modified to match actual capabilities of the host.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must `free()` the returned value.

#### 4.1.4.113 `int virDomainHasCurrentSnapshot ( virDomainPtr domain, unsigned int flags )`

`virDomainHasCurrentSnapshot`: : pointer to the domain object : extra flags; not used yet, so callers should always pass 0

Determine if the domain has a current snapshot.

Returns 1 if such snapshot exists, 0 if it doesn't, -1 on error.

#### 4.1.4.114 `int virDomainHasManagedSavImage ( virDomainPtr dom, unsigned int flags )`

`virDomainHasManagedSavImage`: : pointer to the domain : extra flags; not used yet, so callers should always pass 0

Check if a domain has a managed save image as created by [virDomainManagedSave\(\)](#). Note that any running domain should not have such an image, as it should have been removed on restart.

Returns 0 if no image is present, 1 if an image is present, and -1 in case of error

#### 4.1.4.115 `int virDomainInjectNMI ( virDomainPtr domain, unsigned int flags )`

`virDomainInjectNMI`: : pointer to domain object, or NULL for Domain0 : extra flags; not used yet, so callers should always pass 0

Send NMI to the guest

Returns 0 in case of success, -1 in case of failure.

**4.1.4.116** `int virDomainInterfaceStats ( virDomainPtr dom, const char * path, virDomainInterfaceStatsPtr stats, size_t size )`

`virDomainInterfaceStats`: : pointer to the domain object : path to the interface : network interface stats (returned) : size of stats structure

This function returns network interface stats for interfaces attached to the domain.

The path parameter is the name of the network interface.

Domains may have more than one network interface. To get stats for each you should make multiple calls to this function.

Individual fields within the stats structure may be returned as -1, which indicates that the hypervisor does not support that particular statistic.

Returns: 0 in case of success or -1 in case of failure.

**4.1.4.117** `int virDomainIsActive ( virDomainPtr dom )`

`virDomainIsActive`: : pointer to the domain object

Determine if the domain is currently running

Returns 1 if running, 0 if inactive, -1 on error

**4.1.4.118** `int virDomainIsPersistent ( virDomainPtr dom )`

`virDomainIsPersistent`: : pointer to the domain object

Determine if the domain has a persistent configuration which means it will still exist after shutting down

Returns 1 if persistent, 0 if transient, -1 on error

**4.1.4.119** `int virDomainIsUpdated ( virDomainPtr dom )`

`virDomainIsUpdated`: : pointer to the domain object

Determine if the domain has been updated.

Returns 1 if updated, 0 if not, -1 on error

**4.1.4.120** `int virDomainListAllSnapshots ( virDomainPtr domain, virDomainSnapshotPtr ** snaps, unsigned int flags )`

`virDomainListAllSnapshots`: : a domain object : pointer to variable to store the array containing snapshot objects, or NULL if the list is not required (just returns number of snapshots) : bitwise-OR of supported `virDomainSnapshotListFlags`

Collect the list of domain snapshots for the given domain, and allocate an array to store those objects. This API solves the race inherent in [virDomainSnapshotListNames\(\)](#).

By default, this command covers all snapshots; it is also possible to limit things to just snapshots with no parents, when includes `VIR_DOMAIN_SNAPSHOT_LIST_ROOTS`. Additional filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virDomainSnapshotFree\(\)](#) on each array element, then calling `free()` on .

#### 4.1.4.121 `virDomainPtr virDomainLookupByID ( virConnectPtr conn, int id )`

`virDomainLookupByID`: : pointer to the hypervisor connection : the domain ID number

Try to find a domain based on the hypervisor ID number Note that this won't work for inactive domains which have an ID of -1, in that case a lookup based on the Name or UUID need to be done instead.

Returns a new domain object or NULL in case of failure. If the domain cannot be found, then `VIR_ERR_NO_DOMAIN` error is raised.

#### 4.1.4.122 `virDomainPtr virDomainLookupByName ( virConnectPtr conn, const char * name )`

#### 4.1.4.123 `virDomainPtr virDomainLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

`virDomainLookupByUUID`: : pointer to the hypervisor connection : the raw UUID for the domain

Try to lookup a domain on the given hypervisor based on its UUID.

Returns a new domain object or NULL in case of failure. If the domain cannot be found, then `VIR_ERR_NO_DOMAIN` error is raised.

#### 4.1.4.124 `virDomainPtr virDomainLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

`virDomainLookupByUUIDString`: : pointer to the hypervisor connection : the string UUID for the domain

Try to lookup a domain on the given hypervisor based on its UUID.

Returns a new domain object or NULL in case of failure. If the domain cannot be found, then `VIR_ERR_NO_DOMAIN` error is raised.

#### 4.1.4.125 `int virDomainManagedSave ( virDomainPtr dom, unsigned int flags )`

`virDomainManagedSave`: : pointer to the domain : bitwise-OR of `virDomainSaveRestoreFlags`

This method will suspend a domain and save its memory contents to a file on disk. After the call, if successful, the domain is not listed as running anymore. The difference from [virDomainSave\(\)](#) is that libvirt is keeping track of the saved state itself, and will reuse it once the domain is being restarted (automatically or via an explicit libvirt call). As a result any running domain is sure to not have a managed saved image. This also implies that managed save only works on persistent domains, since the domain must still exist in order to use [virDomainCreate\(\)](#) to restart it.

If includes `VIR_DOMAIN_SAVE_BYPASS_CACHE`, then libvirt will attempt to bypass the file system cache while creating the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing saves to NFS.

Normally, the managed saved state will remember whether the domain was running or paused, and start will resume to the same state. Specifying `VIR_DOMAIN_SAVE_RUNNING` or `VIR_DOMAIN_SAVE_PAUSED` in will override the default saved into the file. These two flags are mutually exclusive.

Returns 0 in case of success or -1 in case of failure

## 4.1.4.126 int virDomainManagedSaveRemove ( virDomainPtr dom, unsigned int flags )

virDomainManagedSaveRemove: : pointer to the domain : extra flags; not used yet, so callers should always pass 0

Remove any managed save image for this domain.

Returns 0 in case of success, and -1 in case of error

## 4.1.4.127 int virDomainMemoryPeek ( virDomainPtr dom, unsigned long long start, size\_t size, void \* buffer, unsigned int flags )

virDomainMemoryPeek: : pointer to the domain object : start of memory to peek : size of memory to peek : return buffer (must be at least size bytes) : bitwise-OR of virDomainMemoryFlags

This function allows you to read the contents of a domain's memory.

The memory which is read is controlled by the 'start', 'size' and 'flags' parameters.

If 'flags' is VIR\_MEMORY\_VIRTUAL then the 'start' and 'size' parameters are interpreted as virtual memory addresses for whichever task happens to be running on the domain at the moment. Although this sounds haphazard it is in fact what you want in order to read Linux kernel state, because it ensures that pointers in the kernel image can be interpreted coherently.

'buffer' is the return buffer and must be at least 'size' bytes. 'size' may be 0 to test if the call would succeed.

NB. The remote driver imposes a 64K byte limit on 'size'. For your program to be able to work reliably over a remote connection you should split large requests to <= 65536 bytes. However, with 0.9.13 this RPC limit has been raised to 1M byte.

Returns: 0 in case of success or -1 in case of failure.

## 4.1.4.128 int virDomainMemoryStats ( virDomainPtr dom, virDomainMemoryStatPtr stats, unsigned int nr\_stats, unsigned int flags )

virDomainMemoryStats: : pointer to the domain object : nr\_stats-sized array of stat structures (returned) : number of memory statistics requested : extra flags; not used yet, so callers should always pass 0

This function provides memory statistics for the domain.

Up to 'nr\_stats' elements of 'stats' will be populated with memory statistics from the domain. Only statistics supported by the domain, the driver, and this version of libvirt will be returned.

Memory Statistics:

VIR\_DOMAIN\_MEMORY\_STAT\_SWAP\_IN: The total amount of data read from swap space (in kb). VIR\_DOMAIN\_MEMORY\_STAT\_SWAP\_OUT: The total amount of memory written out to swap space (in kb). VIR\_DOMAIN\_MEMORY\_STAT\_MAJOR\_FAULT: The number of page faults that required disk IO to service. VIR\_DOMAIN\_MEMORY\_STAT\_MINOR\_FAULT: The number of page faults serviced without disk IO. VIR\_DOMAIN\_MEMORY\_STAT\_UNUSED: The amount of memory which is not being used for any purpose (in kb). VIR\_DOMAIN\_MEMORY\_STAT\_AVAILABLE: The total amount of memory available to the domain's OS (in kb). VIR\_DOMAIN\_MEMORY\_STAT\_ACTUAL\_BALLOON: Current balloon value (in kb).

Returns: The number of stats provided or -1 in case of failure.

## 4.1.4.129 virDomainPtr virDomainMigrate ( virDomainPtr domain, virConnectPtr dconn, unsigned long flags, const char \* dname, const char \* uri, unsigned long bandwidth )

virDomainMigrate: : a domain object : destination host (a connection object) : bitwise-OR of virDomainMigrate-Flags : (optional) rename domain to this at destination : (optional) dest hostname/URI as seen from the source host : (optional) specify migration bandwidth limit in Mbps

Migrate the domain object from its current host to the destination host given by *dconn* (a connection to the destination host).

Flags may be one or more of the following: **VIR\_MIGRATE\_LIVE** Do not pause the VM during migration **VIR\_MIGRATE\_PEER2PEER** Direct connection between source & destination hosts **VIR\_MIGRATE\_TUNNELLED** Tunnel migration data over the libvirt RPC channel **VIR\_MIGRATE\_PERSIST\_DEST** If the migration is successful, persist the domain on the destination host. **VIR\_MIGRATE\_UNDEFINE\_SOURCE** If the migration is successful, undefine the domain on the source host. **VIR\_MIGRATE\_PAUSED** Leave the domain suspended on the remote side. **VIR\_MIGRATE\_CHANGE\_PROTECTION** Protect against domain configuration changes during the migration process (set automatically when supported). **VIR\_MIGRATE\_UNSAFE** Force migration even if it is considered unsafe.

**VIR\_MIGRATE\_TUNNELLED** requires that **VIR\_MIGRATE\_PEER2PEER** be set. Applications using the **VIR\_MIGRATE\_PEER2PEER** flag will probably prefer to invoke `virDomainMigrateToURI`, avoiding the need to open connection to the destination host themselves.

If a hypervisor supports renaming domains during migration, then you may set the *dname* parameter to the new name (otherwise it keeps the same name). If this is not supported by the hypervisor, *dname* must be `NULL` or else you will get an error.

If the **VIR\_MIGRATE\_PEER2PEER** flag is set, the *uri* parameter must be a valid libvirt connection URI, by which the source libvirt driver can connect to the destination libvirt. If omitted, the *dconn* connection object will be queried for its current URI.

If the **VIR\_MIGRATE\_PEER2PEER** flag is NOT set, the *uri* parameter takes a hypervisor specific format. The hypervisor capabilities XML includes details of the support URI schemes. If omitted the *dconn* will be asked for a default URI.

In either case it is typically only necessary to specify a URI if the destination host has multiple interfaces and a specific interface is required to transmit migration data.

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the *bandwidth* parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if *bandwidth* is not 0.

To see which features are supported by the current hypervisor, see `virConnectGetCapabilities`, `/capabilities/host/migration-features`.

There are many limitations on migration imposed by the underlying technology - for example it may not be possible to migrate between different processors even with the same architecture, or between different types of hypervisor.

Returns the new domain object if the migration was successful, or `NULL` in case of error. Note that the new domain object exists in the scope of the destination connection (*dconn*).

**4.1.4.130** `virDomainPtr virDomainMigrate2 ( virDomainPtr domain, virConnectPtr dconn, const char * dxml, unsigned long flags, const char * dname, const char * uri, unsigned long bandwidth )`

`virDomainMigrate2`: : a domain object : destination host (a connection object) : bitwise-OR of `virDomainMigrateFlags` : (optional) XML config for launching guest on target : (optional) rename domain to this at destination : (optional) dest hostname/URI as seen from the source host : (optional) specify migration bandwidth limit in Mbps

Migrate the domain object from its current host to the destination host given by *dconn* (a connection to the destination host).

Flags may be one or more of the following: **VIR\_MIGRATE\_LIVE** Do not pause the VM during migration **VIR\_MIGRATE\_PEER2PEER** Direct connection between source & destination hosts **VIR\_MIGRATE\_TUNNELLED** Tunnel migration data over the libvirt RPC channel **VIR\_MIGRATE\_PERSIST\_DEST** If the migration is successful, persist the domain on the destination host. **VIR\_MIGRATE\_UNDEFINE\_SOURCE** If the migration is successful, undefine the domain on the source host. **VIR\_MIGRATE\_PAUSED** Leave the domain suspended on the remote side. **VIR\_MIGRATE\_CHANGE\_PROTECTION** Protect against domain configuration changes during the migration process (set automatically when supported). **VIR\_MIGRATE\_UNSAFE** Force migration even if it is considered unsafe.

**VIR\_MIGRATE\_TUNNELLED** requires that **VIR\_MIGRATE\_PEER2PEER** be set. Applications using the **VIR\_MIGRATE\_PEER2PEER** flag will probably prefer to invoke `virDomainMigrateToURI`, avoiding the need to open connection to the destination host themselves.



If a hypervisor supports renaming domains during migration, then you may set the `dname` parameter to the new name (otherwise it keeps the same name). If this is not supported by the hypervisor, `dname` must be `NULL` or else you will get an error.

If the `VIR_MIGRATE_PEER2PEER` flag is set, the `uri` parameter must be a valid libvirt connection URI, by which the source libvirt driver can connect to the destination libvirt. If omitted, the `dconn` connection object will be queried for its current URI.

If the `VIR_MIGRATE_PEER2PEER` flag is NOT set, the URI parameter takes a hypervisor specific format. The hypervisor capabilities XML includes details of the support URI schemes. If omitted the `dconn` will be asked for a default URI.

In either case it is typically only necessary to specify a URI if the destination host has multiple interfaces and a specific interface is required to transmit migration data.

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the `bandwidth` parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0.

To see which features are supported by the current hypervisor, see `virConnectGetCapabilities`, `/capabilities/host/migration-features`.

There are many limitations on migration imposed by the underlying technology - for example it may not be possible to migrate between different processors even with the same architecture, or between different types of hypervisor.

If the hypervisor supports it, can be used to alter host-specific portions of the domain XML that will be used on the destination. For example, it is possible to alter the backing filename that is associated with a disk device, in order to account for naming differences between source and destination in accessing the underlying storage. The migration will fail if would cause any guest-visible changes. Pass `NULL` if no changes are needed to the XML between source and destination. cannot be used to rename the domain during migration (use for that purpose). Domain name in must match the original domain name.

Returns the new domain object if the migration was successful, or `NULL` in case of error. Note that the new domain object exists in the scope of the destination connection (`dconn`).

#### 4.1.4.131 `int virDomainMigrateGetMaxSpeed ( virDomainPtr domain, unsigned long * bandwidth, unsigned int flags )`

`virDomainMigrateGetMaxSpeed`: : a domain object : return value of current migration bandwidth limit in Mbps : extra flags; not used yet, so callers should always pass 0

Get the current maximum bandwidth (in Mbps) that will be used if the domain is migrated. Not all hypervisors will support a bandwidth limit.

Returns 0 in case of success, -1 otherwise.

#### 4.1.4.132 `int virDomainMigrateSetMaxDowntime ( virDomainPtr domain, unsigned long long downtime, unsigned int flags )`

`virDomainMigrateSetMaxDowntime`: : a domain object : maximum tolerable downtime for live migration, in milliseconds : extra flags; not used yet, so callers should always pass 0

Sets maximum tolerable time for which the domain is allowed to be paused at the end of live migration. It's supposed to be called while the domain is being live-migrated as a reaction to migration progress.

Returns 0 in case of success, -1 otherwise.

#### 4.1.4.133 `int virDomainMigrateSetMaxSpeed ( virDomainPtr domain, unsigned long bandwidth, unsigned int flags )`

`virDomainMigrateSetMaxSpeed`: : a domain object : migration bandwidth limit in Mbps : extra flags; not used yet, so callers should always pass 0

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the `bandwidth` parameter. Not all hypervisors will support a bandwidth cap

Returns 0 in case of success, -1 otherwise.

**4.1.4.134** `int virDomainMigrateToURI ( virDomainPtr domain, const char * duri, unsigned long flags, const char * dname, unsigned long bandwidth )`

`virDomainMigrateToURI`: : a domain object : mandatory URI for the destination host : bitwise-OR of `virDomainMigrateFlags` : (optional) rename domain to this at destination : (optional) specify migration bandwidth limit in Mbps

Migrate the domain object from its current host to the destination host given by `duri`.

Flags may be one of more of the following: `VIR_MIGRATE_LIVE` Do not pause the VM during migration `VIR_MIGRATE_PEER2PEER` Direct connection between source & destination hosts `VIR_MIGRATE_TUNNELLED` Tunnel migration data over the libvirt RPC channel `VIR_MIGRATE_PERSIST_DEST` If the migration is successful, persist the domain on the destination host. `VIR_MIGRATE_UNDEFINE_SOURCE` If the migration is successful, undefine the domain on the source host. `VIR_MIGRATE_CHANGE_PROTECTION` Protect against domain configuration changes during the migration process (set automatically when supported). `VIR_MIGRATE_UNSAFE` Force migration even if it is considered unsafe.

The operation of this API hinges on the `VIR_MIGRATE_PEER2PEER` flag. If the `VIR_MIGRATE_PEER2PEER` flag is NOT set, the `duri` parameter takes a hypervisor specific format. The `uri_transports` element of the hypervisor capabilities XML includes details of the supported URI schemes. Not all hypervisors will support this mode of migration, so if the `VIR_MIGRATE_PEER2PEER` flag is not set, then it may be necessary to use the alternative `virDomainMigrate` API providing and explicit `virConnectPtr` for the destination host.

If the `VIR_MIGRATE_PEER2PEER` flag IS set, the `duri` parameter must be a valid libvirt connection URI, by which the source libvirt driver can connect to the destination libvirt.

`VIR_MIGRATE_TUNNELLED` requires that `VIR_MIGRATE_PEER2PEER` be set.

If a hypervisor supports renaming domains during migration, the `dname` parameter specifies the new name for the domain. Setting `dname` to NULL keeps the domain name the same. If domain renaming is not supported by the hypervisor, `dname` must be NULL or else an error will be returned.

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the `bandwidth` parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0.

To see which features are supported by the current hypervisor, see `virConnectGetCapabilities`, `/capabilities/host/migration_features`.

There are many limitations on migration imposed by the underlying technology - for example it may not be possible to migrate between different processors even with the same architecture, or between different types of hypervisor.

Returns 0 if the migration succeeded, -1 upon error.

**4.1.4.135** `int virDomainMigrateToURI2 ( virDomainPtr domain, const char * dconnuri, const char * miguri, const char * dxml, unsigned long flags, const char * dname, unsigned long bandwidth )`

`virDomainMigrateToURI2`: : a domain object : (optional) URI for target libvirtd if includes `VIR_MIGRATE_PEER2PEER` : (optional) URI for invoking the migration, not if includes `VIR_MIGRATE_TUNNELLED` : (optional) XML config for launching guest on target : bitwise-OR of `virDomainMigrateFlags` : (optional) rename domain to this at destination : (optional) specify migration bandwidth limit in Mbps

Migrate the domain object from its current host to the destination host given by `duri`.

Flags may be one of more of the following: `VIR_MIGRATE_LIVE` Do not pause the VM during migration `VIR_MIGRATE_PEER2PEER` Direct connection between source & destination hosts `VIR_MIGRATE_TUNNELLED` Tunnel migration data over the libvirt RPC channel `VIR_MIGRATE_PERSIST_DEST` If the migration is successful, persist the domain on the destination host. `VIR_MIGRATE_UNDEFINE_SOURCE` If the migration is successful, undefine the domain on the source host. `VIR_MIGRATE_CHANGE_PROTECTION` Protect against domain configuration changes during the migration process (set automatically when supported). `VIR_MIGRATE_UNSAFE` Force migration even if it is considered unsafe.

The operation of this API hinges on the `VIR_MIGRATE_PEER2PEER` flag.

If the `VIR_MIGRATE_PEER2PEER` flag is set, the parameter must be a valid libvirt connection URI, by which the source libvirt driver can connect to the destination libvirt. If the `VIR_MIGRATE_PEER2PEER` flag is NOT set, then must be NULL.

If the `VIR_MIGRATE_TUNNELLED` flag is NOT set, then the parameter allows specification of a URI to use to initiate the VM migration. It takes a hypervisor specific format. The `uri_transports` element of the hypervisor capabilities XML includes details of the supported URI schemes.

`VIR_MIGRATE_TUNNELLED` requires that `VIR_MIGRATE_PEER2PEER` be set.

If a hypervisor supports changing the configuration of the guest during migration, the parameter specifies the new config for the guest. The configuration must include an identical set of virtual devices, to ensure a stable guest ABI across migration. Only parameters related to host side configuration can be changed in the XML. Hypervisors will validate this and refuse to allow migration if the provided XML would cause a change in the guest ABI,

If a hypervisor supports renaming domains during migration, the `dname` parameter specifies the new name for the domain. Setting `dname` to NULL keeps the domain name the same. If domain renaming is not supported by the hypervisor, `dname` must be NULL or else an error will be returned.

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the `bandwidth` parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0.

To see which features are supported by the current hypervisor, see `virConnectGetCapabilities`, `/capabilities/host/migration-features`.

There are many limitations on migration imposed by the underlying technology - for example it may not be possible to migrate between different processors even with the same architecture, or between different types of hypervisor.

Returns 0 if the migration succeeded, -1 upon error.

#### 4.1.4.136 `int virDomainOpenConsole ( virDomainPtr dom, const char * dev_name, virStreamPtr st, unsigned int flags )`

`virDomainOpenConsole`: : a domain object : the console, serial or parallel port device alias, or NULL : a stream to associate with the console : bitwise-OR of `virDomainConsoleFlags`

This opens the backend associated with a console, serial or parallel port device on a guest, if the backend is supported. If the is omitted, then the first console or serial device is opened. The console is associated with the passed in stream, which should have been opened in non-blocking mode for bi-directional I/O.

By default, when is 0, the open will fail if libvirt detects that the console is already in use by another client; passing `VIR_DOMAIN_CONSOLE_FORCE` will cause libvirt to forcefully remove the other client prior to opening this console.

If flag `VIR_DOMAIN_CONSOLE_SAFE` the console is opened only in the case where the hypervisor driver supports safe (mutually exclusive) console handling.

Older servers did not support either flag, and also did not forbid simultaneous clients on a console, with potentially confusing results. When passing of 0 in order to support a wider range of server versions, it is up to the client to ensure mutual exclusion.

Returns 0 if the console was opened, -1 on error

#### 4.1.4.137 `int virDomainOpenGraphics ( virDomainPtr dom, unsigned int idx, int fd, unsigned int flags )`

`virDomainOpenGraphics`: : pointer to domain object : index of graphics config to open : file descriptor to attach graphics to : bitwise-OR of `virDomainOpenGraphicsFlags`

This will attempt to connect the file descriptor , to the graphics backend of . If has multiple graphics backends configured, then will determine which one is opened, starting from 0.

To disable any authentication, pass the `VIR_DOMAIN_OPEN_GRAPHICS_SKIPAUTH` constant for .

The caller should use an anonymous socketpair to open before invocation.

This method can only be used when connected to a local libvirt hypervisor, over a UNIX domain socket. Attempts to use this method over a TCP connection will always fail

Returns 0 on success, -1 on failure

#### 4.1.4.138 int virDomainPinEmulator ( virDomainPtr domain, unsigned char \* cpumap, int maplen, unsigned int flags )

virDomainPinEmulator: : pointer to domain object, or NULL for Domain0 : pointer to a bit map of real CPUs (in 8-bit bytes) (IN) Each bit set to 1 means that corresponding CPU is usable. Bytes are stored in little-endian order: CPU0-7, 8-15... In each byte, lowest CPU number is least significant bit. : number of bytes in cpumap, from 1 up to size of CPU map in underlying virtualization system (Xen...). If maplen < size, missing bytes are set to zero. If maplen > size, failure code is returned. : bitwise-OR of virDomainModificationImpact

Dynamically change the real CPUs which can be allocated to all emulator threads. This function may require privileged access to the hypervisor.

may include VIR\_DOMAIN\_AFFECT\_LIVE or VIR\_DOMAIN\_AFFECT\_CONFIG. Both flags may be set. If VIR\_DOMAIN\_AFFECT\_LIVE is set, the change affects a running domain and may fail if domain is not alive. If VIR\_DOMAIN\_AFFECT\_CONFIG is set, the change affects persistent state, and will fail for transient domains. If neither flag is specified (that is, is VIR\_DOMAIN\_AFFECT\_CURRENT), then an inactive domain modifies persistent setup, while an active domain is hypervisor-dependent on whether just live or both live and persistent state is changed. Not all hypervisors can support all flag combinations.

See also virDomainGetEmulatorPinInfo for querying this information.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.139 int virDomainPinVcpu ( virDomainPtr domain, unsigned int vcpu, unsigned char \* cpumap, int maplen )

virDomainPinVcpu: : pointer to domain object, or NULL for Domain0 : virtual CPU number : pointer to a bit map of real CPUs (in 8-bit bytes) (IN) Each bit set to 1 means that corresponding CPU is usable. Bytes are stored in little-endian order: CPU0-7, 8-15... In each byte, lowest CPU number is least significant bit. : number of bytes in cpumap, from 1 up to size of CPU map in underlying virtualization system (Xen...). If maplen < size, missing bytes are set to zero. If maplen > size, failure code is returned.

Dynamically change the real CPUs which can be allocated to a virtual CPU. This function may require privileged access to the hypervisor.

This command only changes the runtime configuration of the domain, so can only be called on an active domain.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.140 int virDomainPinVcpuFlags ( virDomainPtr domain, unsigned int vcpu, unsigned char \* cpumap, int maplen, unsigned int flags )

virDomainPinVcpuFlags: : pointer to domain object, or NULL for Domain0 : virtual CPU number : pointer to a bit map of real CPUs (in 8-bit bytes) (IN) Each bit set to 1 means that corresponding CPU is usable. Bytes are stored in little-endian order: CPU0-7, 8-15... In each byte, lowest CPU number is least significant bit. : number of bytes in cpumap, from 1 up to size of CPU map in underlying virtualization system (Xen...). If maplen < size, missing bytes are set to zero. If maplen > size, failure code is returned. : bitwise-OR of virDomainModificationImpact

Dynamically change the real CPUs which can be allocated to a virtual CPU. This function may require privileged access to the hypervisor.

may include VIR\_DOMAIN\_AFFECT\_LIVE or VIR\_DOMAIN\_AFFECT\_CONFIG. Both flags may be set. If VIR\_DOMAIN\_AFFECT\_LIVE is set, the change affects a running domain and may fail if domain is not alive. If VIR\_DOMAIN\_AFFECT\_CONFIG is set, the change affects persistent state, and will fail for transient domains. If neither flag is specified (that is, is VIR\_DOMAIN\_AFFECT\_CURRENT), then an inactive domain modifies persistent setup, while an active domain is hypervisor-dependent on whether just live or both live and persistent state is changed. Not all hypervisors can support all flag combinations.

See also virDomainGetVcpuPinInfo for querying this information.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.141** int virDomainPMSuspendForDuration ( virDomainPtr dom, unsigned int target, unsigned long long duration, unsigned int flags )

virDomainPMSuspendForDuration: : a domain object : a value from virNodeSuspendTarget : duration in seconds to suspend, or 0 for indefinite : extra flags; not used yet, so callers should always pass 0

Attempt to have the guest enter the given power management suspension level. If is non-zero, also schedule the guest to resume normal operation after that many seconds, if nothing else has resumed it earlier. Some hypervisors require that be 0, for an indefinite suspension.

Dependent on hypervisor used, this may require a guest agent to be available, e.g. QEMU.

Returns: 0 on success, -1 on failure.

**4.1.4.142** int virDomainPMWakeup ( virDomainPtr dom, unsigned int flags )

virDomainPMWakeup: : a domain object : extra flags; not used yet, so callers should always pass 0

Inject a wakeup into the guest that previously used virDomainPMSuspendForDuration, rather than waiting for the previously requested duration (if any) to elapse.

Returns: 0 on success, -1 on failure.

**4.1.4.143** int virDomainReboot ( virDomainPtr domain, unsigned int flags )

virDomainReboot: : a domain object : bitwise-OR of virDomainRebootFlagValues

Reboot a domain, the domain object is still usable there after but the domain OS is being stopped for a restart. Note that the guest OS may ignore the request.

If is set to zero, then the hypervisor will choose the method of shutdown it considers best. To have greater control pass exactly one of the virDomainRebootFlagValues.

To use guest agent (VIR\_DOMAIN\_REBOOT\_GUEST\_AGENT) the domain XML must have <channel> configured.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.144** int virDomainRef ( virDomainPtr domain )

virDomainRef: : the domain to hold a reference on

Increment the reference count on the domain. For each additional call to this method, there shall be a corresponding call to virDomainFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a domain would increment the reference count.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.145** int virDomainReset ( virDomainPtr domain, unsigned int flags )

virDomainReset: : a domain object : extra flags; not used yet, so callers should always pass 0

Reset a domain immediately without any guest OS shutdown. Reset emulates the power reset button on a machine, where all hardware sees the RST line set and reinitializes internal state.

Note that there is a risk of data loss caused by reset without any guest OS shutdown.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.146 `int virDomainRestore ( virConnectPtr conn, const char * from )`

`virDomainRestore`: : pointer to the hypervisor connection : path to the input file

This method will restore a domain saved to disk by [virDomainSave\(\)](#).

See [virDomainRestoreFlags\(\)](#) for more control.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.147 `int virDomainRestoreFlags ( virConnectPtr conn, const char * from, const char * dxml, unsigned int flags )`

`virDomainRestoreFlags`: : pointer to the hypervisor connection : path to the input file : (optional) XML config for adjusting guest xml used on restore : bitwise-OR of `virDomainSaveRestoreFlags`

This method will restore a domain saved to disk by [virDomainSave\(\)](#).

If the hypervisor supports it, can be used to alter host-specific portions of the domain XML that will be used when restoring an image. For example, it is possible to alter the backing filename that is associated with a disk device, in order to prepare for file renaming done as part of backing up the disk device while the domain is stopped.

If includes `VIR_DOMAIN_SAVE_BYPASS_CACHE`, then libvirt will attempt to bypass the file system cache while restoring the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing saves to NFS.

Normally, the saved state file will remember whether the domain was running or paused, and restore defaults to the same state. Specifying `VIR_DOMAIN_SAVE_RUNNING` or `VIR_DOMAIN_SAVE_PAUSED` in will override the default read from the file. These two flags are mutually exclusive.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.148 `int virDomainResume ( virDomainPtr domain )`

`virDomainResume`: : a domain object

Resume a suspended domain, the process is restarted from the state where it was frozen by calling [virDomainSuspend\(\)](#). This function may require privileged access. Moreover, resume may not be supported if domain is in some special state like `VIR_DOMAIN_PMSUSPENDED`.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.149 `int virDomainRevertToSnapshot ( virDomainSnapshotPtr snapshot, unsigned int flags )`

`virDomainRevertToSnapshot`: : a domain snapshot object : bitwise-OR of `virDomainSnapshotRevertFlags`

Revert the domain to a given snapshot.

Normally, the domain will revert to the same state the domain was in while the snapshot was taken (whether inactive, running, or paused), except that disk snapshots default to reverting to inactive state. Including `VIR_DOMAIN_SNAPSHOT_REVERT_RUNNING` in overrides the snapshot state to guarantee a running domain after the revert; or including `VIR_DOMAIN_SNAPSHOT_REVERT_PAUSED` in guarantees a paused domain after the revert. These two flags are mutually exclusive. While a persistent domain does not need either flag, it is not possible to revert a transient domain into an inactive state, so transient domains require the use of one of these two flags.

Reverting to any snapshot discards all configuration changes made since the last snapshot. Additionally, reverting to a snapshot from a running domain is a form of data loss, since it discards whatever is in the guest's RAM at the time. Since the very nature of keeping snapshots implies the intent to roll back state, no additional confirmation is normally required for these lossy effects.

However, there are two particular situations where reverting will be refused by default, and where must include `VIR_DOMAIN_SNAPSHOT_REVERT_FORCE` to acknowledge the risks. 1) Any attempt to revert to a snapshot that lacks the metadata to perform ABI compatibility checks (generally the case for snapshots that lack a full `<domain>` when listed by [virDomainSnapshotGetXMLDesc\(\)](#), such as those created prior to libvirt 0.9.5). 2) Any attempt to

revert a running domain to an active state that requires starting a new hypervisor instance rather than reusing the existing hypervisor (since this would terminate all connections to the domain, such as VNC or Spice graphics) - this condition arises from active snapshots that are provably ABI incompatible, as well as from inactive snapshots with a request to start the domain after the revert.

Returns 0 if the creation is successful, -1 on error.

#### 4.1.4.150 `int virDomainSave ( virDomainPtr domain, const char * to )`

`virDomainSave`: : a domain object : path for the output file

This method will suspend a domain and save its memory contents to a file on disk. After the call, if successful, the domain is not listed as running anymore (this ends the life of a transient domain). Use [virDomainRestore\(\)](#) to restore a domain after saving.

See [virDomainSaveFlags\(\)](#) for more control. Also, a save file can be inspected or modified slightly with [virDomainSaveImageGetXMLDesc\(\)](#) and [virDomainSaveImageDefineXML\(\)](#).

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.151 `int virDomainSaveFlags ( virDomainPtr domain, const char * to, const char * dxml, unsigned int flags )`

`virDomainSaveFlags`: : a domain object : path for the output file : (optional) XML config for adjusting guest xml used on restore : bitwise-OR of `virDomainSaveRestoreFlags`

This method will suspend a domain and save its memory contents to a file on disk. After the call, if successful, the domain is not listed as running anymore (this ends the life of a transient domain). Use [virDomainRestore\(\)](#) to restore a domain after saving.

If the hypervisor supports it, can be used to alter host-specific portions of the domain XML that will be used when restoring an image. For example, it is possible to alter the backing filename that is associated with a disk device, in order to prepare for file renaming done as part of backing up the disk device while the domain is stopped.

If includes `VIR_DOMAIN_SAVE_BYPASS_CACHE`, then libvirt will attempt to bypass the file system cache while creating the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing saves to NFS.

Normally, the saved state file will remember whether the domain was running or paused, and restore defaults to the same state. Specifying `VIR_DOMAIN_SAVE_RUNNING` or `VIR_DOMAIN_SAVE_PAUSED` in will override what state gets saved into the file. These two flags are mutually exclusive.

A save file can be inspected or modified slightly with [virDomainSaveImageGetXMLDesc\(\)](#) and [virDomainSaveImageDefineXML\(\)](#).

Some hypervisors may prevent this operation if there is a current block copy operation; in that case, use [virDomainBlockJobAbort\(\)](#) to stop the block copy first.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.152 `int virDomainSaveImageDefineXML ( virConnectPtr conn, const char * file, const char * dxml, unsigned int flags )`

#### 4.1.4.153 `char* virDomainSaveImageGetXMLDesc ( virConnectPtr conn, const char * file, unsigned int flags )`

#### 4.1.4.154 `char* virDomainScreenshot ( virDomainPtr domain, virStreamPtr stream, unsigned int screen, unsigned int flags )`

`virDomainScreenshot`: : a domain object : stream to use as output : monitor ID to take screenshot from : extra flags; not used yet, so callers should always pass 0

Take a screenshot of current domain console as a stream. The image format is hypervisor specific. Moreover, some hypervisors supports multiple displays per domain. These can be distinguished by argument.

This call sets up a stream; subsequent use of stream API is necessary to transfer actual data, determine how much data is successfully transferred, and detect any errors.

The screen ID is the sequential number of screen. In case of multiple graphics cards, heads are enumerated before devices, e.g. having two graphics cards, both with four heads, screen ID 5 addresses the second head on the second card.

Returns a string representing the mime-type of the image format, or NULL upon error. The caller must free() the returned value.

**4.1.4.155** `int virDomainSendKey ( virDomainPtr domain, unsigned int codeset, unsigned int holdtime, unsigned int * keycodes, int nkeycodes, unsigned int flags )`

virDomainSendKey: : pointer to domain object, or NULL for Domain0 : the code set of keycodes, from virKeycode-Set : the duration (in milliseconds) that the keys will be held : array of keycodes : number of keycodes, up to VIR\_DOMAIN\_SEND\_KEY\_MAX\_KEYS : extra flags; not used yet, so callers should always pass 0

Send key(s) to the guest.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.156** `int virDomainSetAutostart ( virDomainPtr domain, int autostart )`

virDomainSetAutostart: : a domain object : whether the domain should be automatically started 0 or 1

Configure the domain to be automatically started when the host machine boots.

Returns -1 in case of error, 0 in case of success

**4.1.4.157** `int virDomainSetBlkioParameters ( virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags )`

virDomainSetBlkioParameters: : pointer to domain object : pointer to blkio parameter objects : number of blkio parameters (this value can be the same or less than the number of parameters supported) : bitwise-OR of virDomainModificationImpact

Change all or a subset of the blkio tunables. This function may require privileged access to the hypervisor.

Returns -1 in case of error, 0 in case of success.

**4.1.4.158** `int virDomainSetBlockIoTune ( virDomainPtr dom, const char * disk, virTypedParameterPtr params, int nparams, unsigned int flags )`

virDomainSetBlockIoTune: : pointer to domain object : path to the block device, or device shorthand : Pointer to blkio parameter objects : Number of blkio parameters (this value can be the same or less than the number of parameters supported) : bitwise-OR of virDomainModificationImpact

Change all or a subset of the per-device block IO tunables.

The parameter is either an unambiguous source name of the block device (the <source file="..."> sub-element, such as "/path/to/image"), or the device target shorthand (the <target dev="..."> sub-element, such as "xvda"). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within //domain/devices/disk.

Returns -1 in case of error, 0 in case of success.

**4.1.4.159** `int virDomainSetInterfaceParameters ( virDomainPtr domain, const char * device, virTypedParameterPtr params, int nparams, unsigned int flags )`

virDomainSetInterfaceParameters: : pointer to domain object : the interface name or mac address : pointer to interface parameter objects : number of interface parameter (this value can be the same or less than the number of



parameters supported) : bitwise-OR of virDomainModificationImpact

Change a subset or all parameters of interface; currently this includes bandwidth parameters. The value of should be either VIR\_DOMAIN\_AFFECT\_CURRENT, or a bitwise-or of values VIR\_DOMAIN\_AFFECT\_LIVE and VIR\_DOMAIN\_AFFECT\_CONFIG, although hypervisors vary in which flags are supported.

This function may require privileged access to the hypervisor.

Returns -1 in case of error, 0 in case of success.

#### 4.1.4.160 int virDomainSetMaxMemory ( virDomainPtr domain, unsigned long memory )

virDomainSetMaxMemory: : a domain object or NULL : the memory size in kibibytes (blocks of 1024 bytes)

Dynamically change the maximum amount of physical memory allocated to a domain. If domain is NULL, then this change the amount of memory reserved to Domain0 i.e. the domain where the application runs. This function may require privileged access to the hypervisor.

This command is hypervisor-specific for whether active, persistent, or both configurations are changed; for more control, use [virDomainSetMemoryFlags\(\)](#).

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.161 int virDomainSetMemory ( virDomainPtr domain, unsigned long memory )

virDomainSetMemory: : a domain object or NULL : the memory size in kibibytes (blocks of 1024 bytes)

Dynamically change the target amount of physical memory allocated to a domain. If domain is NULL, then this change the amount of memory reserved to Domain0 i.e. the domain where the application runs. This function may require privileged access to the hypervisor.

This command only changes the runtime configuration of the domain, so can only be called on an active domain.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.162 int virDomainSetMemoryFlags ( virDomainPtr domain, unsigned long memory, unsigned int flags )

virDomainSetMemoryFlags: : a domain object or NULL : the memory size in kibibytes (blocks of 1024 bytes) : bitwise-OR of virDomainMemoryModFlags

Dynamically change the target amount of physical memory allocated to a domain. If domain is NULL, then this change the amount of memory reserved to Domain0 i.e. the domain where the application runs. This function may require privileged access to the hypervisor.

may include VIR\_DOMAIN\_AFFECT\_LIVE or VIR\_DOMAIN\_AFFECT\_CONFIG. Both flags may be set. If VIR\_DOMAIN\_AFFECT\_LIVE is set, the change affects a running domain and will fail if domain is not active. If VIR\_DOMAIN\_AFFECT\_CONFIG is set, the change affects persistent state, and will fail for transient domains. If neither flag is specified (that is, is VIR\_DOMAIN\_AFFECT\_CURRENT), then an inactive domain modifies persistent setup, while an active domain is hypervisor-dependent on whether just live or both live and persistent state is changed. If VIR\_DOMAIN\_MEM\_MAXIMUM is set, the change affects domain's maximum memory size rather than current memory size. Not all hypervisors can support all flag combinations.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.163 int virDomainSetMemoryParameters ( virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags )

virDomainSetMemoryParameters: : pointer to domain object : pointer to memory parameter objects : number of memory parameter (this value can be the same or less than the number of parameters supported) : bitwise-OR of virDomainModificationImpact

Change all or a subset of the memory tunables. This function may require privileged access to the hypervisor.

Returns -1 in case of error, 0 in case of success.

**4.1.4.164** `int virDomainSetMetadata ( virDomainPtr domain, int type, const char * metadata, const char * key, const char * uri, unsigned int flags )`

`virDomainSetMetadata`: : a domain object : type of description, from `virDomainMetadataType` : new metadata text : XML namespace key, or NULL : XML namespace URI, or NULL : bitwise-OR of `virDomainModificationImpact`

Sets the appropriate domain element given by to the value of . A of `VIR_DOMAIN_METADATA_DESCRIPTION` is free-form text; `VIR_DOMAIN_METADATA_TITLE` is free-form, but no newlines are permitted, and should be short (although the length is not enforced). For these two options and are irrelevant and must be set to NULL.

For type `VIR_DOMAIN_METADATA_ELEMENT` must be well-formed XML belonging to namespace defined by with local name .

Passing NULL for says to remove that element from the domain XML (passing the empty string leaves the element present).

The resulting metadata will be present in [virDomainGetXMLDesc\(\)](#), as well as quick access through [virDomainGetMetadata\(\)](#).

controls whether the live domain, persistent configuration, or both will be modified.

Returns 0 on success, -1 in case of failure.

**4.1.4.165** `int virDomainSetNumaParameters ( virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags )`

`virDomainSetNumaParameters`: : pointer to domain object : pointer to numa parameter objects : number of numa parameters (this value can be the same or less than the number of parameters supported) : bitwise-OR of `virDomainModificationImpact`

Change all or a subset of the numa tunables. This function may require privileged access to the hypervisor.

Returns -1 in case of error, 0 in case of success.

**4.1.4.166** `int virDomainSetSchedulerParameters ( virDomainPtr domain, virTypedParameterPtr params, int nparams )`

`virDomainSetSchedulerParameters`: : pointer to domain object : pointer to scheduler parameter objects : number of scheduler parameter objects (this value can be the same or less than the returned value `nparams` of `virDomainGetSchedulerType`)

Change all or a subset or the scheduler parameters. It is hypervisor-specific whether this sets live, persistent, or both settings; for more control, use `virDomainSetSchedulerParametersFlags`.

Returns -1 in case of error, 0 in case of success.

**4.1.4.167** `int virDomainSetSchedulerParametersFlags ( virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags )`

`virDomainSetSchedulerParametersFlags`: : pointer to domain object : pointer to scheduler parameter objects : number of scheduler parameter objects (this value can be the same or less than the returned value `nparams` of `virDomainGetSchedulerType`) : bitwise-OR of `virDomainModificationImpact`

Change a subset or all scheduler parameters. The value of should be either `VIR_DOMAIN_AFFECT_CURRENT`, or a bitwise-or of values from `VIR_DOMAIN_AFFECT_LIVE` and `VIR_DOMAIN_AFFECT_CURRENT`, although hypervisors vary in which flags are supported.

Returns -1 in case of error, 0 in case of success.

**4.1.4.168** int virDomainSetVcpus ( virDomainPtr domain, unsigned int nvcpus )

virDomainSetVcpus: : pointer to domain object, or NULL for Domain0 : the new number of virtual CPUs for this domain

Dynamically change the number of virtual CPUs used by the domain. Note that this call may fail if the underlying virtualization hypervisor does not support it or if growing the number is arbitrary limited. This function may require privileged access to the hypervisor.

This command only changes the runtime configuration of the domain, so can only be called on an active domain. It is hypervisor-dependent whether it also affects persistent configuration; for more control, use [virDomainSetVcpusFlags\(\)](#).

Returns 0 in case of success, -1 in case of failure.

**4.1.4.169** int virDomainSetVcpusFlags ( virDomainPtr domain, unsigned int nvcpus, unsigned int flags )

virDomainSetVcpusFlags: : pointer to domain object, or NULL for Domain0 : the new number of virtual CPUs for this domain, must be at least 1 : bitwise-OR of virDomainVcpuFlags

Dynamically change the number of virtual CPUs used by the domain. Note that this call may fail if the underlying virtualization hypervisor does not support it or if growing the number is arbitrary limited. This function may require privileged access to the hypervisor.

may include VIR\_DOMAIN\_AFFECT\_LIVE to affect a running domain (which may fail if domain is not active), or VIR\_DOMAIN\_AFFECT\_CONFIG to affect the next boot via the XML description of the domain. Both flags may be set. If neither flag is specified (that is, is VIR\_DOMAIN\_AFFECT\_CURRENT), then an inactive domain modifies persistent setup, while an active domain is hypervisor-dependent on whether just live or both live and persistent state is changed.

If includes VIR\_DOMAIN\_VCPU\_MAXIMUM, then VIR\_DOMAIN\_AFFECT\_LIVE must be clear, and only the maximum virtual CPU limit is altered; generally, this value must be less than or equal to [virConnectGetMaxVcpus\(\)](#). Otherwise, this call affects the current virtual CPU limit, which must be less than or equal to the maximum limit. Not all hypervisors can support all flag combinations.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.170** int virDomainShutdown ( virDomainPtr domain )

virDomainShutdown: : a domain object

Shutdown a domain, the domain object is still usable thereafter but the domain OS is being stopped. Note that the guest OS may ignore the request. For guests that react to a shutdown request, the differences from [virDomainDestroy\(\)](#) are that the guests disk storage will be in a stable state rather than having the (virtual) power cord pulled, and this command returns as soon as the shutdown request is issued rather than blocking until the guest is no longer running.

If the domain is transient and has any snapshot metadata (see [virDomainSnapshotNum\(\)](#)), then that metadata will automatically be deleted when the domain quits.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.171** int virDomainShutdownFlags ( virDomainPtr domain, unsigned int flags )

virDomainShutdownFlags: : a domain object : bitwise-OR of virDomainShutdownFlagValues

Shutdown a domain, the domain object is still usable thereafter but the domain OS is being stopped. Note that the guest OS may ignore the request. For guests that react to a shutdown request, the differences from [virDomainDestroy\(\)](#) are that the guest's disk storage will be in a stable state rather than having the (virtual) power cord pulled, and this command returns as soon as the shutdown request is issued rather than blocking until the guest is no longer running.

If the domain is transient and has any snapshot metadata (see [virDomainSnapshotNum\(\)](#)), then that metadata will automatically be deleted when the domain quits.

If is set to zero, then the hypervisor will choose the method of shutdown it considers best. To have greater control pass exactly one of the [virDomainShutdownFlagValues](#).

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.172 **virDomainSnapshotPtr virDomainSnapshotCreateXML ( virDomainPtr domain, const char \* xmlDesc, unsigned int flags )**

**virDomainSnapshotCreateXML**: : a domain object : string containing an XML description of the domain : bitwise-OR of [virDomainSnapshotCreateFlags](#)

Creates a new snapshot of a domain based on the snapshot xml contained in xmlDesc.

If is 0, the domain can be active, in which case the snapshot will be a system checkpoint (both disk state and runtime VM state such as RAM contents), where reverting to the snapshot is the same as resuming from hibernation (TCP connections may have timed out, but everything else picks up where it left off); or the domain can be inactive, in which case the snapshot includes just the disk state prior to booting. The newly created snapshot becomes current (see [virDomainSnapshotCurrent\(\)](#)), and is a child of any previous current snapshot.

If includes [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_REDEFINE](#), then this is a request to reinstate snapshot metadata that was previously discarded, rather than creating a new snapshot. This can be used to recreate a snapshot hierarchy on a destination, then remove it on the source, in order to allow migration (since migration normally fails if snapshot metadata still remains on the source machine). When redefining snapshot metadata, the current snapshot will not be altered unless the [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_CURRENT](#) flag is also present. It is an error to request the [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_CURRENT](#) flag without [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_REDEFINE](#). On some hypervisors, redefining an existing snapshot can be used to alter host-specific portions of the domain XML to be used during revert (such as backing filenames associated with disk devices), but must not alter guest-visible layout. When redefining a snapshot name that does not exist, the hypervisor may validate that reverting to the snapshot appears to be possible (for example, disk images have snapshot contents by the requested name). Not all hypervisors support these flags.

If includes [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_NO\\_METADATA](#), then the domain's disk images are modified according to , but then the just-created snapshot has its metadata deleted. This flag is incompatible with [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_REDEFINE](#).

If includes [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_HALT](#), then the domain will be inactive after the snapshot completes, regardless of whether it was active before; otherwise, a running domain will still be running after the snapshot. This flag is invalid on transient domains, and is incompatible with [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_REDEFINE](#).

If includes [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_DISK\\_ONLY](#), then the snapshot will be limited to the disks described in , and no VM state will be saved. For an active guest, the disk image may be inconsistent (as if power had been pulled), and specifying this with the [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_HALT](#) flag risks data loss.

If includes [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_QUIESCE](#), then the libvirt will attempt to use guest agent to freeze and thaw all file systems in use within domain OS. However, if the guest agent is not present, an error is thrown. Moreover, this flag requires [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_DISK\\_ONLY](#) to be passed as well.

By default, if the snapshot involves external files, and any of the destination files already exist as a non-empty regular file, the snapshot is rejected to avoid losing contents of those files. However, if includes [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_REUSE\\_EXT](#), then the destination files must already exist and contain content identical to the source files (this allows a management app to pre-create files with relative backing file names, rather than the default of creating with absolute backing file names).

Be aware that although libvirt prefers to report errors up front with no other effect, some hypervisors have certain types of failures where the overall command can easily fail even though the guest configuration was partially altered (for example, if a disk snapshot request for two disks fails on the second disk, but the first disk alteration cannot be rolled back). If this API call fails, it is therefore normally necessary to follow up with [virDomainGetXMLDesc\(\)](#) and check each disk to determine if any partial changes occurred. However, if contains [VIR\\_DOMAIN\\_SNAPSHOT\\_CREATE\\_ATOMIC](#), then libvirt guarantees that this command will not alter any disks unless the entire set of changes

can be done atomically, making failure recovery simpler (note that it is still possible to fail after disks have changed, but only in the much rarer cases of running out of memory or disk space).

Some hypervisors may prevent this operation if there is a current block copy operation; in that case, use [virDomainBlockJobAbort\(\)](#) to stop the block copy first.

Returns an (opaque) virDomainSnapshotPtr on success, NULL on failure.

#### 4.1.4.173 virDomainSnapshotPtr virDomainSnapshotCurrent ( virDomainPtr domain, unsigned int flags )

virDomainSnapshotCurrent: : a domain object : extra flags; not used yet, so callers should always pass 0

Get the current snapshot for a domain, if any.

Returns a domain snapshot object or NULL in case of failure. If the current domain snapshot cannot be found, then the VIR\_ERR\_NO\_DOMAIN\_SNAPSHOT error is raised.

#### 4.1.4.174 int virDomainSnapshotDelete ( virDomainSnapshotPtr snapshot, unsigned int flags )

virDomainSnapshotDelete: : a domain snapshot object : bitwise-OR of supported virDomainSnapshotDeleteFlags

Delete the snapshot.

If is 0, then just this snapshot is deleted, and changes from this snapshot are automatically merged into children snapshots. If includes VIR\_DOMAIN\_SNAPSHOT\_DELETE\_CHILDREN, then this snapshot and any descendant snapshots are deleted. If includes VIR\_DOMAIN\_SNAPSHOT\_DELETE\_CHILDREN\_ONLY, then any descendant snapshots are deleted, but this snapshot remains. These two flags are mutually exclusive.

If includes VIR\_DOMAIN\_SNAPSHOT\_DELETE\_METADATA\_ONLY, then any snapshot metadata tracked by libvirt is removed while keeping the snapshot contents intact; if a hypervisor does not require any libvirt metadata to track snapshots, then this flag is silently ignored.

Returns 0 if the selected snapshot(s) were successfully deleted, -1 on error.

#### 4.1.4.175 int virDomainSnapshotFree ( virDomainSnapshotPtr snapshot )

virDomainSnapshotFree: : a domain snapshot object

Free the domain snapshot object. The snapshot itself is not modified. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.176 virConnectPtr virDomainSnapshotGetConnect ( virDomainSnapshotPtr snapshot )

virDomainSnapshotGetConnect: : a snapshot object

Get the connection that owns the domain that a snapshot was created for

Returns the connection or NULL.

#### 4.1.4.177 virDomainPtr virDomainSnapshotGetDomain ( virDomainSnapshotPtr snapshot )

virDomainSnapshotGetDomain: : a snapshot object

Get the domain that a snapshot was created for

Returns the domain or NULL.

#### 4.1.4.178 `const char* virDomainSnapshotGetName ( virDomainSnapshotPtr snapshot )`

`virDomainSnapshotGetName`: : a snapshot object

Get the public name for that snapshot

Returns a pointer to the name or NULL, the string need not be deallocated as its lifetime will be the same as the snapshot object.

#### 4.1.4.179 `virDomainSnapshotPtr virDomainSnapshotGetParent ( virDomainSnapshotPtr snapshot, unsigned int flags )`

`virDomainSnapshotGetParent`: : a snapshot object : extra flags; not used yet, so callers should always pass 0

Get the parent snapshot for , if any.

Returns a domain snapshot object or NULL in case of failure. If the given snapshot is a root (no parent), then the VIR\_ERR\_NO\_DOMAIN\_SNAPSHOT error is raised.

#### 4.1.4.180 `char* virDomainSnapshotGetXMLDesc ( virDomainSnapshotPtr snapshot, unsigned int flags )`

`virDomainSnapshotGetXMLDesc`: : a domain snapshot object : bitwise-OR of subset of `virDomainXMLFlags`

Provide an XML description of the domain snapshot.

No security-sensitive data will be included unless contains VIR\_DOMAIN\_XML\_SECURE; this flag is rejected on read-only connections. For this API, should not contain either VIR\_DOMAIN\_XML\_INACTIVE or VIR\_DOMAIN\_XML\_UPDATE\_CPU.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

#### 4.1.4.181 `int virDomainSnapshotHasMetadata ( virDomainSnapshotPtr snapshot, unsigned int flags )`

`virDomainSnapshotHasMetadata`: : a snapshot object : extra flags; not used yet, so callers should always pass 0

Determine if the given snapshot is associated with libvirt metadata that would prevent the deletion of the domain.

Returns 1 if the snapshot has metadata, 0 if the snapshot exists without help from libvirt, or -1 on error.

#### 4.1.4.182 `int virDomainSnapshotIsCurrent ( virDomainSnapshotPtr snapshot, unsigned int flags )`

`virDomainSnapshotIsCurrent`: : a snapshot object : extra flags; not used yet, so callers should always pass 0

Determine if the given snapshot is the domain's current snapshot. See also [virDomainHasCurrentSnapshot\(\)](#).

Returns 1 if current, 0 if not current, or -1 on error.

#### 4.1.4.183 `int virDomainSnapshotListAllChildren ( virDomainSnapshotPtr snapshot, virDomainSnapshotPtr ** snaps, unsigned int flags )`

`virDomainSnapshotListAllChildren`: : a domain snapshot object : pointer to variable to store the array containing snapshot objects, or NULL if the list is not required (just returns number of snapshots) : bitwise-OR of supported `virDomainSnapshotListFlags`

Collect the list of domain snapshots that are children of the given snapshot, and allocate an array to store those objects. This API solves the race inherent in [virDomainSnapshotListChildrenNames\(\)](#).

By default, this command covers only direct children; it is also possible to expand things to cover all descendants, when includes VIR\_DOMAIN\_SNAPSHOT\_LIST\_DESCENDANTS. Also, some filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within



a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virDomainSnapshotFree\(\)](#) on each array element, then calling `free()` on .

**4.1.4.184** `int virDomainSnapshotListChildrenNames ( virDomainSnapshotPtr snapshot, char ** names, int nameslen, unsigned int flags )`

`virDomainSnapshotListChildrenNames`: : a domain snapshot object : array to collect the list of names of snapshots : size of : bitwise-OR of supported `virDomainSnapshotListFlags`

Collect the list of domain snapshots that are children of the given snapshot, and store their names in . The value to use for can be determined by [virDomainSnapshotNumChildren\(\)](#) with the same .

By default, this command covers only direct children; it is also possible to expand things to cover all descendants, when includes `VIR_DOMAIN_SNAPSHOT_LIST_DESCENDANTS`. Also, some filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 in case of error. Note that this command is inherently racy: another connection can define a new snapshot between a call to [virDomainSnapshotNumChildren\(\)](#) and this call. You are only guaranteed that all currently defined snapshots were listed if the return is less than . Likewise, you should be prepared for [virDomainSnapshotLookupByName\(\)](#) to fail when converting a name from this call into a snapshot object, if another connection deletes the snapshot in the meantime. For more control over the results, see [virDomainSnapshotListAllChildren\(\)](#).

Returns the number of domain snapshots found or -1 in case of error. The caller is responsible for freeing each member of the array.

**4.1.4.185** `int virDomainSnapshotListNames ( virDomainPtr domain, char ** names, int nameslen, unsigned int flags )`

`virDomainSnapshotListNames`: : a domain object : array to collect the list of names of snapshots : size of : bitwise-OR of supported `virDomainSnapshotListFlags`

Collect the list of domain snapshots for the given domain, and store their names in . The value to use for can be determined by [virDomainSnapshotNum\(\)](#) with the same .

By default, this command covers all snapshots; it is also possible to limit things to just snapshots with no parents, when includes `VIR_DOMAIN_SNAPSHOT_LIST_ROOTS`. Additional filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor

cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Note that this command is inherently racy: another connection can define a new snapshot between a call to `virDomainSnapshotNum()` and this call. You are only guaranteed that all currently defined snapshots were listed if the return is less than . Likewise, you should be prepared for `virDomainSnapshotLookupByName()` to fail when converting a name from this call into a snapshot object, if another connection deletes the snapshot in the meantime. For more control over the results, see `virDomainListAllSnapshots()`.

Returns the number of domain snapshots found or -1 in case of error. The caller is responsible for freeing each member of the array.

**4.1.4.186** `virDomainSnapshotPtr virDomainSnapshotLookupByName ( virDomainPtr domain, const char * name, unsigned int flags )`

**4.1.4.187** `int virDomainSnapshotNum ( virDomainPtr domain, unsigned int flags )`

`virDomainSnapshotNum`: : a domain object : bitwise-OR of supported `virDomainSnapshotListFlags`

Provides the number of domain snapshots for this domain.

By default, this command covers all snapshots; it is also possible to limit things to just snapshots with no parents, when includes `VIR_DOMAIN_SNAPSHOT_LIST_ROOTS`. Additional filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 in case of error.

**4.1.4.188** `int virDomainSnapshotNumChildren ( virDomainSnapshotPtr snapshot, unsigned int flags )`

`virDomainSnapshotNumChildren`: : a domain snapshot object : bitwise-OR of supported `virDomainSnapshotListFlags`

Provides the number of child snapshots for this domain snapshot.

By default, this command covers only direct children; it is also possible to expand things to cover all descendants, when includes `VIR_DOMAIN_SNAPSHOT_LIST_DESCENDANTS`. Also, some filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).



The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 in case of error.

#### 4.1.4.189 `int virDomainSnapshotRef ( virDomainSnapshotPtr snapshot )`

`virDomainSnapshotRef`: : the snapshot to hold a reference on

Increment the reference count on the snapshot. For each additional call to this method, there shall be a corresponding call to `virDomainSnapshotFree` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection and domain remain open until all threads have finished using the snapshot. ie, each new thread using a snapshot would increment the reference count.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.190 `int virDomainSuspend ( virDomainPtr domain )`

`virDomainSuspend`: : a domain object

Suspends an active domain, the process is frozen without further access to CPU resources and I/O but the memory used by the domain at the hypervisor level will stay allocated. Use `virDomainResume()` to reactivate the domain. This function may require privileged access. Moreover, suspend may not be supported if domain is in some special state like `VIR_DOMAIN_PMSUSPENDED`.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.191 `int virDomainUndefine ( virDomainPtr domain )`

`virDomainUndefine`: : pointer to a defined domain

Undefine a domain. If the domain is running, it's converted to transient domain, without stopping it. If the domain is inactive, the domain configuration is removed.

If the domain has a managed save image (see `virDomainHasManagedSaveImage()`), or if it is inactive and has any snapshot metadata (see `virDomainSnapshotNum()`), then the undefine will fail. See `virDomainUndefineFlags()` for more control.

Returns 0 in case of success, -1 in case of error

#### 4.1.4.192 `int virDomainUndefineFlags ( virDomainPtr domain, unsigned int flags )`

`virDomainUndefineFlags`: : pointer to a defined domain : bitwise-OR of supported `virDomainUndefineFlagsValues`

Undefine a domain. If the domain is running, it's converted to transient domain, without stopping it. If the domain is inactive, the domain configuration is removed.

If the domain has a managed save image (see `virDomainHasManagedSaveImage()`), then including `VIR_DOMAIN_UNDEFINE_MANAGED_SAVE` in will also remove that file, and omitting the flag will cause the undefine process to fail.

If the domain is inactive and has any snapshot metadata (see `virDomainSnapshotNum()`), then including `VIR_DOMAIN_UNDEFINE_SNAPSHOTS_METADATA` in will also remove that metadata. Omitting the flag will cause the undefine of an inactive domain to fail. Active snapshots will retain snapshot metadata until the (now-transient) domain halts, regardless of whether this flag is present. On hypervisors where snapshots do not use libvirt metadata, this flag has no effect.

Returns 0 in case of success, -1 in case of error

4.1.4.193 `int virDomainUpdateDeviceFlags ( virDomainPtr domain, const char * xml, unsigned int flags )`

`virDomainUpdateDeviceFlags`: : pointer to domain object : pointer to XML description of one device : bitwise-OR of `virDomainDeviceModifyFlags`

Change a virtual device on a domain, using the `flags` parameter to control how the device is changed. `VIR_DOMAIN_AFFECT_CURRENT` specifies that the device change is made based on current domain state. `VIR_DOMAIN_AFFECT_LIVE` specifies that the device shall be changed on the active domain instance only and is not added to the persisted domain configuration. `VIR_DOMAIN_AFFECT_CONFIG` specifies that the device shall be changed on the persisted domain configuration only. Note that the target hypervisor must return an error if unable to satisfy flags. E.g. the hypervisor driver will return failure if `LIVE` is specified but it only supports modifying the persisted device allocation.

This method is used for actions such changing CDROM/Floppy device media, altering the graphics configuration such as password, reconfiguring the NIC device backend connectivity, etc.

Returns 0 in case of success, -1 in case of failure.

4.1.4.194 `int virEventAddHandle ( int fd, int events, virEventHandlerCallback cb, void * opaque, virFreeCallback ff )`

4.1.4.195 `int virEventAddTimeout ( int frequency, virEventTimeoutCallback cb, void * opaque, virFreeCallback ff )`

4.1.4.196 `int virEventRegisterDefaultImpl ( void )`

4.1.4.197 `void virEventRegisterImpl ( virEventAddHandleFunc addHandle, virEventUpdateHandleFunc updateHandle, virEventRemoveHandleFunc removeHandle, virEventAddTimeoutFunc addTimeout, virEventUpdateTimeoutFunc updateTimeout, virEventRemoveTimeoutFunc removeTimeout )`

4.1.4.198 `int virEventRemoveHandle ( int watch )`

4.1.4.199 `int virEventRemoveTimeout ( int timer )`

4.1.4.200 `int virEventRunDefaultImpl ( void )`

4.1.4.201 `void virEventUpdateHandle ( int watch, int events )`

4.1.4.202 `void virEventUpdateTimeout ( int timer, int frequency )`

4.1.4.203 `int virGetVersion ( unsigned long * libVer, const char * type, unsigned long * typeVer )`

4.1.4.204 `int virInitialize ( void )`

`virInitialize`:

Initialize the library. It's better to call this routine at startup in multithreaded applications to avoid potential race when initializing the library.

Calling `virInitialize` is mandatory, unless your first API call is one of `virConnectOpen*`.

Returns 0 in case of success, -1 in case of error

4.1.4.205 `int virInterfaceChangeBegin ( virConnectPtr conn, unsigned int flags )`

`virInterfaceChangeBegin`: : pointer to hypervisor connection : extra flags; not used yet, so callers should always pass 0

This function creates a restore point to which one can return later by calling [virInterfaceChangeRollback\(\)](#). This function should be called before any transaction with interface configuration. Once it is known that a new configuration works, it can be committed via [virInterfaceChangeCommit\(\)](#), which frees the restore point.

If [virInterfaceChangeBegin\(\)](#) is called when a transaction is already opened, this function will fail, and a `VIR_ERR_INVALID_OPERATION` will be logged.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.206 `int virInterfaceChangeCommit ( virConnectPtr conn, unsigned int flags )`

`virInterfaceChangeCommit`: : pointer to hypervisor connection : extra flags; not used yet, so callers should always pass 0

This commits the changes made to interfaces and frees the restore point created by [virInterfaceChangeBegin\(\)](#).

If [virInterfaceChangeCommit\(\)](#) is called when a transaction is not opened, this function will fail, and a `VIR_ERR_INVALID_OPERATION` will be logged.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.207 `int virInterfaceChangeRollback ( virConnectPtr conn, unsigned int flags )`

`virInterfaceChangeRollback`: : pointer to hypervisor connection : extra flags; not used yet, so callers should always pass 0

This cancels changes made to interfaces settings by restoring previous state created by [virInterfaceChangeBegin\(\)](#).

If [virInterfaceChangeRollback\(\)](#) is called when a transaction is not opened, this function will fail, and a `VIR_ERR_INVALID_OPERATION` will be logged.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.208 `int virInterfaceCreate ( virInterfacePtr iface, unsigned int flags )`

`virInterfaceCreate`: : pointer to a defined interface : extra flags; not used yet, so callers should always pass 0

Activate an interface (i.e. call "ifup").

If there was an open network config transaction at the time this interface was defined (that is, if [virInterfaceChangeBegin\(\)](#) had been called), the interface will be brought back down (and then undefined) if [virInterfaceChangeRollback\(\)](#) is called. p \* Returns 0 in case of success, -1 in case of error

#### 4.1.4.209 `virInterfacePtr virInterfaceDefineXML ( virConnectPtr conn, const char * xml, unsigned int flags )`

`virInterfaceDefineXML`: : pointer to the hypervisor connection : the XML description for the interface, preferably in UTF-8 : extra flags; not used yet, so callers should always pass 0

Define an interface (or modify existing interface configuration).

Normally this change in the interface configuration is immediately permanent/persistent, but if [virInterfaceChangeBegin\(\)](#) has been previously called (i.e. if an interface config transaction is open), the new interface definition will only become permanent if [virInterfaceChangeCommit\(\)](#) is called prior to the next reboot of the system running libvirtd. Prior to that time, it can be explicitly removed using [virInterfaceChangeRollback\(\)](#), or will be automatically removed during the next reboot of the system running libvirtd.

Returns NULL in case of error, a pointer to the interface otherwise

#### 4.1.4.210 `int virInterfaceDestroy ( virInterfacePtr iface, unsigned int flags )`

`virInterfaceDestroy`: : an interface object : extra flags; not used yet, so callers should always pass 0

deactivate an interface (ie call "ifdown") This does not remove the interface from the config, and does not free the associated `virInterfacePtr` object.

If there is an open network config transaction at the time this interface is destroyed (that is, if [virInterfaceChangeBegin\(\)](#) had been called), and if the interface is later undefined and then [virInterfaceChangeRollback\(\)](#) is called, the restoral of the interface definition will also bring the interface back up.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.211 `int virInterfaceFree ( virInterfacePtr iface )`

`virInterfaceFree`: : an interface object

Free the interface object. The interface itself is unaltered. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.212 `virConnectPtr virInterfaceGetConnect ( virInterfacePtr iface )`

`virInterfaceGetConnect`: : pointer to an interface

Provides the connection pointer associated with an interface. The reference counter on the connection is not increased by this call.

WARNING: When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the interface object together.

Returns the `virConnectPtr` or NULL in case of failure.

#### 4.1.4.213 `const char* virInterfaceGetMACString ( virInterfacePtr iface )`

`virInterfaceGetMACString`: : an interface object

Get the MAC for an interface as string. For more information about MAC see RFC4122.

Returns a pointer to the MAC address (in null-terminated ASCII format) or NULL, the string need not be deallocated its lifetime will be the same as the interface object.

#### 4.1.4.214 `const char* virInterfaceGetName ( virInterfacePtr iface )`

`virInterfaceGetName`: : an interface object

Get the public name for that interface

Returns a pointer to the name or NULL, the string need not be deallocated its lifetime will be the same as the interface object.

#### 4.1.4.215 `char* virInterfaceGetXMLDesc ( virInterfacePtr iface, unsigned int flags )`

`virInterfaceGetXMLDesc`: : an interface object : bitwise-OR of extraction flags. Current valid bits:

```
VIR_INTERFACE_XML_INACTIVE - return the static configuration,
                             suitable for use redefining the
                             interface via virInterfaceDefineXML()
```

Provide an XML description of the interface. If `VIR_INTERFACE_XML_INACTIVE` is set, the description may be reused later to redefine the interface with [virInterfaceDefineXML\(\)](#). If it is not set, the ip address and netmask will be the current live setting of the interface, not the settings from the config files.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must `free()` the returned value.

**4.1.4.216 int virInterfaceIsActive ( virInterfacePtr iface )**

virInterfaceIsActive: : pointer to the interface object

Determine if the interface is currently running

Returns 1 if running, 0 if inactive, -1 on error

**4.1.4.217 virInterfacePtr virInterfaceLookupByMACString ( virConnectPtr conn, const char \* macstr )**

virInterfaceLookupByMACString: : pointer to the hypervisor connection : the MAC for the interface (null-terminated ASCII format)

Try to lookup an interface on the given hypervisor based on its MAC.

Returns a new interface object or NULL in case of failure. If the interface cannot be found, then VIR\_ERR\_NO\_INTERFACE error is raised.

**4.1.4.218 virInterfacePtr virInterfaceLookupByName ( virConnectPtr conn, const char \* name )****4.1.4.219 int virInterfaceRef ( virInterfacePtr iface )**

virInterfaceRef: : the interface to hold a reference on

Increment the reference count on the interface. For each additional call to this method, there shall be a corresponding call to virInterfaceFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using an interface would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.220 int virInterfaceUndefine ( virInterfacePtr iface )**

virInterfaceUndefine: : pointer to a defined interface

Undefine an interface, ie remove it from the config. This does not free the associated virInterfacePtr object.

Normally this change in the interface configuration is permanent/persistent, but if [virInterfaceChangeBegin\(\)](#) has been previously called (i.e. if an interface config transaction is open), the removal of the interface definition will only become permanent if [virInterfaceChangeCommit\(\)](#) is called prior to the next reboot of the system running libvirt. Prior to that time, the definition can be explicitly restored using [virInterfaceChangeRollback\(\)](#), or will be automatically restored during the next reboot of the system running libvirtd.

Returns 0 in case of success, -1 in case of error

**4.1.4.221 int virNetworkCreate ( virNetworkPtr network )**

virNetworkCreate: : pointer to a defined network

Create and start a defined network. If the call succeed the network moves from the defined to the running networks pools.

Returns 0 in case of success, -1 in case of error

**4.1.4.222 virNetworkPtr virNetworkCreateXML ( virConnectPtr conn, const char \* xmlDesc )**

virNetworkCreateXML: : pointer to the hypervisor connection : an XML description of the network

Create and start a new virtual network, based on an XML description similar to the one returned by [virNetworkGetXMLDesc\(\)](#)

Returns a new network object or NULL in case of failure

#### 4.1.4.223 **virNetworkPtr** virNetworkDefineXML ( **virConnectPtr** *conn*, **const char \*** *xml* )

virNetworkDefineXML: : pointer to the hypervisor connection : the XML description for the network, preferably in UTF-8

Define a network, but does not create it

Returns NULL in case of error, a pointer to the network otherwise

#### 4.1.4.224 **int** virNetworkDestroy ( **virNetworkPtr** *network* )

virNetworkDestroy: : a network object

Destroy the network object. The running instance is shutdown if not down already and all resources used by it are given back to the hypervisor. This does not free the associated virNetworkPtr object. This function may require privileged access

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.225 **int** virNetworkFree ( **virNetworkPtr** *network* )

virNetworkFree: : a network object

Free the network object. The running instance is kept alive. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

#### 4.1.4.226 **int** virNetworkGetAutostart ( **virNetworkPtr** *network*, **int \*** *autostart* )

virNetworkGetAutostart: : a network object : the value returned

Provides a boolean value indicating whether the network configured to be automatically started when the host machine boots.

Returns -1 in case of error, 0 in case of success

#### 4.1.4.227 **char\*** virNetworkGetBridgeName ( **virNetworkPtr** *network* )

virNetworkGetBridgeName: : a network object

Provides a bridge interface name to which a domain may connect a network interface in order to join the network.

Returns a 0 terminated interface name, or NULL in case of error. the caller must free() the returned value.

#### 4.1.4.228 **POL New** **char\*** virNetworkGetBridgeType ( **virNetworkPtr** *network* )

virNetworkGetBridgeType: : a network object

Provides a bridge interface type to which a domain may connect a network interface in order to join the network.

Returns a 0 terminated interface name, or NULL in case of error. the caller must free() the returned value.

**4.1.4.229 virConnectPtr virNetworkGetConnect ( virNetworkPtr net )**

virNetworkGetConnect: : pointer to a network

Provides the connection pointer associated with a network. The reference counter on the connection is not increased by this call.

WARNING: When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the network object together.

Returns the virConnectPtr or NULL in case of failure.

**4.1.4.230 const char\* virNetworkGetName ( virNetworkPtr network )**

virNetworkGetName: : a network object

Get the public name for that network

Returns a pointer to the name or NULL, the string need not be deallocated its lifetime will be the same as the network object.

**4.1.4.231 int virNetworkGetUUID ( virNetworkPtr network, unsigned char \* uuid )**

virNetworkGetUUID: : a network object : pointer to a VIR\_UUID\_BUFLen bytes array

Get the UUID for a network

Returns -1 in case of error, 0 in case of success

**4.1.4.232 int virNetworkGetUUIDString ( virNetworkPtr network, char \* buf )**

virNetworkGetUUIDString: : a network object : pointer to a VIR\_UUID\_STRING\_BUFLen bytes array

Get the UUID for a network as string. For more information about UUID see RFC4122.

Returns -1 in case of error, 0 in case of success

**4.1.4.233 char\* virNetworkGetXMLDesc ( virNetworkPtr network, unsigned int flags )**

virNetworkGetXMLDesc: : a network object : bitwise-OR of virNetworkXMLFlags

Provide an XML description of the network. The description may be reused later to relaunch the network with [virNetworkCreateXML\(\)](#).

Normally, if a network included a physical function, the output includes all virtual functions tied to that physical interface. If includes VIR\_NETWORK\_XML\_INACTIVE, then the expansion of virtual interfaces is not performed.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

**4.1.4.234 int virNetworkIsActive ( virNetworkPtr net )**

virNetworkIsActive: : pointer to the network object

Determine if the network is currently running

Returns 1 if running, 0 if inactive, -1 on error

**4.1.4.235 int virNetworkIsPersistent ( virNetworkPtr net )**

virNetworkIsPersistent: : pointer to the network object

Determine if the network has a persistent configuration which means it will still exist after shutting down

Returns 1 if persistent, 0 if transient, -1 on error

**4.1.4.236** `virNetworkPtr virNetworkLookupByName ( virConnectPtr conn, const char * name )`

**4.1.4.237** `virNetworkPtr virNetworkLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

`virNetworkLookupByUUID`: : pointer to the hypervisor connection : the raw UUID for the network

Try to lookup a network on the given hypervisor based on its UUID.

Returns a new network object or NULL in case of failure. If the network cannot be found, then `VIR_ERR_NO_NETWORK` error is raised.

**4.1.4.238** `virNetworkPtr virNetworkLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

`virNetworkLookupByUUIDString`: : pointer to the hypervisor connection : the string UUID for the network

Try to lookup a network on the given hypervisor based on its UUID.

Returns a new network object or NULL in case of failure. If the network cannot be found, then `VIR_ERR_NO_NETWORK` error is raised.

**4.1.4.239** `int virNetworkRef ( virNetworkPtr network )`

`virNetworkRef`: : the network to hold a reference on

Increment the reference count on the network. For each additional call to this method, there shall be a corresponding call to `virNetworkFree` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a network would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.240** `int virNetworkSetAutostart ( virNetworkPtr network, int autostart )`

`virNetworkSetAutostart`: : a network object : whether the network should be automatically started 0 or 1

Configure the network to be automatically started when the host machine boots.

Returns -1 in case of error, 0 in case of success

**4.1.4.241** `int virNetworkUndefine ( virNetworkPtr network )`

`virNetworkUndefine`: : pointer to a defined network

Undefine a network but does not stop it if it is running

Returns 0 in case of success, -1 in case of error

**4.1.4.242** `int virNetworkUpdate ( virNetworkPtr network, unsigned int command, unsigned int section, int parentIndex, const char * xml, unsigned int flags )`

`virNetworkUpdate`: : pointer to a defined network



**4.1.4.243 virNodeDevicePtr virNodeDeviceCreateXML ( virConnectPtr conn, const char \* xmlDesc, unsigned int flags )**

virNodeDeviceCreateXML: : pointer to the hypervisor connection : string containing an XML description of the device to be created : extra flags; not used yet, so callers should always pass 0

Create a new device on the VM host machine, for example, virtual HBAs created using vport\_create.

Returns a node device object if successful, NULL in case of failure

**4.1.4.244 int virNodeDeviceDestroy ( virNodeDevicePtr dev )**

virNodeDeviceDestroy: : a device object

Destroy the device object. The virtual device is removed from the host operating system. This function may require privileged access

Returns 0 in case of success and -1 in case of failure.

**4.1.4.245 int virNodeDeviceDetach ( virNodeDevicePtr dev )**

virNodeDeviceDetach: : pointer to the node device

Detach the node device from the node itself so that it may be assigned to a guest domain.

Depending on the hypervisor, this may involve operations such as unbinding any device drivers from the device, binding the device to a dummy device driver and resetting the device.

If the device is currently in use by the node, this method may fail.

Once the device is not assigned to any guest, it may be re-attached to the node using the virNodeDeviceReattach() method.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.246 int virNodeDeviceFree ( virNodeDevicePtr dev )**

virNodeDeviceFree: : pointer to the node device

Drops a reference to the node device, freeing it if this was the last reference.

Returns the 0 for success, -1 for error.

**4.1.4.247 const char\* virNodeDeviceGetName ( virNodeDevicePtr dev )**

virNodeDeviceGetName: : the device

Just return the device name

Returns the device name or NULL in case of error

**4.1.4.248 const char\* virNodeDeviceGetParent ( virNodeDevicePtr dev )**

virNodeDeviceGetParent: : the device

Accessor for the parent of the device

Returns the name of the device's parent, or NULL if the device has no parent.

**4.1.4.249 char\* virNodeDeviceGetXMLDesc ( virNodeDevicePtr dev, unsigned int flags )**

virNodeDeviceGetXMLDesc: : pointer to the node device : extra flags; not used yet, so callers should always pass 0

Fetch an XML document describing all aspects of the device.

Returns the XML document, or NULL on error

**4.1.4.250** `int virNodeDeviceListCaps ( virNodeDevicePtr dev, char **const names, int maxnames )`

virNodeDeviceListCaps: : the device : array to collect the list of capability names : size of

Lists the names of the capabilities supported by the device.

Returns the number of capability names listed in .

**4.1.4.251** `virNodeDevicePtr virNodeDeviceLookupByName ( virConnectPtr conn, const char * name )`

**4.1.4.252** `int virNodeDeviceNumOfCaps ( virNodeDevicePtr dev )`

virNodeDeviceNumOfCaps: : the device

Accessor for the number of capabilities supported by the device.

Returns the number of capabilities supported by the device.

**4.1.4.253** `int virNodeDeviceReAttach ( virNodeDevicePtr dev )`

virNodeDeviceReAttach: : pointer to the node device

Re-attach a previously detached node device to the node so that it may be used by the node again.

Depending on the hypervisor, this may involve operations such as resetting the device, unbinding it from a dummy device driver and binding it to its appropriate driver.

If the device is currently in use by a guest, this method may fail.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.254** `int virNodeDeviceRef ( virNodeDevicePtr dev )`

virNodeDeviceRef: : the dev to hold a reference on

Increment the reference count on the dev. For each additional call to this method, there shall be a corresponding call to virNodeDeviceFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a dev would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.255** `int virNodeDeviceReset ( virNodeDevicePtr dev )`

virNodeDeviceReset: : pointer to the node device

Reset a previously detached node device to the node before or after assigning it to a guest.

The exact reset semantics depends on the hypervisor and device type but, for example, KVM will attempt to reset PCI devices with a Function Level Reset, Secondary Bus Reset or a Power Management D-State reset.

If the reset will affect other devices which are currently in use, this function may fail.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.256** `int virNodeGetCellsFreeMemory ( virConnectPtr conn, unsigned long long * freeMems, int startCell, int maxCells )`

`virNodeGetCellsFreeMemory`: : pointer to the hypervisor connection : pointer to the array of unsigned long long : index of first cell to return freeMems info on. : Maximum number of cells for which freeMems information can be returned.

This call returns the amount of free memory in one or more NUMA cells. The array must be allocated by the caller and will be filled with the amount of free memory in bytes for each cell requested, starting with `startCell` (in `freeMems[0]`), up to either `(startCell + maxCells)`, or the number of additional cells in the node, whichever is smaller.

Returns the number of entries filled in `freeMems`, or -1 in case of error.

**4.1.4.257** `int virNodeGetCPUStats ( virConnectPtr conn, int cpuNum, virNodeCPUStatsPtr params, int * nparams, unsigned int flags )`

`virNodeGetCPUStats`: : pointer to the hypervisor connection. : number of node cpu. (`VIR_NODE_CPU_STATS_ALL_CPUS` means total cpu statistics) : pointer to node cpu time parameter objects : number of node cpu time parameter (this value should be same or less than the number of parameters supported) : extra flags; not used yet, so callers should always pass 0

This function provides individual cpu statistics of the node. If you want to get total cpu statistics of the node, you must specify `VIR_NODE_CPU_STATS_ALL_CPUS` to . The array will be filled with the values equal to the number of parameters suggested by

As the value of `cpuNum` is dynamic, call the API setting to 0 and as NULL, the API returns the number of parameters supported by the HV by updating on SUCCESS. The caller should then allocate array, i.e. `(sizeof() * )` bytes and call the API again.

Here is a sample code snippet:

```
if ((virNodeGetCPUStats(conn, cpuNum, NULL, &nparams, 0) == 0) && (nparams != 0)) { if ((params = malloc(sizeof(virNodeCPUStats) * nparams)) == NULL) goto error; memset(params, 0, sizeof(virNodeCPUStats) * nparams); if (virNodeGetCPUStats(conn, cpuNum, params, &nparams, 0)) goto error; }
```

This function doesn't require privileged access to the hypervisor. This function expects the caller to allocate the .

CPU time Statistics:

`VIR_NODE_CPU_STATS_KERNEL`: The cumulative CPU time which spends by kernel, when the node booting up.(nanoseconds) `VIR_NODE_CPU_STATS_USER`: The cumulative CPU time which spends by user processes, when the node booting up.(nanoseconds) `VIR_NODE_CPU_STATS_IDLE`: The cumulative idle CPU time, when the node booting up.(nanoseconds) `VIR_NODE_CPU_STATS_IOWAIT`: The cumulative I/O wait CPU time, when the node booting up.(nanoseconds) `VIR_NODE_CPU_STATS_UTILIZATION`: The CPU utilization. The usage value is in percent and 100% represents all CPUs on the server.

Returns -1 in case of error, 0 in case of success.

**4.1.4.258** `unsigned long long virNodeGetFreeMemory ( virConnectPtr conn )`

`virNodeGetFreeMemory`: : pointer to the hypervisor connection

provides the free memory available on the Node Note: most libvirt APIs provide memory sizes in kibibytes, but in this function the returned value is in bytes. Divide by 1024 as necessary.

Returns the available free memory in bytes or 0 in case of error

**4.1.4.259** `int virNodeGetInfo ( virConnectPtr conn, virNodeInfoPtr info )`

`virNodeGetInfo`: : pointer to the hypervisor connection : pointer to a `virNodeInfo` structure allocated by the user

Extract hardware information about the node.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.260** `int virNodeGetMemoryParameters ( virConnectPtr conn, virTypedParameterPtr params, int * nparams, unsigned int flags )`

**4.1.4.261** `int virNodeGetMemoryStats ( virConnectPtr conn, int cellNum, virNodeMemoryStatsPtr params, int * nparams, unsigned int flags )`

`virNodeGetMemoryStats`: : pointer to the hypervisor connection. : number of node cell. (VIR\_NODE\_MEMORY\_STATS\_ALL\_CELLS means total cell statistics) : pointer to node memory stats objects : number of node memory stats (this value should be same or less than the number of stats supported) : extra flags; not used yet, so callers should always pass 0

This function provides memory stats of the node. If you want to get total cpu statistics of the node, you must specify VIR\_NODE\_MEMORY\_STATS\_ALL\_CELLS to . The array will be filled with the values equal to the number of stats suggested by

As the value of is dynamic, call the API setting to 0 and as NULL, the API returns the number of parameters supported by the HV by updating on SUCCESS. The caller should then allocate array, i.e. (sizeof() \* ) bytes and call the API again.

Here is the sample code snippet:

```
if ((virNodeGetMemoryStats(conn, cellNum, NULL, &nparams, 0) == 0) && (nparams != 0)) { if ((params = malloc(sizeof(virNodeMemoryStats) * nparams)) == NULL) goto error; memset(params, cellNum, 0, sizeof(virNodeMemoryStats) * nparams); if (virNodeGetMemoryStats(conn, params, &nparams, 0)) goto error; }
```

This function doesn't require privileged access to the hypervisor. This function expects the caller to allocate the .

Memory Stats:

VIR\_NODE\_MEMORY\_STATS\_TOTAL: The total memory usage.(KB) VIR\_NODE\_MEMORY\_STATS\_FREE: : The free memory usage.(KB) On linux, this usage includes buffers and cached. VIR\_NODE\_MEMORY\_STATS\_BUFFERS: The buffers memory usage.(KB) VIR\_NODE\_MEMORY\_STATS\_CACHED: The cached memory usage.(KB)

Returns -1 in case of error, 0 in case of success.

**4.1.4.262** `int virNodeGetSecurityModel ( virConnectPtr conn, virSecurityModelPtr secmodel )`

`virNodeGetSecurityModel`: : a connection object : pointer to a virSecurityModel structure

Extract the security model of a hypervisor. The 'model' field in the argument may be initialized to the empty string if the driver has not activated a security model.

Returns 0 in case of success, -1 in case of failure

**4.1.4.263** `int virNodeListDevices ( virConnectPtr conn, const char * cap, char **const names, int maxnames, unsigned int flags )`

`virNodeListDevices`: : pointer to the hypervisor connection : capability name : array to collect the list of node device names : size of : extra flags; not used yet, so callers should always pass 0

Collect the list of node devices, and store their names in

For more control over the results, see [virConnectListAllNodeDevices\(\)](#).

If the optional 'cap' argument is non-NULL, then the count will be restricted to devices with the specified capability

Returns the number of node devices found or -1 in case of error

**4.1.4.264** `int virNodeNumOfDevices ( virConnectPtr conn, const char * cap, unsigned int flags )`

`virNodeNumOfDevices`: : pointer to the hypervisor connection : capability name : extra flags; not used yet, so callers should always pass 0

Provides the number of node devices.

If the optional 'cap' argument is non-NULL, then the count will be restricted to devices with the specified capability

Returns the number of node devices or -1 in case of error

**4.1.4.265** `int virNodeSetMemoryParameters ( virConnectPtr conn, virTypedParameterPtr params, int nparams, unsigned int flags )`

**4.1.4.266** `int virNodeSuspendForDuration ( virConnectPtr conn, unsigned int target, unsigned long long duration, unsigned int flags )`

`virNodeSuspendForDuration`: : pointer to the hypervisor connection : the state to which the host must be suspended to, such as: `VIR_NODE_SUSPEND_TARGET_MEM` (Suspend-to-RAM) `VIR_NODE_SUSPEND_TARGET_DISK` (Suspend-to-Disk) `VIR_NODE_SUSPEND_TARGET_HYBRID` (Hybrid-Suspend, which is a combination of the former modes). : the time duration in seconds for which the host has to be suspended : extra flags; not used yet, so callers should always pass 0

Attempt to suspend the node (host machine) for the given duration of time in the specified state (Suspend-to-RAM, Suspend-to-Disk or Hybrid-Suspend). Schedule the node's Real-Time-Clock interrupt to resume the node after the duration is complete.

Returns 0 on success (i.e., the node will be suspended after a short delay), -1 on failure (the operation is not supported, or an attempted suspend is already underway).

**4.1.4.267** `virNWFilterPtr virNWFilterDefineXML ( virConnectPtr conn, const char * xmlDesc )`

`virNWFilterDefineXML`: : pointer to the hypervisor connection : an XML description of the nwfilter

Define a new network filter, based on an XML description similar to the one returned by [virNWFilterGetXMLDesc\(\)](#)

Returns a new nwfilter object or NULL in case of failure

**4.1.4.268** `int virNWFilterFree ( virNWFilterPtr nwfilter )`

`virNWFilterFree`: : a nwfilter object

Free the nwfilter object. The running instance is kept alive. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.269** `const char* virNWFilterGetName ( virNWFilterPtr nwfilter )`

`virNWFilterGetName`: : a nwfilter object

Get the public name for the network filter

Returns a pointer to the name or NULL, the string need not be deallocated its lifetime will be the same as the nwfilter object.

**4.1.4.270** `int virNWFilterGetUUID ( virNWFilterPtr nwfilter, unsigned char * uuid )`

`virNWFilterGetUUID`: : a nwfilter object : pointer to a `VIR_UUID_BUFLen` bytes array

Get the UUID for a network filter

Returns -1 in case of error, 0 in case of success

#### 4.1.4.271 `int virNWFilterGetUUIDString ( virNWFilterPtr nwfilter, char * buf )`

virNWFilterGetUUIDString: : a nwfilter object : pointer to a VIR\_UUID\_STRING\_BUFLen bytes array

Get the UUID for a network filter as string. For more information about UUID see RFC4122.

Returns -1 in case of error, 0 in case of success

#### 4.1.4.272 `char* virNWFilterGetXMLDesc ( virNWFilterPtr nwfilter, unsigned int flags )`

virNWFilterGetXMLDesc: : a nwfilter object : extra flags; not used yet, so callers should always pass 0

Provide an XML description of the network filter. The description may be reused later to redefine the network filter with virNWFilterCreateXML().

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

#### 4.1.4.273 `virNWFilterPtr virNWFilterLookupByName ( virConnectPtr conn, const char * name )`

#### 4.1.4.274 `virNWFilterPtr virNWFilterLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

virNWFilterLookupByUUID: : pointer to the hypervisor connection : the raw UUID for the network filter

Try to lookup a network filter on the given hypervisor based on its UUID.

Returns a new nwfilter object or NULL in case of failure. If the nwfilter cannot be found, then VIR\_ERR\_NO\_NWFILTER error is raised.

#### 4.1.4.275 `virNWFilterPtr virNWFilterLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

virNWFilterLookupByUUIDString: : pointer to the hypervisor connection : the string UUID for the nwfilter

Try to lookup an nwfilter on the given hypervisor based on its UUID.

Returns a new nwfilter object or NULL in case of failure. If the nwfilter cannot be found, then VIR\_ERR\_NO\_NWFILTER error is raised.

#### 4.1.4.276 `int virNWFilterRef ( virNWFilterPtr nwfilter )`

virNWFilterRef: : the nwfilter to hold a reference on

Increment the reference count on the nwfilter. For each additional call to this method, there shall be a corresponding call to virNWFilterFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using an nwfilter would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

#### 4.1.4.277 `int virNWFilterUndefine ( virNWFilterPtr nwfilter )`

virNWFilterUndefine: : a nwfilter object

Undefine the nwfilter object. This call will not succeed if a running VM is referencing the filter. This does not free the associated virNWFilterPtr object.

Returns 0 in case of success and -1 in case of failure.

**4.1.4.278 virSecretPtr virSecretDefineXML ( virConnectPtr conn, const char \* xml, unsigned int flags )**

virSecretDefineXML: : virConnect connection : XML describing the secret. : extra flags; not used yet, so callers should always pass 0

If XML specifies a UUID, locates the specified secret and replaces all attributes of the secret specified by UUID by attributes specified in xml (any attributes not specified in xml are discarded).

Otherwise, creates a new secret with an automatically chosen UUID, and initializes its attributes from xml.

Returns a the secret on success, NULL on failure.

**4.1.4.279 int virSecretFree ( virSecretPtr secret )**

virSecretFree: : pointer to a secret

Release the secret handle. The underlying secret continues to exist.

Returns 0 on success, or -1 on error

**4.1.4.280 virConnectPtr virSecretGetConnect ( virSecretPtr secret )**

virSecretGetConnect: : A virSecret secret

Provides the connection pointer associated with a secret. The reference counter on the connection is not increased by this call.

WARNING: When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the secret object together.

Returns the virConnectPtr or NULL in case of failure.

**4.1.4.281 const char\* virSecretGetUsageID ( virSecretPtr secret )**

virSecretGetUsageID: : a secret object

Get the unique identifier of the object with which this secret is to be used. The format of the identifier is dependant on the usage type of the secret. For a secret with a usage type of VIR\_SECRET\_USAGE\_TYPE\_VOLUME the identifier will be a fully qualified path name. The identifiers are intended to be unique within the set of all secrets sharing the same usage type. ie, there shall only ever be one secret for each volume path.

Returns a string identifying the object using the secret, or NULL upon error

**4.1.4.282 int virSecretGetUsageType ( virSecretPtr secret )**

virSecretGetUsageType: : a secret object

Get the type of object which uses this secret. The returned value is one of the constants defined in the virSecretUsageType enumeration. More values may be added to this enumeration in the future, so callers should expect to see usage types they do not explicitly know about.

Returns a positive integer identifying the type of object, or -1 upon error.

**4.1.4.283 int virSecretGetUUID ( virSecretPtr secret, unsigned char \* uuid )**

virSecretGetUUID: : A virSecret secret : buffer of VIR\_UUID\_BUFLen bytes in size

Fetches the UUID of the secret.

Returns 0 on success with the uuid buffer being filled, or -1 upon failure.

**4.1.4.284** `int virSecretGetUUIDString ( virSecretPtr secret, char * buf )`

`virSecretGetUUIDString`: : a secret object : pointer to a VIR\_UUID\_STRING\_BUFLen bytes array

Get the UUID for a secret as string. For more information about UUID see RFC4122.

Returns -1 in case of error, 0 in case of success

**4.1.4.285** `unsigned char* virSecretGetValue ( virSecretPtr secret, size_t * value_size, unsigned int flags )`

`virSecretGetValue`: : A virSecret connection : Place for storing size of the secret value : extra flags; not used yet, so callers should always pass 0

Fetches the value of a secret.

Returns the secret value on success, NULL on failure. The caller must free() the secret value.

**4.1.4.286** `char* virSecretGetXMLDesc ( virSecretPtr secret, unsigned int flags )`

`virSecretGetXMLDesc`: : A virSecret secret : extra flags; not used yet, so callers should always pass 0

Fetches an XML document describing attributes of the secret.

Returns the XML document on success, NULL on failure. The caller must free() the XML.

**4.1.4.287** `virSecretPtr virSecretLookupByUsage ( virConnectPtr conn, int usageType, const char * usageID )`

`virSecretLookupByUsage`: : pointer to the hypervisor connection : the type of secret usage : identifier of the object using the secret

Try to lookup a secret on the given hypervisor based on its usage The usageID is unique within the set of secrets sharing the same usageType value.

Returns a new secret object or NULL in case of failure. If the secret cannot be found, then VIR\_ERR\_NO\_SECRET error is raised.

**4.1.4.288** `virSecretPtr virSecretLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

`virSecretLookupByUUID`: : pointer to the hypervisor connection : the raw UUID for the secret

Try to lookup a secret on the given hypervisor based on its UUID. Uses the 16 bytes of raw data to describe the UUID

Returns a new secret object or NULL in case of failure. If the secret cannot be found, then VIR\_ERR\_NO\_SECRET error is raised.

**4.1.4.289** `virSecretPtr virSecretLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

`virSecretLookupByUUIDString`: : pointer to the hypervisor connection : the string UUID for the secret

Try to lookup a secret on the given hypervisor based on its UUID. Uses the printable string value to describe the UUID

Returns a new secret object or NULL in case of failure. If the secret cannot be found, then VIR\_ERR\_NO\_SECRET error is raised.

**4.1.4.290** `int virSecretRef ( virSecretPtr secret )`

`virSecretRef`: : the secret to hold a reference on



Increment the reference count on the secret. For each additional call to this method, there shall be a corresponding call to `virSecretFree` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a secret would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.291** `int virSecretSetValue ( virSecretPtr secret, const unsigned char * value, size_t value_size, unsigned int flags )`

`virSecretSetValue`: : A `virSecret` secret : Value of the secret : Size of the value : extra flags; not used yet, so callers should always pass 0

Sets the value of a secret.

Returns 0 on success, -1 on failure.

**4.1.4.292** `int virSecretUndefine ( virSecretPtr secret )`

`virSecretUndefine`: : A `virSecret` secret

Deletes the specified secret. This does not free the associated `virSecretPtr` object.

Returns 0 on success, -1 on failure.

**4.1.4.293** `int virStoragePoolBuild ( virStoragePoolPtr pool, unsigned int flags )`

`virStoragePoolBuild`: : pointer to storage pool : bitwise-OR of `virStoragePoolBuildFlags`

Currently only filesystem pool accepts flags `VIR_STORAGE_POOL_BUILD_OVERWRITE` and `VIR_STORAGE_POOL_BUILD_NO_OVERWRITE`.

Build the underlying storage pool

Returns 0 on success, or -1 upon failure

**4.1.4.294** `int virStoragePoolCreate ( virStoragePoolPtr pool, unsigned int flags )`

`virStoragePoolCreate`: : pointer to storage pool : extra flags; not used yet, so callers should always pass 0

Starts an inactive storage pool

Returns 0 on success, or -1 if it could not be started

**4.1.4.295** `virStoragePoolPtr virStoragePoolCreateXML ( virConnectPtr conn, const char * xmlDesc, unsigned int flags )`

`virStoragePoolCreateXML`: : pointer to hypervisor connection : XML description for new pool : extra flags; not used yet, so callers should always pass 0

Create a new storage based on its XML description. The pool is not persistent, so its definition will disappear when it is destroyed, or if the host is restarted

Returns a `virStoragePoolPtr` object, or NULL if creation failed

**4.1.4.296** `virStoragePoolPtr virStoragePoolDefineXML ( virConnectPtr conn, const char * xml, unsigned int flags )`

`virStoragePoolDefineXML`: : pointer to hypervisor connection : XML description for new pool : extra flags; not used yet, so callers should always pass 0

Define a new inactive storage pool based on its XML description. The pool is persistent, until explicitly undefined.

Returns a `virStoragePoolPtr` object, or NULL if creation failed

#### 4.1.4.297 `int virStoragePoolDelete ( virStoragePoolPtr pool, unsigned int flags )`

`virStoragePoolDelete`: : pointer to storage pool : bitwise-OR of `virStoragePoolDeleteFlags`

Delete the underlying pool resources. This is a non-recoverable operation. The `virStoragePoolPtr` object itself is not free'd.

Returns 0 on success, or -1 if it could not be obliterate

#### 4.1.4.298 `int virStoragePoolDestroy ( virStoragePoolPtr pool )`

`virStoragePoolDestroy`: : pointer to storage pool

Destroy an active storage pool. This will deactivate the pool on the host, but keep any persistent config associated with it. If it has a persistent config it can later be restarted with `virStoragePoolCreate()`. This does not free the associated `virStoragePoolPtr` object.

Returns 0 on success, or -1 if it could not be destroyed

#### 4.1.4.299 `int virStoragePoolFree ( virStoragePoolPtr pool )`

`virStoragePoolFree`: : pointer to storage pool

Free a storage pool object, releasing all memory associated with it. Does not change the state of the pool on the host.

Returns 0 on success, or -1 if it could not be free'd.

#### 4.1.4.300 `int virStoragePoolGetAutostart ( virStoragePoolPtr pool, int * autostart )`

`virStoragePoolGetAutostart`: : pointer to storage pool : location in which to store autostart flag

Fetches the value of the autostart flag, which determines whether the pool is automatically started at boot time

Returns 0 on success, -1 on failure

#### 4.1.4.301 `virConnectPtr virStoragePoolGetConnect ( virStoragePoolPtr pool )`

`virStoragePoolGetConnect`: : pointer to a pool

Provides the connection pointer associated with a storage pool. The reference counter on the connection is not increased by this call.

**WARNING:** When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the pool object together.

Returns the `virConnectPtr` or NULL in case of failure.

#### 4.1.4.302 `int virStoragePoolGetInfo ( virStoragePoolPtr pool, virStoragePoolInfoPtr info )`

`virStoragePoolGetInfo`: : pointer to storage pool : pointer at which to store info

Get volatile information about the storage pool such as free space / usage summary

Returns 0 on success, or -1 on failure.

**4.1.4.303** `const char* virStoragePoolGetName ( virStoragePoolPtr pool )`

virStoragePoolGetName: : pointer to storage pool

Fetch the locally unique name of the storage pool

Returns the name of the pool, or NULL on error

**4.1.4.304** `int virStoragePoolGetUUID ( virStoragePoolPtr pool, unsigned char * uuid )`

virStoragePoolGetUUID: : pointer to storage pool : buffer of VIR\_UUID\_BUFLen bytes in size

Fetch the globally unique ID of the storage pool

Returns 0 on success, or -1 on error;

**4.1.4.305** `int virStoragePoolGetUUIDString ( virStoragePoolPtr pool, char * buf )`

virStoragePoolGetUUIDString: : pointer to storage pool : buffer of VIR\_UUID\_STRING\_BUFLen bytes in size

Fetch the globally unique ID of the storage pool as a string

Returns 0 on success, or -1 on error;

**4.1.4.306** `char* virStoragePoolGetXMLDesc ( virStoragePoolPtr pool, unsigned int flags )`

virStoragePoolGetXMLDesc: : pointer to storage pool : bitwise-OR of virStorageXMLFlags

Fetch an XML document describing all aspects of the storage pool. This is suitable for later feeding back into the virStoragePoolCreateXML method.

Returns a XML document, or NULL on error

**4.1.4.307** `int virStoragePoolsActive ( virStoragePoolPtr pool )`

virStoragePoolsActive: : pointer to the storage pool object

Determine if the storage pool is currently running

Returns 1 if running, 0 if inactive, -1 on error

**4.1.4.308** `int virStoragePoolsPersistent ( virStoragePoolPtr pool )`

virStoragePoolsPersistent: : pointer to the storage pool object

Determine if the storage pool has a persistent configuration which means it will still exist after shutting down

Returns 1 if persistent, 0 if transient, -1 on error

**4.1.4.309** `int virStoragePoolListAllVolumes ( virStoragePoolPtr pool, virStorageVolPtr ** vols, unsigned int flags )`

virStoragePoolListAllVolumes: : Pointer to storage pool : Pointer to a variable to store the array containing storage volume objects or NULL if the list is not required (just returns number of volumes). : extra flags; not used yet, so callers should always pass 0

Collect the list of storage volumes, and allocate an array to store those objects.

Returns the number of storage volumes found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virStorageVolFree\(\)](#) on each array element, then calling free() on .

**4.1.4.310** `int virStoragePoolListVolumes ( virStoragePoolPtr pool, char **const names, int maxnames )`

virStoragePoolListVolumes: : pointer to storage pool : array in which to store volume names : size of names array  
Fetch list of storage volume names, limiting to at most maxnames.

To list the volume objects directly, see [virStoragePoolListAllVolumes\(\)](#).

Returns the number of names fetched, or -1 on error

**4.1.4.311** `virStoragePoolPtr virStoragePoolLookupByName ( virConnectPtr conn, const char * name )`

**4.1.4.312** `virStoragePoolPtr virStoragePoolLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

virStoragePoolLookupByUUID: : pointer to hypervisor connection : globally unique id of pool to fetch

Fetch a storage pool based on its globally unique id

Returns a virStoragePoolPtr object, or NULL if no matching pool is found

**4.1.4.313** `virStoragePoolPtr virStoragePoolLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

virStoragePoolLookupByUUIDString: : pointer to hypervisor connection : globally unique id of pool to fetch

Fetch a storage pool based on its globally unique id

Returns a virStoragePoolPtr object, or NULL if no matching pool is found

**4.1.4.314** `virStoragePoolPtr virStoragePoolLookupByVolume ( virStorageVolPtr vol )`

virStoragePoolLookupByVolume: : pointer to storage volume

Fetch a storage pool which contains a particular volume

Returns a virStoragePoolPtr object, or NULL if no matching pool is found

**4.1.4.315** `int virStoragePoolNumOfVolumes ( virStoragePoolPtr pool )`

virStoragePoolNumOfVolumes: : pointer to storage pool

Fetch the number of storage volumes within a pool

Returns the number of storage pools, or -1 on failure

**4.1.4.316** `int virStoragePoolRef ( virStoragePoolPtr pool )`

virStoragePoolRef: : the pool to hold a reference on

Increment the reference count on the pool. For each additional call to this method, there shall be a corresponding call to virStoragePoolFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a pool would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.317** `int virStoragePoolRefresh ( virStoragePoolPtr pool, unsigned int flags )`

virStoragePoolRefresh: : pointer to storage pool : extra flags; not used yet, so callers should always pass 0

Request that the pool refresh its list of volumes. This may involve communicating with a remote server, and/or initializing new devices at the OS layer

Returns 0 if the volume list was refreshed, -1 on failure

#### 4.1.4.318 int virStoragePoolSetAutostart ( virStoragePoolPtr pool, int autostart )

virStoragePoolSetAutostart: : pointer to storage pool : new flag setting

Sets the autostart flag

Returns 0 on success, -1 on failure

#### 4.1.4.319 int virStoragePoolUndefine ( virStoragePoolPtr pool )

virStoragePoolUndefine: : pointer to storage pool

Undefine an inactive storage pool

Returns 0 on success, -1 on failure

#### 4.1.4.320 virStorageVolPtr virStorageVolCreateXML ( virStoragePoolPtr pool, const char \* xmldesc, unsigned int flags )

virStorageVolCreateXML: : pointer to storage pool : description of volume to create : extra flags; not used yet, so callers should always pass 0

Create a storage volume within a pool based on an XML description. Not all pools support creation of volumes

Returns the storage volume, or NULL on error

#### 4.1.4.321 virStorageVolPtr virStorageVolCreateXMLFrom ( virStoragePoolPtr pool, const char \* xmldesc, virStorageVolPtr clonevol, unsigned int flags )

virStorageVolCreateXMLFrom: : pointer to parent pool for the new volume : description of volume to create : storage volume to use as input : extra flags; not used yet, so callers should always pass 0

Create a storage volume in the parent pool, using the 'clonevol' volume as input. Information for the new volume (name, perms) are passed via a typical volume XML description.

Returns the storage volume, or NULL on error

#### 4.1.4.322 int virStorageVolDelete ( virStorageVolPtr vol, unsigned int flags )

virStorageVolDelete: : pointer to storage volume : extra flags; not used yet, so callers should always pass 0

Delete the storage volume from the pool

Returns 0 on success, or -1 on error

#### 4.1.4.323 int virStorageVolDownload ( virStorageVolPtr vol, virStreamPtr stream, unsigned long long offset, unsigned long long length, unsigned int flags )

virStorageVolDownload: : pointer to volume to download from : stream to use as output : position in to start reading from : limit on amount of data to download : extra flags; not used yet, so callers should always pass 0

Download the content of the volume as a stream. If is zero, then the remaining contents of the volume after will be downloaded.

This call sets up an asynchronous stream; subsequent use of stream APIs is necessary to transfer the actual data, determine how much data is successfully transferred, and detect any errors. The results will be unpredictable if another active stream is writing to the storage volume.

Returns 0, or -1 upon error.

#### 4.1.4.324 `int virStorageVolFree ( virStorageVolPtr vol )`

`virStorageVolFree`: : pointer to storage volume

Release the storage volume handle. The underlying storage volume continues to exist.

Returns 0 on success, or -1 on error

#### 4.1.4.325 `virConnectPtr virStorageVolGetConnect ( virStorageVolPtr vol )`

`virStorageVolGetConnect`: : pointer to a pool

Provides the connection pointer associated with a storage volume. The reference counter on the connection is not increased by this call.

WARNING: When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the volume object together.

Returns the `virConnectPtr` or NULL in case of failure.

#### 4.1.4.326 `int virStorageVolGetInfo ( virStorageVolPtr vol, virStorageVolInfoPtr info )`

`virStorageVolGetInfo`: : pointer to storage volume : pointer at which to store info

Fetches volatile information about the storage volume such as its current allocation

Returns 0 on success, or -1 on failure

#### 4.1.4.327 `const char* virStorageVolGetKey ( virStorageVolPtr vol )`

`virStorageVolGetKey`: : pointer to storage volume

Fetch the storage volume key. This is globally unique, so the same volume will have the same key no matter what host it is accessed from

Returns the volume key, or NULL on error

#### 4.1.4.328 `const char* virStorageVolGetName ( virStorageVolPtr vol )`

`virStorageVolGetName`: : pointer to storage volume

Fetch the storage volume name. This is unique within the scope of a pool

Returns the volume name, or NULL on error

#### 4.1.4.329 `char* virStorageVolGetPath ( virStorageVolPtr vol )`

`virStorageVolGetPath`: : pointer to storage volume

Fetch the storage volume path. Depending on the pool configuration this is either persistent across hosts, or dynamically assigned at pool startup. Consult pool documentation for information on getting the persistent naming

Returns the storage volume path, or NULL on error. The caller must `free()` the returned path after use.

**4.1.4.330** `char* virStorageVolGetXMLDesc ( virStorageVolPtr vol, unsigned int flags )`

`virStorageVolGetXMLDesc`: : pointer to storage volume : extra flags; not used yet, so callers should always pass 0  
 Fetch an XML document describing all aspects of the storage volume  
 Returns the XML document, or NULL on error

**4.1.4.331** `virStorageVolPtr virStorageVolLookupByKey ( virConnectPtr conn, const char * key )`

`virStorageVolLookupByKey`: : pointer to hypervisor connection : globally unique key  
 Fetch a pointer to a storage volume based on its globally unique key  
 Returns a storage volume, or NULL if not found / error

**4.1.4.332** `virStorageVolPtr virStorageVolLookupByName ( virStoragePoolPtr pool, const char * name )`**4.1.4.333** `virStorageVolPtr virStorageVolLookupByPath ( virConnectPtr conn, const char * path )`

`virStorageVolLookupByPath`: : pointer to hypervisor connection : locally unique path  
 Fetch a pointer to a storage volume based on its locally (host) unique path  
 Returns a storage volume, or NULL if not found / error

**4.1.4.334** `int virStorageVolRef ( virStorageVolPtr vol )`

`virStorageVolRef`: : the vol to hold a reference on

Increment the reference count on the vol. For each additional call to this method, there shall be a corresponding call to `virStorageVolFree` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a vol would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.1.4.335** `int virStorageVolResize ( virStorageVolPtr vol, unsigned long long capacity, unsigned int flags )`

`virStorageVolResize`: : pointer to storage volume : new capacity, in bytes : bitwise-OR of `virStorageVolResizeFlags`  
 Changes the capacity of the storage volume to . The operation will fail if the new capacity requires allocation that would exceed the remaining free space in the parent pool. The contents of the new capacity will appear as all zero bytes.

Normally, the operation will attempt to affect capacity with a minimum impact on allocation (that is, the default operation favors a sparse resize). If contains `VIR_STORAGE_VOL_RESIZE_ALLOCATE`, then the operation will ensure that allocation is sufficient for the new capacity; this may make the operation take noticeably longer.

Normally, the operation treats as the new size in bytes; but if contains `VIR_STORAGE_VOL_RESIZE_DELTA`, then represents the size difference to add to the current size. It is up to the storage pool implementation whether unaligned requests are rounded up to the next valid boundary, or rejected.

Normally, this operation should only be used to enlarge capacity; but if contains `VIR_STORAGE_VOL_RESIZE_SHRINK`, it is possible to attempt a reduction in capacity even though it might cause data loss. If `VIR_STORAGE_VOL_RESIZE_DELTA` is also present, then is subtracted from the current size; without it, represents the absolute new size regardless of whether it is larger or smaller than the current size.

Returns 0 on success, or -1 on error.

**4.1.4.336** `int virStorageVolUpload ( virStorageVolPtr vol, virStreamPtr stream, unsigned long long offset, unsigned long long length, unsigned int flags )`

virStorageVolUpload: : pointer to volume to upload : stream to use as input : position to start writing to : limit on amount of data to upload : extra flags; not used yet, so callers should always pass 0

Upload new content to the volume from a stream. This call will fail if + exceeds the size of the volume. Otherwise, if is non-zero, an error will be raised if an attempt is made to upload greater than bytes of data.

This call sets up an asynchronous stream; subsequent use of stream APIs is necessary to transfer the actual data, determine how much data is successfully transferred, and detect any errors. The results will be unpredictable if another active stream is writing to the storage volume.

Returns 0, or -1 upon error.

**4.1.4.337** `int virStorageVolWipe ( virStorageVolPtr vol, unsigned int flags )`

virStorageVolWipe: : pointer to storage volume : extra flags; not used yet, so callers should always pass 0

Ensure data previously on a volume is not accessible to future reads

Returns 0 on success, or -1 on error

**4.1.4.338** `int virStorageVolWipePattern ( virStorageVolPtr vol, unsigned int algorithm, unsigned int flags )`

virStorageVolWipePattern: : pointer to storage volume : one of virStorageVolWipeAlgorithm : future flags, use 0 for now

Similar to virStorageVolWipe, but one can choose between different wiping algorithms.

Returns 0 on success, or -1 on error.

**4.1.4.339** `int virStreamAbort ( virStreamPtr stream )`

virStreamAbort: : pointer to the stream object

Request that the in progress data transfer be cancelled abnormally before the end of the stream has been reached. For output streams this can be used to inform the driver that the stream is being terminated early. For input streams this can be used to inform the driver that it should stop sending data.

Returns 0 on success, -1 upon error

**4.1.4.340** `int virStreamEventAddCallback ( virStreamPtr stream, int events, virStreamEventCallback cb, void * opaque, virFreeCallback ff )`

virStreamEventAddCallback: : pointer to the stream object : set of events to monitor : callback to invoke when an event occurs : application defined data : callback to free data

Register a callback to be notified when a stream becomes writable, or readable. This is most commonly used in conjunction with non-blocking data streams to integrate into an event loop

Returns 0 on success, -1 upon error

**4.1.4.341** `int virStreamEventRemoveCallback ( virStreamPtr stream )`

virStreamEventRemoveCallback: : pointer to the stream object

Remove an event callback from the stream

Returns 0 on success, -1 on error



**4.1.4.342 int virStreamEventUpdateCallback ( virStreamPtr stream, int events )**

virStreamEventUpdateCallback: : pointer to the stream object : set of events to monitor

Changes the set of events to monitor for a stream. This allows for event notification to be changed without having to unregister & register the callback completely. This method is guaranteed to succeed if a callback is already registered

Returns 0 on success, -1 if no callback is registered

**4.1.4.343 int virStreamFinish ( virStreamPtr stream )**

virStreamFinish: : pointer to the stream object

Indicate that there is no further data is to be transmitted on the stream. For output streams this should be called once all data has been written. For input streams this should be called once virStreamRecv returns end-of-file.

This method is a synchronization point for all asynchronous errors, so if this returns a success code the application can be sure that all data has been successfully processed.

Returns 0 on success, -1 upon error

**4.1.4.344 int virStreamFree ( virStreamPtr stream )**

virStreamFree: : pointer to the stream object

Decrement the reference count on a stream, releasing the stream object if the reference count has hit zero.

There must not be an active data transfer in progress when releasing the stream. If a stream needs to be disposed of prior to end of stream being reached, then the virStreamAbort function should be called first.

Returns 0 upon success, or -1 on error

**4.1.4.345 virStreamPtr virStreamNew ( virConnectPtr conn, unsigned int flags )**

virStreamNew: : pointer to the connection : bitwise-OR of virStreamFlags

Creates a new stream object which can be used to perform streamed I/O with other public API function.

When no longer needed, a stream object must be released with virStreamFree. If a data stream has been used, then the application must call virStreamFinish or virStreamAbort before free'ing to, in order to notify the driver of termination.

If a non-blocking data stream is required passed VIR\_STREAM\_NONBLOCK for flags, otherwise pass 0.

Returns the new stream, or NULL upon error

**4.1.4.346 int virStreamRecv ( virStreamPtr stream, char \* data, size\_t nbytes )**

virStreamRecv: : pointer to the stream object : buffer to read into from stream : size of buffer

Reads a series of bytes from the stream. This method may block the calling application for an arbitrary amount of time.

Errors are not guaranteed to be reported synchronously with the call, but may instead be delayed until a subsequent call.

An example using this with a hypothetical file download API looks like

```
virStreamPtr st = virStreamNew(conn, 0); int fd = open("demo.iso", O_WRONLY, 0600)
```

```
virConnectDownloadFile(conn, "demo.iso", st);
```

```
while (1) { char buf[1024]; int got = virStreamRecv(st, buf, 1024); if (got < 0) break; if (got == 0) { virStreamFinish(st); break; } int offset = 0; while (offset < got) { int sent = write(fd, buf+offset, got-offset) if (sent < 0) { virStreamAbort(st);
```

goto done; } offset += sent; } } if (virStreamFinish(st) < 0) ... report an error .... done: virStreamFree(st); close(fd);

Returns the number of bytes read, which may be less than requested.

Returns 0 when the end of the stream is reached, at which time the caller should invoke [virStreamFinish\(\)](#) to get confirmation of stream completion.

Returns -1 upon error, at which time the stream will be marked as aborted, and the caller should now release the stream with [virStreamFree](#).

Returns -2 if there is no data pending to be read & the stream is marked as non-blocking.

#### 4.1.4.347 int virStreamRecvAll ( virStreamPtr stream, virStreamSinkFunc handler, void \* opaque )

virStreamRecvAll: : pointer to the stream object : sink callback for writing data to application : application defined data

Receive the entire data stream, sending the data to the requested data sink. This is simply a convenient alternative to [virStreamRecv](#), for apps that do blocking-I/O.

An example using this with a hypothetical file download API looks like

```
int mysink(virStreamPtr st, const char *buf, int nbytes, void *opaque) { int *fd = opaque;
return write(*fd, buf, nbytes); }
```

```
virStreamPtr st = virStreamNew(conn, 0); int fd = open("demo.iso", O_WRONLY)
```

```
virConnectUploadFile(conn, st); if (virStreamRecvAll(st, mysink, &fd) < 0) { ...report an error ... goto done; } if
(virStreamFinish(st) < 0) ...report an error... virStreamFree(st); close(fd);
```

Returns 0 if all the data was successfully received. The caller should invoke [virStreamFinish\(st\)](#) to flush the stream upon success and then [virStreamFree](#)

Returns -1 upon any error, with [virStreamAbort\(\)](#) already having been called, so the caller need only call [virStreamFree\(\)](#)

#### 4.1.4.348 int virStreamRef ( virStreamPtr stream )

virStreamRef: : pointer to the stream

Increment the reference count on the stream. For each additional call to this method, there shall be a corresponding call to [virStreamFree](#) to release the reference count, once the caller no longer needs the reference to this object.

Returns 0 in case of success, -1 in case of failure

#### 4.1.4.349 int virStreamSend ( virStreamPtr stream, const char \* data, size\_t nbytes )

virStreamSend: : pointer to the stream object : buffer to write to stream : size of buffer

Write a series of bytes to the stream. This method may block the calling application for an arbitrary amount of time. Once an application has finished sending data it should call [virStreamFinish](#) to wait for successful confirmation from the driver, or detect any error.

This method may not be used if a stream source has been registered.

Errors are not guaranteed to be reported synchronously with the call, but may instead be delayed until a subsequent call.

An example using this with a hypothetical file upload API looks like

```
virStreamPtr st = virStreamNew(conn, 0); int fd = open("demo.iso", O_RDONLY)
```

```
virConnectUploadFile(conn, "demo.iso", st);
```

```
while (1) { char buf[1024]; int got = read(fd, buf, 1024); if (got < 0) { virStreamAbort(st); break; } if (got == 0) {
virStreamFinish(st); break; } int offset = 0; while (offset < got) { int sent = virStreamSend(st, buf+offset, got-offset)
```

```
if (sent < 0) { virStreamAbort(st); goto done; } offset += sent; } } if (virStreamFinish(st) < 0) ... report an error ...
done: virStreamFree(st); close(fd);
```

Returns the number of bytes written, which may be less than requested.

Returns -1 upon error, at which time the stream will be marked as aborted, and the caller should now release the stream with `virStreamFree`.

Returns -2 if the outgoing transmit buffers are full & the stream is marked as non-blocking.

#### 4.1.4.350 int virStreamSendAll ( virStreamPtr stream, virStreamSourceFunc handler, void \* opaque )

`virStreamSendAll`: : pointer to the stream object : source callback for reading data from application : application defined data

Send the entire data stream, reading the data from the requested data source. This is simply a convenient alternative to `virStreamSend`, for apps that do blocking-I/O.

An example using this with a hypothetical file upload API looks like

```
int mysource(virStreamPtr st, char *buf, int nbytes, void *opaque) { int *fd = opaque;
return read(*fd, buf, nbytes); }
```

```
virStreamPtr st = virStreamNew(conn, 0); int fd = open("demo.iso", O_RDONLY)
```

```
virConnectUploadFile(conn, st); if (virStreamSendAll(st, mysource, &fd) < 0) { ...report an error ... goto done; } if
(virStreamFinish(st) < 0) ...report an error... virStreamFree(st); close(fd);
```

Returns 0 if all the data was successfully sent. The caller should invoke `virStreamFinish(st)` to flush the stream upon success and then `virStreamFree`

Returns -1 upon any error, with [virStreamAbort\(\)](#) already having been called, so the caller need only call [virStreamFree\(\)](#)

### 4.1.5 Variable Documentation

#### 4.1.5.1 VIR\_EXPORT\_VAR virConnectAuthPtr virConnectAuthPtrDefault

## 4.2 src/conf/domain\_conf.c File Reference

```
#include <config.h>
```

```
#include <dirent.h>
#include <fcntl.h>
#include <strings.h>
#include <sys/stat.h>
#include <unistd.h>
#include "internal.h"
#include "virterror_internal.h"
#include "datatypes.h"
#include "domain_conf.h"
#include "snapshot_conf.h"
#include "memory.h"
#include "verify.h"
#include "uuid.h"
#include "logging.h"
#include "nwfilter_conf.h"
#include "storage_file.h"
#include "virfile.h"
#include "count-one-bits.h"
#include "secret_conf.h"
#include "netdev_vport_profile_conf.h"
#include "netdev_bandwidth_conf.h"
#include "netdev_vlan_conf.h"
#include "device_conf.h"
```

## Data Structures

- struct [virDomainIDData](#)
- struct [virDomainNameData](#)
- struct [virDomainListData](#)

## Macros

- `#define VIR_FROM_THIS VIR_FROM_DOMAIN`
- `#define VIR_DOMAIN_NOSTATE_LAST (VIR_DOMAIN_NOSTATE_UNKNOWN + 1)`
- `#define VIR_DOMAIN_RUNNING_LAST (VIR_DOMAIN_RUNNING_SAVE_CANCELED + 1)`
- `#define VIR_DOMAIN_BLOCKED_LAST (VIR_DOMAIN_BLOCKED_UNKNOWN + 1)`
- `#define VIR_DOMAIN_PAUSED_LAST (VIR_DOMAIN_PAUSED_SHUTTING_DOWN + 1)`
- `#define VIR_DOMAIN_SHUTDOWN_LAST (VIR_DOMAIN_SHUTDOWN_USER + 1)`
- `#define VIR_DOMAIN_SHUTOFF_LAST (VIR_DOMAIN_SHUTOFF_FROM_SNAPSHOT + 1)`
- `#define VIR_DOMAIN_CRASHED_LAST (VIR_DOMAIN_CRASHED_UNKNOWN + 1)`
- `#define VIR_DOMAIN_XML_WRITE_FLAGS VIR_DOMAIN_XML_SECURE`
- `#define VIR_DOMAIN_XML_READ_FLAGS VIR_DOMAIN_XML_INACTIVE`
- `#define NET_MODEL_CHARS "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ091234567890_-"`
- `#define DUMPXML_FLAGS`
- `#define MATCH(FLAG) (data->flags & (FLAG))`

## Enumerations

- enum [virDomainXMLInternalFlags](#) {  
[VIR\\_DOMAIN\\_XML\\_INTERNAL\\_STATUS](#) = (1<<16), [VIR\\_DOMAIN\\_XML\\_INTERNAL\\_ACTUAL\\_NET](#) = (1<<17), [VIR\\_DOMAIN\\_XML\\_INTERNAL\\_PCI\\_ORIG\\_STATES](#) = (1<<18), [VIR\\_DOMAIN\\_XML\\_INTERNAL\\_ALLOW\\_ROM](#) = (1<<19),  
[VIR\\_DOMAIN\\_XML\\_INTERNAL\\_ALLOW\\_BOOT](#) = (1<<20) }

## Functions

- [verify](#) ([VIR\\_DOMAIN\\_VIRT\\_LAST](#) ≤ 32)
- [VIR\\_ENUM\\_IMPL](#) ([virDomainTaint](#), [VIR\\_DOMAIN\\_TAINT\\_LAST](#), "custom-argv", "custom-monitor", "high-privileges", "shell-scripts", "disk-probing", "external-launch", "host-cpu")
- [VIR\\_ENUM\\_IMPL](#) ([virDomainVirt](#), [VIR\\_DOMAIN\\_VIRT\\_LAST](#), [VIR\\_ENUM\\_IMPL](#)("qemu", "kqemu", "kvm", [VIR\\_ENUM\\_IMPL](#)("xen", [VIR\\_ENUM\\_IMPL](#)("lxc", [VIR\\_ENUM\\_IMPL](#)("uml", "openvz", "test", "vmware", "hyperv", [VIR\\_ENUM\\_IMPL](#)("vbox", "phyp", "parallels"))
- void [virBlkioDeviceWeightArrayClear](#) ([virBlkioDeviceWeightPtr](#) deviceWeights, int ndevices)
- static int [virDomainBlkioDeviceWeightParseXML](#) ([xmlNodePtr](#) root, [virBlkioDeviceWeightPtr](#) dw)
- static void [virDomainObjListDataFree](#) (void \*payload, const void \*name [ATTRIBUTE\\_UNUSED](#))
- int [virDomainObjListInit](#) ([virDomainObjListPtr](#) doms)
- void [virDomainObjListDeinit](#) ([virDomainObjListPtr](#) doms)
- static int [virDomainObjListSearchID](#) (const void \*payload, const void \*name [ATTRIBUTE\\_UNUSED](#), const void \*data)
- [virDomainObjPtr](#) [virDomainFindByID](#) (const [virDomainObjListPtr](#) doms, int id)
- [virDomainObjPtr](#) [virDomainFindByUUID](#) (const [virDomainObjListPtr](#) doms, const unsigned char \*uuid)
- static int [virDomainObjListSearchName](#) (const void \*payload, const void \*name [ATTRIBUTE\\_UNUSED](#), const void \*data)
- [virDomainObjPtr](#) [virDomainFindByName](#) (const [virDomainObjListPtr](#) doms, const char \*name)
- bool [virDomainObjTaint](#) ([virDomainObjPtr](#) obj, enum [virDomainTaintFlags](#) taint)
- static void [virDomainDeviceInfoFree](#) ([virDomainDeviceInfoPtr](#) info)
- static void [virDomainGraphicsAuthDefClear](#) ([virDomainGraphicsAuthDefPtr](#) def)
- static void [virDomainGraphicsListenDefClear](#) ([virDomainGraphicsListenDefPtr](#) def)
- static void [virSecurityLabelDefFree](#) ([virSecurityLabelDefPtr](#) def)
- static void [virSecurityDeviceLabelDefFree](#) ([virSecurityDeviceLabelDefPtr](#) def)
- void [virDomainGraphicsDefFree](#) ([virDomainGraphicsDefPtr](#) def)
- void [virDomainInputDefFree](#) ([virDomainInputDefPtr](#) def)
- void [virDomainLeaseDefFree](#) ([virDomainLeaseDefPtr](#) def)
- void [virDomainDiskDefFree](#) ([virDomainDiskDefPtr](#) def)
- void [virDomainDiskHostDefFree](#) ([virDomainDiskHostDefPtr](#) def)
- void [virDomainControllerDefFree](#) ([virDomainControllerDefPtr](#) def)
- void [virDomainFSDefFree](#) ([virDomainFSDefPtr](#) def)
- [POL Mod](#) void [virDomainActualNetDefFree](#) ([virDomainActualNetDefPtr](#) def)
- [POL Mod](#) void [virDomainNetDefFree](#) ([virDomainNetDefPtr](#) def)
- static void [ATTRIBUTE\\_NONNULL](#) (1)
- int [virDomainChrSourceDefCopy](#) ([virDomainChrSourceDefPtr](#) dest, [virDomainChrSourceDefPtr](#) src)
- void [virDomainChrSourceDefFree](#) ([virDomainChrSourceDefPtr](#) def)
- static bool [virDomainChrSourceDefsEqual](#) (const [virDomainChrSourceDef](#) \*src, const [virDomainChrSourceDef](#) \*tgt)
- void [virDomainChrDefFree](#) ([virDomainChrDefPtr](#) def)
- void [virDomainSmartcardDefFree](#) ([virDomainSmartcardDefPtr](#) def)
- void [virDomainSoundCodecDefFree](#) ([virDomainSoundCodecDefPtr](#) def)
- void [virDomainSoundDefFree](#) ([virDomainSoundDefPtr](#) def)
- void [virDomainMemballoonDefFree](#) ([virDomainMemballoonDefPtr](#) def)
- void [virDomainWatchdogDefFree](#) ([virDomainWatchdogDefPtr](#) def)
- void [virDomainVideoDefFree](#) ([virDomainVideoDefPtr](#) def)
- [virDomainHostdevDefPtr](#) [virDomainHostdevDefAlloc](#) (void)
- void [virDomainHostdevDefClear](#) ([virDomainHostdevDefPtr](#) def)
- void [virDomainHostdevDefFree](#) ([virDomainHostdevDefPtr](#) def)
- void [virDomainHubDefFree](#) ([virDomainHubDefPtr](#) def)
- void [virDomainRedirdevDefFree](#) ([virDomainRedirdevDefPtr](#) def)
- void [virDomainRedirFilterDefFree](#) ([virDomainRedirFilterDefPtr](#) def)
- void [virDomainDeviceDefFree](#) ([virDomainDeviceDefPtr](#) def)
- static void [virDomainClockDefClear](#) ([virDomainClockDefPtr](#) def)

- `virDomainVcpuPinDefPtr * virDomainVcpuPinDefCopy (virDomainVcpuPinDefPtr *src, int nvcpupin)`
- `void virDomainVcpuPinDefFree (virDomainVcpuPinDefPtr def)`
- `void virDomainVcpuPinDefArrayFree (virDomainVcpuPinDefPtr *def, int nvcpupin)`
- `void virDomainDefFree (virDomainDefPtr def)`
- `static void virDomainObjDispose (void *obj)`
- `virDomainObjPtr virDomainObjNew (virCapsPtr caps)`
- `void virDomainObjAssignDef (virDomainObjPtr domain, const virDomainDefPtr def, bool live)`
- `virDomainObjPtr virDomainAssignDef (virCapsPtr caps, virDomainObjListPtr doms, const virDomainDefPtr def, bool live)`
- `int virDomainObjSetDefTransient (virCapsPtr caps, virDomainObjPtr domain, bool live)`
- `virDomainDefPtr virDomainObjGetPersistentDef (virCapsPtr caps, virDomainObjPtr domain)`
- `int virDomainLiveConfigHelperMethod (virCapsPtr caps, virDomainObjPtr dom, unsigned int *flags, virDomainDefPtr *persistentDef)`
- `void virDomainRemoveInactive (virDomainObjListPtr doms, virDomainObjPtr dom)`
- `int virDomainDeviceAddressIsValid (virDomainDeviceInfoPtr info, int type)`
- `static bool virDomainDeviceInfosSet (virDomainDeviceInfoPtr info, unsigned int flags)`
- `void virDomainDeviceInfoClear (virDomainDeviceInfoPtr info)`
- `static int virDomainDeviceInfoClearAlias (virDomainDefPtr def ATTRIBUTE_UNUSED, virDomainDeviceDefPtr device ATTRIBUTE_UNUSED, virDomainDeviceInfoPtr info, void *opaque ATTRIBUTE_UNUSED)`
- `static int virDomainDeviceInfoClearPCIAddress (virDomainDefPtr def ATTRIBUTE_UNUSED, virDomainDeviceDefPtr device ATTRIBUTE_UNUSED, virDomainDeviceInfoPtr info, void *opaque ATTRIBUTE_UNUSED)`
- `int virDomainDeviceInfoIterate (virDomainDefPtr def, virDomainDeviceInfoCallback cb, void *opaque)`
- `void virDomainDefClearPCIAddresses (virDomainDefPtr def)`
- `void virDomainDefClearDeviceAliases (virDomainDefPtr def)`
- `static int ATTRIBUTE_NONNULL (2)`
- `static int virDomainDeviceDriveAddressParseXML (xmlNodePtr node, virDomainDeviceDriveAddressPtr addr)`
- `static int virDomainDeviceVirtioSerialAddressParseXML (xmlNodePtr node, virDomainDeviceVirtioSerialAddressPtr addr)`
- `static int virDomainDeviceCcidAddressParseXML (xmlNodePtr node, virDomainDeviceCcidAddressPtr addr)`
- `static int virDomainDeviceUSBAddressParseXML (xmlNodePtr node, virDomainDeviceUSBAddressPtr addr)`
- `static int virDomainDeviceSpaprVioAddressParseXML (xmlNodePtr node, virDomainDeviceSpaprVioAddressPtr addr)`
- `static int virDomainDeviceUSBMasterParseXML (xmlNodePtr node, virDomainDeviceUSBMasterPtr master)`
- `static int virDomainDeviceBootParseXML (xmlNodePtr node, int *bootIndex, virBitmapPtr bootMap)`
- `static int virDomainDeviceInfoParseXML (xmlNodePtr node, virBitmapPtr bootMap, virDomainDeviceInfoPtr info, unsigned int flags)`
- `static int virDomainParseLegacyDeviceAddress (char *devaddr, virDevicePCIAddressPtr pci)`
- `static int virDomainHostdevSubsysUsbDefParseXML (const xmlNodePtr node, virDomainHostdevDefPtr def)`
- `static int virDomainHostdevSubsysPciOrigStatesDefParseXML (const xmlNodePtr node, virDomainHostdevOrigStatesPtr def)`
- `static int virDomainHostdevSubsysPciDefParseXML (const xmlNodePtr node, virDomainHostdevDefPtr def, unsigned int flags)`
- `static int virDomainHostdevPartsParse (xmlNodePtr node, xmlXPathContextPtr ctxt, const char *mode, const char *type, virDomainHostdevDefPtr def, unsigned int flags)`
- `int virDomainDiskFindControllerModel (virDomainDefPtr def, virDomainDiskDefPtr disk, int controllerType)`
- `int virDomainDiskDefAssignAddress (virCapsPtr caps, virDomainDiskDefPtr def)`
- `static virSecurityLabelDefPtr virSecurityLabelDefParseXML (xmlXPathContextPtr ctxt, unsigned int flags)`
- `static int virSecurityLabelDefsParseXML (virDomainDefPtr def, xmlXPathContextPtr ctxt, virCapsPtr caps, unsigned int flags)`
- `static int virSecurityDeviceLabelDefParseXML (virSecurityDeviceLabelDefPtr **seclabels_rtn, size_t *nseclabels_rtn, virSecurityLabelDefPtr *vmSeclabels, int nvMSeclabels, xmlXPathContextPtr ctxt)`
- `static virDomainLeaseDefPtr virDomainLeaseDefParseXML (xmlNodePtr node)`

- static [virDomainDiskDefPtr](#) [virDomainDiskDefParseXML](#) ([virCapsPtr](#) caps, [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt, [virBitmapPtr](#) bootMap, [virSecurityLabelDefPtr](#) \*vmSeclabels, int nvmSeclabels, unsigned int flags)
- static int [virDomainControllerModelTypeFromString](#) (const [virDomainControllerDefPtr](#) def, const char \*model)
- static [virDomainControllerDefPtr](#) [virDomainControllerDefParseXML](#) ([xmlNodePtr](#) node, unsigned int flags)
- static int [virDomainParseScaledValue](#) (const char \*xpath, [xmlXPathContextPtr](#) ctxt, unsigned long long \*val, unsigned long long scale, unsigned long long max, bool required)
- static [virDomainFSDefPtr](#) [virDomainFSDefParseXML](#) ([xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt, unsigned int flags)
- static int [virDomainActualNetDefParseXML](#) ([xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt, [virDomainNetDefPtr](#) parent, [virDomainActualNetDefPtr](#) \*def, unsigned int flags)
- [POL Mod](#) static [virDomainNetDefPtr](#) [virDomainNetDefParseXML](#) ([virCapsPtr](#) caps, [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt, [virBitmapPtr](#) bootMap, unsigned int flags)
- static int [virDomainChrDefaultTargetType](#) ([virCapsPtr](#) caps, [virDomainDefPtr](#) def, int devtype)
- static int [virDomainChrTargetTypeFromString](#) ([virCapsPtr](#) caps, [virDomainDefPtr](#) def, int devtype, const char \*targetType)
- static int [virDomainChrDefParseTargetXML](#) ([virCapsPtr](#) caps, [virDomainDefPtr](#) vmdef, [virDomainChrDefPtr](#) def, [xmlNodePtr](#) cur)
- static int [virDomainChrSourceDefParseXML](#) ([virDomainChrSourceDefPtr](#) def, [xmlNodePtr](#) cur, unsigned int flags, [virDomainChrDefPtr](#) chr\_def, [xmlXPathContextPtr](#) ctxt, [virSecurityLabelDefPtr](#) \*vmSeclabels, int nvmSeclabels)
- [virDomainChrDefPtr](#) [virDomainChrDefNew](#) (void)
- static [virDomainChrDefPtr](#) [virDomainChrDefParseXML](#) ([virCapsPtr](#) caps, [virDomainDefPtr](#) vmdef, [xmlXPathContextPtr](#) ctxt, [xmlNodePtr](#) node, [virSecurityLabelDefPtr](#) \*vmSeclabels, int nvmSeclabels, unsigned int flags)
- static [virDomainSmartcardDefPtr](#) [virDomainSmartcardDefParseXML](#) ([xmlNodePtr](#) node, unsigned int flags)
- static [virDomainInputDefPtr](#) [virDomainInputDefParseXML](#) (const char \*ostype, [xmlNodePtr](#) node, unsigned int flags)
- static [virDomainHubDefPtr](#) [virDomainHubDefParseXML](#) ([xmlNodePtr](#) node, unsigned int flags)
- static [virDomainTimerDefPtr](#) [virDomainTimerDefParseXML](#) (const [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt)
- static int [virDomainGraphicsAuthDefParseXML](#) ([xmlNodePtr](#) node, [virDomainGraphicsAuthDefPtr](#) def, int type)
- static int [virDomainGraphicsListenDefParseXML](#) ([virDomainGraphicsListenDefPtr](#) def, [xmlNodePtr](#) node, unsigned int flags)
- static [virDomainGraphicsDefPtr](#) [virDomainGraphicsDefParseXML](#) ([xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt, unsigned int flags)
- static [virDomainSoundCodecDefPtr](#) [virDomainSoundCodecDefParseXML](#) (const [xmlNodePtr](#) node)
- static [virDomainSoundDefPtr](#) [virDomainSoundDefParseXML](#) (const [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt, unsigned int flags)
- static [virDomainWatchdogDefPtr](#) [virDomainWatchdogDefParseXML](#) (const [xmlNodePtr](#) node, unsigned int flags)
- static [virDomainMemballoonDefPtr](#) [virDomainMemballoonDefParseXML](#) (const [xmlNodePtr](#) node, unsigned int flags)
- static [virSysinfoDefPtr](#) [virSysinfoParseXML](#) (const [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt)
- int [virDomainVideoDefaultRAM](#) ([virDomainDefPtr](#) def, int type)
- int [virDomainVideoDefaultType](#) ([virDomainDefPtr](#) def)
- static [virDomainVideoAccelDefPtr](#) [virDomainVideoAccelDefParseXML](#) (const [xmlNodePtr](#) node)
- static [virDomainVideoDefPtr](#) [virDomainVideoDefParseXML](#) (const [xmlNodePtr](#) node, [virDomainDefPtr](#) dom, unsigned int flags)
- static [virDomainHostdevDefPtr](#) [virDomainHostdevDefParseXML](#) (const [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt, [virBitmapPtr](#) bootMap, unsigned int flags)
- static [virDomainRedirdevDefPtr](#) [virDomainRedirdevDefParseXML](#) (const [xmlNodePtr](#) node, [virBitmapPtr](#) bootMap, unsigned int flags)
- static int [virDomainRedirFilterUsbVersionHelper](#) (const char \*version, [virDomainRedirFilterUsbDevDefPtr](#) def)



- static  
[virDomainRedirFilterUsbDevDefPtr](#) [virDomainRedirFilterUsbDevDefParseXML](#) (const xmlNodePtr node)
- static [virDomainRedirFilterDefPtr](#) [virDomainRedirFilterDefParseXML](#) (const xmlNodePtr node, xmlXPathContextPtr ctxt)
- static int [virDomainLifecycleParseXML](#) (xmlXPathContextPtr ctxt, const char \*xpath, int \*val, int defaultVal, [virLifecycleFromStringFunc](#) convFunc)
- static int [virDomainPMStateParseXML](#) (xmlXPathContextPtr ctxt, const char \*xpath, int \*val)
- [virDomainDeviceDefPtr](#) [virDomainDeviceDefParse](#) (virCapsPtr caps, [virDomainDefPtr](#) def, const char \*xmlStr, unsigned int flags)
- static const char \* [virDomainChrTargetTypeToString](#) (int deviceType, int targetType)
- int [virDomainHostdevInsert](#) ([virDomainDefPtr](#) def, [virDomainHostdevDefPtr](#) hostdev)
- [virDomainHostdevDefPtr](#) [virDomainHostdevRemove](#) ([virDomainDefPtr](#) def, size\_t i)
- int [virDomainHostdevFind](#) ([virDomainDefPtr](#) def, [virDomainHostdevDefPtr](#) match, [virDomainHostdevDefPtr](#) \*found)
- int [virDomainDiskIndexByName](#) ([virDomainDefPtr](#) def, const char \*name, bool allow\_ambiguous)
- const char \* [virDomainDiskPathByName](#) ([virDomainDefPtr](#) def, const char \*name)
- int [virDomainDiskInsert](#) ([virDomainDefPtr](#) def, [virDomainDiskDefPtr](#) disk)
- void [virDomainDiskInsertPreAlloced](#) ([virDomainDefPtr](#) def, [virDomainDiskDefPtr](#) disk)
- [virDomainDiskDefPtr](#) [virDomainDiskRemove](#) ([virDomainDefPtr](#) def, size\_t i)
- [virDomainDiskDefPtr](#) [virDomainDiskRemoveByName](#) ([virDomainDefPtr](#) def, const char \*name)
- int [virDomainNetInsert](#) ([virDomainDefPtr](#) def, [virDomainNetDefPtr](#) net)
- int [virDomainNetIndexByMac](#) ([virDomainDefPtr](#) def, const virMacAddrPtr mac)
- [virDomainNetDefPtr](#) [virDomainNetRemove](#) ([virDomainDefPtr](#) def, size\_t i)
- [virDomainNetDefPtr](#) [virDomainNetRemoveByMac](#) ([virDomainDefPtr](#) def, const virMacAddrPtr mac)
- int [virDomainControllerInsert](#) ([virDomainDefPtr](#) def, [virDomainControllerDefPtr](#) controller)
- void [virDomainControllerInsertPreAlloced](#) ([virDomainDefPtr](#) def, [virDomainControllerDefPtr](#) controller)
- int [virDomainControllerFind](#) ([virDomainDefPtr](#) def, int type, int idx)
- [virDomainControllerDefPtr](#) [virDomainControllerRemove](#) ([virDomainDefPtr](#) def, size\_t i)
- int [virDomainLeaseIndex](#) ([virDomainDefPtr](#) def, [virDomainLeaseDefPtr](#) lease)
- int [virDomainLeaseInsertPreAlloc](#) ([virDomainDefPtr](#) def)
- int [virDomainLeaseInsert](#) ([virDomainDefPtr](#) def, [virDomainLeaseDefPtr](#) lease)
- void [virDomainLeaseInsertPreAlloced](#) ([virDomainDefPtr](#) def, [virDomainLeaseDefPtr](#) lease)
- [virDomainLeaseDefPtr](#) [virDomainLeaseRemoveAt](#) ([virDomainDefPtr](#) def, size\_t i)
- [virDomainLeaseDefPtr](#) [virDomainLeaseRemove](#) ([virDomainDefPtr](#) def, [virDomainLeaseDefPtr](#) lease)
- static char \* [virDomainDefDefaultEmulator](#) ([virDomainDefPtr](#) def, virCapsPtr caps)
- static int [virDomainDefParseBootXML](#) (xmlXPathContextPtr ctxt, [virDomainDefPtr](#) def, unsigned long \*bootCount)
- static [virDomainVcpuPinDefPtr](#) [virDomainVcpuPinDefParseXML](#) (const xmlNodePtr node, xmlXPathContextPtr ctxt, int maxvcpus, int emulator)
- static int [virDomainDefMaybeAddController](#) ([virDomainDefPtr](#) def, int type, int idx)
- static int [virDomainParseMemory](#) (const char \*xpath, xmlXPathContextPtr ctxt, unsigned long long \*mem, bool required)
- static [virDomainDefPtr](#) [virDomainDefParseXML](#) (virCapsPtr caps, xmlDocPtr xml, xmlNodePtr root, xmlXPathContextPtr ctxt, unsigned int expectedVirtTypes, unsigned int flags)
- static [virDomainObjPtr](#) [virDomainObjParseXML](#) (virCapsPtr caps, xmlDocPtr xml, xmlXPathContextPtr ctxt, unsigned int expectedVirtTypes, unsigned int flags)
- static [virDomainDefPtr](#) [virDomainDefParse](#) (const char \*xmlStr, const char \*filename, virCapsPtr caps, unsigned int expectedVirtTypes, unsigned int flags)
- [virDomainDefPtr](#) [virDomainDefParseString](#) (virCapsPtr caps, const char \*xmlStr, unsigned int expectedVirtTypes, unsigned int flags)
- [virDomainDefPtr](#) [virDomainDefParseFile](#) (virCapsPtr caps, const char \*filename, unsigned int expectedVirtTypes, unsigned int flags)
- [virDomainDefPtr](#) [virDomainDefParseNode](#) (virCapsPtr caps, xmlDocPtr xml, xmlNodePtr root, unsigned int expectedVirtTypes, unsigned int flags)
- static [virDomainObjPtr](#) [virDomainObjParseNode](#) (virCapsPtr caps, xmlDocPtr xml, xmlNodePtr root, unsigned int expectedVirtTypes, unsigned int flags)



- static [virDomainObjPtr](#) [virDomainObjParseFile](#) ([virCapsPtr](#) caps, const char \*filename, unsigned int expectedVirtTypes, unsigned int flags)
- static bool [virDomainTimerDefCheckABIStability](#) ([virDomainTimerDefPtr](#) src, [virDomainTimerDefPtr](#) dst)
- static bool [virDomainDeviceInfoCheckABIStability](#) ([virDomainDeviceInfoPtr](#) src, [virDomainDeviceInfoPtr](#) dst)
- static bool [virDomainDiskDefCheckABIStability](#) ([virDomainDiskDefPtr](#) src, [virDomainDiskDefPtr](#) dst)
- static bool [virDomainControllerDefCheckABIStability](#) ([virDomainControllerDefPtr](#) src, [virDomainControllerDefPtr](#) dst)
- static bool [virDomainFsDefCheckABIStability](#) ([virDomainFSDefPtr](#) src, [virDomainFSDefPtr](#) dst)
- static bool [virDomainNetDefCheckABIStability](#) ([virDomainNetDefPtr](#) src, [virDomainNetDefPtr](#) dst)
- static bool [virDomainInputDefCheckABIStability](#) ([virDomainInputDefPtr](#) src, [virDomainInputDefPtr](#) dst)
- static bool [virDomainSoundDefCheckABIStability](#) ([virDomainSoundDefPtr](#) src, [virDomainSoundDefPtr](#) dst)
- static bool [virDomainVideoDefCheckABIStability](#) ([virDomainVideoDefPtr](#) src, [virDomainVideoDefPtr](#) dst)
- static bool [virDomainHostdevDefCheckABIStability](#) ([virDomainHostdevDefPtr](#) src, [virDomainHostdevDefPtr](#) dst)
- static bool [virDomainSmartcardDefCheckABIStability](#) ([virDomainSmartcardDefPtr](#) src, [virDomainSmartcardDefPtr](#) dst)
- static bool [virDomainSerialDefCheckABIStability](#) ([virDomainChrDefPtr](#) src, [virDomainChrDefPtr](#) dst)
- static bool [virDomainParallelDefCheckABIStability](#) ([virDomainChrDefPtr](#) src, [virDomainChrDefPtr](#) dst)
- static bool [virDomainChannelDefCheckABIStability](#) ([virDomainChrDefPtr](#) src, [virDomainChrDefPtr](#) dst)
- static bool [virDomainConsoleDefCheckABIStability](#) ([virDomainChrDefPtr](#) src, [virDomainChrDefPtr](#) dst)
- static bool [virDomainWatchdogDefCheckABIStability](#) ([virDomainWatchdogDefPtr](#) src, [virDomainWatchdogDefPtr](#) dst)
- static bool [virDomainMemballoonDefCheckABIStability](#) ([virDomainMemballoonDefPtr](#) src, [virDomainMemballoonDefPtr](#) dst)
- static bool [virDomainHubDefCheckABIStability](#) ([virDomainHubDefPtr](#) src, [virDomainHubDefPtr](#) dst)
- static bool [virDomainRedirFilterDefCheckABIStability](#) ([virDomainRedirFilterDefPtr](#) src, [virDomainRedirFilterDefPtr](#) dst)
- bool [virDomainDefCheckABIStability](#) ([virDomainDefPtr](#) src, [virDomainDefPtr](#) dst)
- static int [virDomainDefAddDiskControllersForType](#) ([virDomainDefPtr](#) def, int controllerType, int diskBus)
- static int [virDomainDefMaybeAddVirtioSerialController](#) ([virDomainDefPtr](#) def)
- static int [virDomainDefMaybeAddSmartcardController](#) ([virDomainDefPtr](#) def)
- int [virDomainDefAddImplicitControllers](#) ([virDomainDefPtr](#) def)
- int [virDomainVcpuPinIsDuplicate](#) ([virDomainVcpuPinDefPtr](#) \*def, int nvcpupin, int vcpu)
- [virDomainVcpuPinDefPtr](#) [virDomainVcpuPinFindByVcpu](#) ([virDomainVcpuPinDefPtr](#) \*def, int nvcpupin, int vcpu)
- int [virDomainVcpuPinAdd](#) ([virDomainVcpuPinDefPtr](#) \*\*vcpupin\_list, int \*nvcpupin, unsigned char \*cpumap, int maplen, int vcpu)
- int [virDomainVcpuPinDel](#) ([virDomainDefPtr](#) def, int vcpu)
- int [virDomainEmulatorPinAdd](#) ([virDomainDefPtr](#) def, unsigned char \*cpumap, int maplen)
- int [virDomainEmulatorPinDel](#) ([virDomainDefPtr](#) def)
- static int [virDomainLifecycleDefFormat](#) ([virBufferPtr](#) buf, int type, const char \*name, [virLifecycleToStringFunc](#) convFunc)
- static void [virSecurityLabelDefFormat](#) ([virBufferPtr](#) buf, [virSecurityLabelDefPtr](#) def)
- static void [virSecurityDeviceLabelDefFormat](#) ([virBufferPtr](#) buf, [virSecurityDeviceLabelDefPtr](#) def)
- static int [virDomainLeaseDefFormat](#) ([virBufferPtr](#) buf, [virDomainLeaseDefPtr](#) def)
- static void [virDomainDiskGeometryDefFormat](#) ([virBufferPtr](#) buf, [virDomainDiskDefPtr](#) def)
- static void [virDomainDiskBlockIoDefFormat](#) ([virBufferPtr](#) buf, [virDomainDiskDefPtr](#) def)
- static int [virDomainDiskDefFormat](#) ([virBufferPtr](#) buf, [virDomainDiskDefPtr](#) def, unsigned int flags)
- static const char \* [virDomainControllerModelTypeToString](#) ([virDomainControllerDefPtr](#) def, int model)
- static int [virDomainControllerDefFormat](#) ([virBufferPtr](#) buf, [virDomainControllerDefPtr](#) def, unsigned int flags)
- int [virDomainFSIndexByName](#) ([virDomainDefPtr](#) def, const char \*name)
- static int [virDomainFSDefFormat](#) ([virBufferPtr](#) buf, [virDomainFSDefPtr](#) def, unsigned int flags)
- static int [virDomainHostdevSourceFormat](#) ([virBufferPtr](#) buf, [virDomainHostdevDefPtr](#) def, unsigned int flags, bool includeTypeInAddr)
- [POL Mod](#) static int [virDomainActualNetDefFormat](#) ([virBufferPtr](#) buf, [virDomainActualNetDefPtr](#) def, unsigned int flags)

- static int [virDomainNetDefFormat](#) (virBufferPtr buf, [virDomainNetDefPtr](#) def, unsigned int flags)
- static int [virDomainChrSourceDefFormat](#) (virBufferPtr buf, [virDomainChrSourceDefPtr](#) def, bool tty\_compat, unsigned int flags)
- static int [virDomainChrDefFormat](#) (virBufferPtr buf, [virDomainChrDefPtr](#) def, unsigned int flags)
- static int [virDomainSmartcardDefFormat](#) (virBufferPtr buf, [virDomainSmartcardDefPtr](#) def, unsigned int flags)
- static int [virDomainSoundCodecDefFormat](#) (virBufferPtr buf, [virDomainSoundCodecDefPtr](#) def)
- static int [virDomainSoundDefFormat](#) (virBufferPtr buf, [virDomainSoundDefPtr](#) def, unsigned int flags)
- static int [virDomainMemballoonDefFormat](#) (virBufferPtr buf, [virDomainMemballoonDefPtr](#) def, unsigned int flags)
- static int [virDomainSysinfoDefFormat](#) (virBufferPtr buf, [virDomainSysinfoDefPtr](#) def)
- static int [virDomainWatchdogDefFormat](#) (virBufferPtr buf, [virDomainWatchdogDefPtr](#) def, unsigned int flags)
- static void [virDomainVideoAccelDefFormat](#) (virBufferPtr buf, [virDomainVideoAccelDefPtr](#) def)
- static int [virDomainVideoDefFormat](#) (virBufferPtr buf, [virDomainVideoDefPtr](#) def, unsigned int flags)
- static int [virDomainInputDefFormat](#) (virBufferPtr buf, [virDomainInputDefPtr](#) def, unsigned int flags)
- static int [virDomainTimerDefFormat](#) (virBufferPtr buf, [virDomainTimerDefPtr](#) def)
- static void [virDomainGraphicsAuthDefFormatAttr](#) (virBufferPtr buf, [virDomainGraphicsAuthDefPtr](#) def, unsigned int flags)
- static void [virDomainGraphicsListenDefFormat](#) (virBufferPtr buf, [virDomainGraphicsListenDefPtr](#) def, unsigned int flags)
- static int [virDomainGraphicsDefFormat](#) (virBufferPtr buf, [virDomainGraphicsDefPtr](#) def, unsigned int flags)
- static int [virDomainHostdevDefFormat](#) (virBufferPtr buf, [virDomainHostdevDefPtr](#) def, unsigned int flags)
- static int [virDomainRedirdevDefFormat](#) (virBufferPtr buf, [virDomainRedirdevDefPtr](#) def, unsigned int flags)
- static int [virDomainRedirFilterDefFormat](#) (virBufferPtr buf, [virDomainRedirFilterDefPtr](#) filter)
- static int [virDomainHubDefFormat](#) (virBufferPtr buf, [virDomainHubDefPtr](#) def, unsigned int flags)
- [verify](#) ((([VIR\\_DOMAIN\\_XML\\_INTERNAL\\_STATUS](#)|[VIR\\_DOMAIN\\_XML\\_INTERNAL\\_ACTUAL\\_NET](#)|[VIR\\_DOMAIN\\_XML\\_INTERNAL\\_PCI\\_ORIG\\_STATES](#))&[DUMPXML\\_FLAGS](#))==0)
- int [virDomainDefFormatInternal](#) ([virDomainDefPtr](#) def, unsigned int flags, virBufferPtr buf)
- char \* [virDomainDefFormat](#) ([virDomainDefPtr](#) def, unsigned int flags)
- static char \* [virDomainObjFormat](#) (virCapsPtr caps, [virDomainObjPtr](#) obj, unsigned int flags)
- static bool [virDomainDefHasUSB](#) ([virDomainDefPtr](#) def)
- static bool [virDomainDevicelsUSB](#) ([virDomainDeviceDefPtr](#) dev)
- int [virDomainDefCompatibleDevice](#) ([virDomainDefPtr](#) def, [virDomainDeviceDefPtr](#) dev)
- int [virDomainSaveXML](#) (const char \*configDir, [virDomainDefPtr](#) def, const char \*xml)
- int [virDomainSaveConfig](#) (const char \*configDir, [virDomainDefPtr](#) def)
- int [virDomainSaveStatus](#) (virCapsPtr caps, const char \*statusDir, [virDomainObjPtr](#) obj)
- static [virDomainObjPtr](#) [virDomainLoadConfig](#) (virCapsPtr caps, [virDomainObjListPtr](#) doms, const char \*configDir, const char \*autostartDir, const char \*name, unsigned int expectedVirtTypes, [virDomainLoadConfigNotify](#) notify, void \*opaque)
- static [virDomainObjPtr](#) [virDomainLoadStatus](#) (virCapsPtr caps, [virDomainObjListPtr](#) doms, const char \*statusDir, const char \*name, unsigned int expectedVirtTypes, [virDomainLoadConfigNotify](#) notify, void \*opaque)
- int [virDomainLoadAllConfigs](#) (virCapsPtr caps, [virDomainObjListPtr](#) doms, const char \*configDir, const char \*autostartDir, int liveStatus, unsigned int expectedVirtTypes, [virDomainLoadConfigNotify](#) notify, void \*opaque)
- int [virDomainDeleteConfig](#) (const char \*configDir, const char \*autostartDir, [virDomainObjPtr](#) dom)
- char \* [virDomainConfigFile](#) (const char \*dir, const char \*name)
- int [virDiskNameToBusDeviceIndex](#) (const [virDomainDiskDefPtr](#) disk, int \*busIdx, int \*devIdx)
- [virDomainFSDefPtr](#) [virDomainGetRootFilesystem](#) ([virDomainDefPtr](#) def)
- int [virDomainObjsDuplicate](#) ([virDomainObjListPtr](#) doms, [virDomainDefPtr](#) def, unsigned int check\_active)
- void [virDomainObjLock](#) ([virDomainObjPtr](#) obj)
- void [virDomainObjUnlock](#) ([virDomainObjPtr](#) obj)
- static void [virDomainObjListCountActive](#) (void \*payload, const void \*name ATTRIBUTE\_UNUSED, void \*data)
- static void [virDomainObjListCountInactive](#) (void \*payload, const void \*name ATTRIBUTE\_UNUSED, void \*data)

- int [virDomainObjListNumOfDomains](#) (virDomainObjListPtr doms, int active)
- static void [virDomainObjListCopyActiveIDs](#) (void \*payload, const void \*name ATTRIBUTE\_UNUSED, void \*opaque)
- int [virDomainObjListGetActiveIDs](#) (virDomainObjListPtr doms, int \*ids, int maxids)
- static void [virDomainObjListCopyInactiveNames](#) (void \*payload, const void \*name ATTRIBUTE\_UNUSED, void \*opaque)
- int [virDomainObjListGetInactiveNames](#) (virDomainObjListPtr doms, char \*\*const names, int maxnames)
- int [virDomainChrDefForeach](#) (virDomainDefPtr def, bool abortOnError, [virDomainChrDefIterator](#) iter, void \*opaque)
- int [virDomainSmartcardDefForeach](#) (virDomainDefPtr def, bool abortOnError, [virDomainSmartcardDefIterator](#) iter, void \*opaque)
- int [virDomainDiskDefForeachPath](#) (virDomainDiskDefPtr disk, bool allowProbing, bool ignoreOpenFailure, uid\_t uid, gid\_t gid, [virDomainDiskDefPathIterator](#) iter, void \*opaque)
- [virDomainDefPtr](#) [virDomainObjCopyPersistentDef](#) (virCapsPtr caps, [virDomainObjPtr](#) dom)
- [virDomainState](#) [virDomainObjGetState](#) ([virDomainObjPtr](#) dom, int \*reason)
- void [virDomainObjSetState](#) ([virDomainObjPtr](#) dom, [virDomainState](#) state, int reason)
- const char \* [virDomainStateReasonToString](#) ([virDomainState](#) state, int reason)
- int [virDomainStateReasonFromString](#) ([virDomainState](#) state, const char \*reason)
- int [virDomainNetGetActualType](#) ([virDomainNetDefPtr](#) iface)
- const char \* [virDomainNetGetActualBridgeName](#) ([virDomainNetDefPtr](#) iface)
- **POL New** const char \* [virDomainNetGetActualBridgeType](#) ([virDomainNetDefPtr](#) iface)
- const char \* [virDomainNetGetActualDirectDev](#) ([virDomainNetDefPtr](#) iface)
- int [virDomainNetGetActualDirectMode](#) ([virDomainNetDefPtr](#) iface)
- [virDomainHostdevDefPtr](#) [virDomainNetGetActualHostdev](#) ([virDomainNetDefPtr](#) iface)
- **POL Mod** [virNetDevVPortProfilePtr](#) [virDomainNetGetActualVirtPortProfile](#) ([virDomainNetDefPtr](#) iface)
- [virNetDevBandwidthPtr](#) [virDomainNetGetActualBandwidth](#) ([virDomainNetDefPtr](#) iface)
- [virNetDevVlanPtr](#) [virDomainNetGetActualVlan](#) ([virDomainNetDefPtr](#) iface)
- static [virDomainGraphicsListenDefPtr](#) [virDomainGraphicsGetListen](#) ([virDomainGraphicsDefPtr](#) def, size\_t ii, bool force0)
- int [virDomainGraphicsListenGetType](#) ([virDomainGraphicsDefPtr](#) def, size\_t ii)
- int [virDomainGraphicsListenSetType](#) ([virDomainGraphicsDefPtr](#) def, size\_t ii, int val)
- const char \* [virDomainGraphicsListenGetAddress](#) ([virDomainGraphicsDefPtr](#) def, size\_t ii)
- int [virDomainGraphicsListenSetAddress](#) ([virDomainGraphicsDefPtr](#) def, size\_t ii, const char \*address, int len, bool setType)
- const char \* [virDomainGraphicsListenGetNetwork](#) ([virDomainGraphicsDefPtr](#) def, size\_t ii)
- int [virDomainGraphicsListenSetNetwork](#) ([virDomainGraphicsDefPtr](#) def, size\_t ii, const char \*network, int len)
- [virDomainNetDefPtr](#) [virDomainNetFind](#) ([virDomainDefPtr](#) def, const char \*device)
- [virDomainDeviceDefPtr](#) [virDomainDeviceDefCopy](#) (virCapsPtr caps, const [virDomainDefPtr](#) def, [virDomainDeviceDefPtr](#) src)
- static void [virDomainListPopulate](#) (void \*payload, const void \*name ATTRIBUTE\_UNUSED, void \*opaque)
- int [virDomainList](#) ([virConnectPtr](#) conn, virHashTablePtr domobjs, [virDomainPtr](#) \*\*domains, unsigned int flags)
- [virSecurityLabelDefPtr](#) [virDomainDefGetSecurityLabelDef](#) ([virDomainDefPtr](#) def, const char \*model)
- [virSecurityDeviceLabelDefPtr](#) [virDomainDiskDefGetSecurityLabelDef](#) ([virDomainDiskDefPtr](#) def, const char \*model)
- [virSecurityDeviceLabelDefPtr](#) [virDomainChrDefGetSecurityLabelDef](#) ([virDomainChrDefPtr](#) def, const char \*model)
- [virSecurityLabelDefPtr](#) [virDomainDefAddSecurityLabelDef](#) ([virDomainDefPtr](#) def, const char \*model)

## 4.2.1 Macro Definition Documentation

### 4.2.1.1 #define DUMPXML\_FLAGS

Value:

```
(VIR_DOMAIN_XML_SECURE |
 VIR_DOMAIN_XML_INACTIVE |
 VIR_DOMAIN_XML_UPDATE_CPU)
```

```

4.2.1.2 #define MATCH( FLAG ) (data->flags & (FLAG))

4.2.1.3 #define NET_MODEL_CHARS "abcdefghijklmnopqrstuvwxyABCDEFGHIJKLMNOPQRSTUVWXYZ091234567890_"

4.2.1.4 #define VIR_DOMAIN_BLOCKED_LAST (VIR_DOMAIN_BLOCKED_UNKNOWN + 1)

4.2.1.5 #define VIR_DOMAIN_CRASHED_LAST (VIR_DOMAIN_CRASHED_UNKNOWN + 1)

4.2.1.6 #define VIR_DOMAIN_NOSTATE_LAST (VIR_DOMAIN_NOSTATE_UNKNOWN + 1)

4.2.1.7 #define VIR_DOMAIN_PAUSED_LAST (VIR_DOMAIN_PAUSED_SHUTTING_DOWN + 1)

4.2.1.8 #define VIR_DOMAIN_RUNNING_LAST (VIR_DOMAIN_RUNNING_SAVE_CANCELED + 1)

4.2.1.9 #define VIR_DOMAIN_SHUTDOWN_LAST (VIR_DOMAIN_SHUTDOWN_USER + 1)

4.2.1.10 #define VIR_DOMAIN_SHUTOFF_LAST (VIR_DOMAIN_SHUTOFF_FROM_SNAPSHOT + 1)

4.2.1.11 #define VIR_DOMAIN_XML_READ_FLAGS VIR_DOMAIN_XML_INACTIVE

4.2.1.12 #define VIR_DOMAIN_XML_WRITE_FLAGS VIR_DOMAIN_XML_SECURE

4.2.1.13 #define VIR_FROM_THIS VIR_FROM_DOMAIN

```

## 4.2.2 Enumeration Type Documentation

### 4.2.2.1 enum virDomainXMLInternalFlags

Enumerator

```

VIR_DOMAIN_XML_INTERNAL_STATUS
VIR_DOMAIN_XML_INTERNAL_ACTUAL_NET
VIR_DOMAIN_XML_INTERNAL_PCI_ORIG_STATES
VIR_DOMAIN_XML_INTERNAL_ALLOW_ROM
VIR_DOMAIN_XML_INTERNAL_ALLOW_BOOT

```

## 4.2.3 Function Documentation

```

4.2.3.1 static void ATTRIBUTE_NONNULL ( 1 ) [static]

4.2.3.2 static int ATTRIBUTE_NONNULL ( 2 ) [static]

4.2.3.3 verify ( VIR_DOMAIN_VIRT_LAST<= 32 )

4.2.3.4 verify ( ((VIR_DOMAIN_XML_INTERNAL_STATUS|VIR_DOMAIN_XML_INTERNAL_ACTU-
AL_NET|VIR_DOMAIN_XML_INTERNAL_PCI_ORIG_STATES)&DUMPXML_FLAGS) == 0
)

4.2.3.5 VIR_ENUM_IMPL ( virDomainTaint , VIR_DOMAIN_TAINT_LAST , "custom-argv" , "custom-monitor" ,
"high-privileges" , "shell-scripts" , "disk-probing" , "external-launch" , "host-cpu" )

4.2.3.6 VIR_ENUM_IMPL ( virDomainVirt , VIR_DOMAIN_VIRT_LAST , VIR_ENUM_IMPL( "qemu" , "kqemu" , "kvm" ,
VIR_ENUM_IMPL( "xen" , VIR_ENUM_IMPL( "xc" , VIR_ENUM_IMPL( "uml" , "openvz" , "test" , "vmware" , "hyperv" ,
VIR_ENUM_IMPL( "vbox" , "phyp" , "parallels" )

```

4.2.3.7 void virBlkioDeviceWeightArrayClear ( virBlkioDeviceWeightPtr *deviceWeights*, int *ndevices* )

4.2.3.8 int virDiskNameToBusDeviceIndex ( const virDomainDiskDefPtr *disk*, int \* *busidx*, int \* *devidx* )

4.2.3.9 **POL Mod** static int virDomainActualNetDefFormat ( virBufferPtr *buf*, virDomainActualNetDefPtr *def*, unsigned int *flags* ) [static]

virDomainActualNetDefFormat

#### Parameters

<i>buf</i>	pointer to virBuffer structure
<i>def</i>	pointer to virDomainActualNetDef structure
<i>flags</i>	flags

Display the Domain network configuration

POL modification:

- Display the bridgetype attribute in the 'source' block

4.2.3.10 **POL Mod** void virDomainActualNetDefFree ( virDomainActualNetDefPtr *def* )

virDomainActualNetDefFree

#### Parameters

<i>def</i>	pointer to virDomainActualNetDef structure
------------	--

Free a virDomainActualNetDef structure.

POL modification:

- Free also *def*->data.bridge.brtype

4.2.3.11 static int virDomainActualNetDefParseXML ( xmlNodePtr *node*, xmlXPathContextPtr *ctxt*, virDomainNetDefPtr *parent*, virDomainActualNetDefPtr \* *def*, unsigned int *flags* ) [static]

4.2.3.12 virDomainObjPtr virDomainAssignDef ( virCapsPtr *caps*, virDomainObjListPtr *doms*, const virDomainDefPtr *def*, bool *live* )

4.2.3.13 static int virDomainBlkioDeviceWeightParseXML ( xmlNodePtr *root*, virBlkioDeviceWeightPtr *dw* ) [static]

virDomainBlkioDeviceWeightParseXML

this function parses a XML node:

```
<device> <path>/fully/qualified/device/path</path> <weight>weight</weight> </device>
```

and fills a virBlkioDeviceWeight struct.

4.2.3.14 static bool virDomainChannelDefCheckABIStability ( virDomainChrDefPtr *src*, virDomainChrDefPtr *dst* ) [static]

4.2.3.15 static int virDomainChrDefaultTargetType ( virCapsPtr *caps*, virDomainDefPtr *def*, int *devtype* ) [static]

- 4.2.3.16 `int virDomainChrDefForeach ( virDomainDefPtr def, bool abortOnError, virDomainChrDefIterator iter, void * opaque )`
- 4.2.3.17 `static int virDomainChrDefFormat ( virBufferPtr buf, virDomainChrDefPtr def, unsigned int flags ) [static]`
- 4.2.3.18 `void virDomainChrDefFree ( virDomainChrDefPtr def )`
- 4.2.3.19 `virSecurityDeviceLabelDefPtr virDomainChrDefGetSecurityLabelDef ( virDomainChrDefPtr def, const char * model )`
- 4.2.3.20 `virDomainChrDefPtr virDomainChrDefNew ( void )`
- 4.2.3.21 `static int virDomainChrDefParseTargetXML ( virCapsPtr caps, virDomainDefPtr vmdef, virDomainChrDefPtr def, xmlNodePtr cur ) [static]`
- 4.2.3.22 `static virDomainChrDefPtr virDomainChrDefParseXML ( virCapsPtr caps, virDomainDefPtr vmdef, xmlXPathContextPtr ctxt, xmlNodePtr node, virSecurityLabelDefPtr * vmSeclabels, int nvmSeclabels, unsigned int flags ) [static]`
- 4.2.3.23 `int virDomainChrSourceDefCopy ( virDomainChrSourceDefPtr dest, virDomainChrSourceDefPtr src )`
- 4.2.3.24 `static int virDomainChrSourceDefFormat ( virBufferPtr buf, virDomainChrSourceDefPtr def, bool tty_compat, unsigned int flags ) [static]`
- 4.2.3.25 `void virDomainChrSourceDefFree ( virDomainChrSourceDefPtr def )`
- 4.2.3.26 `static bool virDomainChrSourceDefIsEqual ( const virDomainChrSourceDef * src, const virDomainChrSourceDef * tgt ) [static]`
- 4.2.3.27 `static int virDomainChrSourceDefParseXML ( virDomainChrSourceDefPtr def, xmlNodePtr cur, unsigned int flags, virDomainChrDefPtr chr_def, xmlXPathContextPtr ctxt, virSecurityLabelDefPtr * vmSeclabels, int nvmSeclabels ) [static]`
- 4.2.3.28 `static int virDomainChrTargetTypeFromString ( virCapsPtr caps, virDomainDefPtr def, int devtype, const char * targetType ) [static]`
- 4.2.3.29 `static const char* virDomainChrTargetTypeToString ( int deviceType, int targetType ) [static]`
- 4.2.3.30 `static void virDomainClockDefClear ( virDomainClockDefPtr def ) [static]`
- 4.2.3.31 `char* virDomainConfigFile ( const char * dir, const char * name )`
- 4.2.3.32 `static bool virDomainConsoleDefCheckABIStability ( virDomainChrDefPtr src, virDomainChrDefPtr dst ) [static]`
- 4.2.3.33 `static bool virDomainControllerDefCheckABIStability ( virDomainControllerDefPtr src, virDomainControllerDefPtr dst ) [static]`
- 4.2.3.34 `static int virDomainControllerDefFormat ( virBufferPtr buf, virDomainControllerDefPtr def, unsigned int flags ) [static]`
- 4.2.3.35 `void virDomainControllerDefFree ( virDomainControllerDefPtr def )`
- 4.2.3.36 `static virDomainControllerDefPtr virDomainControllerDefParseXML ( xmlNodePtr node, unsigned int flags ) [static]`

- 4.2.3.37 `int virDomainControllerFind ( virDomainDefPtr def, int type, int idx )`
- 4.2.3.38 `int virDomainControllerInsert ( virDomainDefPtr def, virDomainControllerDefPtr controller )`
- 4.2.3.39 `void virDomainControllerInsertPreAlloced ( virDomainDefPtr def, virDomainControllerDefPtr controller )`
- 4.2.3.40 `static int virDomainControllerModelTypeFromString ( const virDomainControllerDefPtr def, const char * model )`  
[static]
- 4.2.3.41 `static const char* virDomainControllerModelTypeToString ( virDomainControllerDefPtr def, int model )`  
[static]
- 4.2.3.42 `virDomainControllerDefPtr virDomainControllerRemove ( virDomainDefPtr def, size_t i )`
- 4.2.3.43 `static int virDomainDefAddDiskControllersForType ( virDomainDefPtr def, int controllerType, int diskBus )`  
[static]
- 4.2.3.44 `int virDomainDefAddImplicitControllers ( virDomainDefPtr def )`
- 4.2.3.45 `virSecurityLabelDefPtr virDomainDefAddSecurityLabelDef ( virDomainDefPtr def, const char * model )`
- 4.2.3.46 `bool virDomainDefCheckABIStability ( virDomainDefPtr src, virDomainDefPtr dst )`
- 4.2.3.47 `void virDomainDefClearDeviceAliases ( virDomainDefPtr def )`
- 4.2.3.48 `void virDomainDefClearPCIAddresses ( virDomainDefPtr def )`
- 4.2.3.49 `int virDomainDefCompatibleDevice ( virDomainDefPtr def, virDomainDeviceDefPtr dev )`
- 4.2.3.50 `static char* virDomainDefDefaultEmulator ( virDomainDefPtr def, virCapsPtr caps )` [static]
- 4.2.3.51 `char* virDomainDefFormat ( virDomainDefPtr def, unsigned int flags )`
- 4.2.3.52 `int virDomainDefFormatInternal ( virDomainDefPtr def, unsigned int flags, virBufferPtr buf )`
- 4.2.3.53 `void virDomainDefFree ( virDomainDefPtr def )`
- 4.2.3.54 `virSecurityLabelDefPtr virDomainDefGetSecurityLabelDef ( virDomainDefPtr def, const char * model )`
- 4.2.3.55 `static bool virDomainDefHasUSB ( virDomainDefPtr def )` [static]
- 4.2.3.56 `static int virDomainDefMaybeAddController ( virDomainDefPtr def, int type, int idx )` [static]
- 4.2.3.57 `static int virDomainDefMaybeAddSmartcardController ( virDomainDefPtr def )` [static]
- 4.2.3.58 `static int virDomainDefMaybeAddVirtioSerialController ( virDomainDefPtr def )` [static]
- 4.2.3.59 `static virDomainDefPtr virDomainDefParse ( const char * xmlStr, const char * filename, virCapsPtr caps, unsigned int expectedVirtTypes, unsigned int flags )` [static]
- 4.2.3.60 `static int virDomainDefParseBootXML ( xmlXPathContextPtr ctxt, virDomainDefPtr def, unsigned long * bootCount )` [static]
- 4.2.3.61 `virDomainDefPtr virDomainDefParseFile ( virCapsPtr caps, const char * filename, unsigned int expectedVirtTypes, unsigned int flags )`

- 4.2.3.62 **virDomainDefPtr** virDomainDefParseNode ( *virCapsPtr caps*, *xmlDocPtr xml*, *xmlNodePtr root*, unsigned int *expectedVirtTypes*, unsigned int *flags* )
- 4.2.3.63 **virDomainDefPtr** virDomainDefParseString ( *virCapsPtr caps*, const char \* *xmlStr*, unsigned int *expectedVirtTypes*, unsigned int *flags* )
- 4.2.3.64 **static virDomainDefPtr** virDomainDefParseXML ( *virCapsPtr caps*, *xmlDocPtr xml*, *xmlNodePtr root*, *xmlXPathContextPtr ctxt*, unsigned int *expectedVirtTypes*, unsigned int *flags* ) [static]
- 4.2.3.65 **int** virDomainDeleteConfig ( const char \* *configDir*, const char \* *autostartDir*, **virDomainObjPtr** *dom* )
- 4.2.3.66 **int** virDomainDeviceAddressIsValid ( **virDomainDeviceInfoPtr** *info*, int *type* )
- 4.2.3.67 **static int** virDomainDeviceBootParseXML ( *xmlNodePtr node*, int \* *bootIndex*, **virBitmapPtr** *bootMap* ) [static]
- 4.2.3.68 **static int** virDomainDeviceCcidAddressParseXML ( *xmlNodePtr node*, **virDomainDeviceCcidAddressPtr** *addr* ) [static]
- 4.2.3.69 **virDomainDeviceDefPtr** virDomainDeviceDefCopy ( *virCapsPtr caps*, const **virDomainDefPtr** *def*, **virDomainDeviceDefPtr** *src* )
- 4.2.3.70 **void** virDomainDeviceDefFree ( **virDomainDeviceDefPtr** *def* )
- 4.2.3.71 **virDomainDeviceDefPtr** virDomainDeviceDefParse ( *virCapsPtr caps*, **virDomainDefPtr** *def*, const char \* *xmlStr*, unsigned int *flags* )
- 4.2.3.72 **static int** virDomainDeviceDriveAddressParseXML ( *xmlNodePtr node*, **virDomainDeviceDriveAddressPtr** *addr* ) [static]
- 4.2.3.73 **static bool** virDomainDeviceInfoCheckABIStability ( **virDomainDeviceInfoPtr** *src*, **virDomainDeviceInfoPtr** *dst* ) [static]
- 4.2.3.74 **void** virDomainDeviceInfoClear ( **virDomainDeviceInfoPtr** *info* )
- 4.2.3.75 **static int** virDomainDeviceInfoClearAlias ( **virDomainDefPtr** *def* *ATTRIBUTE\_UNUSED*, **virDomainDeviceDefPtr** *device* *ATTRIBUTE\_UNUSED*, **virDomainDeviceInfoPtr** *info*, void \**opaque* *ATTRIBUTE\_UNUSED* ) [static]
- 4.2.3.76 **static int** virDomainDeviceInfoClearPCIAddress ( **virDomainDefPtr** *def* *ATTRIBUTE\_UNUSED*, **virDomainDeviceDefPtr** *device* *ATTRIBUTE\_UNUSED*, **virDomainDeviceInfoPtr** *info*, void \**opaque* *ATTRIBUTE\_UNUSED* ) [static]
- 4.2.3.77 **static void** virDomainDeviceInfoFree ( **virDomainDeviceInfoPtr** *info* ) [static]
- 4.2.3.78 **static bool** virDomainDeviceInfosSet ( **virDomainDeviceInfoPtr** *info*, unsigned int *flags* ) [static]
- 4.2.3.79 **int** virDomainDeviceInfoIterate ( **virDomainDefPtr** *def*, **virDomainDeviceInfoCallback** *cb*, void \* *opaque* )
- 4.2.3.80 **static int** virDomainDeviceInfoParseXML ( *xmlNodePtr node*, **virBitmapPtr** *bootMap*, **virDomainDeviceInfoPtr** *info*, unsigned int *flags* ) [static]
- 4.2.3.81 **static bool** virDomainDevicesUSB ( **virDomainDeviceDefPtr** *dev* ) [static]
- 4.2.3.82 **static int** virDomainDeviceSpaprVioAddressParseXML ( *xmlNodePtr node*, **virDomainDeviceSpaprVioAddressPtr** *addr* ) [static]



- 4.2.3.83 `static int virDomainDeviceUSBAddressParseXML ( xmlDocPtr node, virDomainDeviceUSBAddressPtr addr )`  
[static]
- 4.2.3.84 `static int virDomainDeviceUSBMasterParseXML ( xmlDocPtr node, virDomainDeviceUSBMasterPtr master )`  
[static]
- 4.2.3.85 `static int virDomainDeviceVirtioSerialAddressParseXML ( xmlDocPtr node, virDomainDeviceVirtioSerial-  
AddressPtr addr )` [static]
- 4.2.3.86 `static void virDomainDiskBlockDefFormat ( virBufferPtr buf, virDomainDiskDefPtr def )` [static]
- 4.2.3.87 `int virDomainDiskDefAssignAddress ( virCapsPtr caps, virDomainDiskDefPtr def )`
- 4.2.3.88 `static bool virDomainDiskDefCheckABIStability ( virDomainDiskDefPtr src, virDomainDiskDefPtr dst )`  
[static]
- 4.2.3.89 `int virDomainDiskDefForeachPath ( virDomainDiskDefPtr disk, bool allowProbing, bool ignoreOpenFailure, uid_t  
uid, gid_t gid, virDomainDiskDefPathIterator iter, void * opaque )`
- 4.2.3.90 `static int virDomainDiskDefFormat ( virBufferPtr buf, virDomainDiskDefPtr def, unsigned int flags )`  
[static]
- 4.2.3.91 `void virDomainDiskDefFree ( virDomainDiskDefPtr def )`
- 4.2.3.92 `virSecurityDeviceLabelDefPtr virDomainDiskDefGetSecurityLabelDef ( virDomainDiskDefPtr def, const char  
* model )`
- 4.2.3.93 `static virDomainDiskDefPtr virDomainDiskDefParseXML ( virCapsPtr caps, xmlDocPtr node, xmlDocPtr ctx, virBitmapPtr bootMap, virSecurityDeviceLabelDefPtr * vmSeclabels, int nvmSeclabels, unsigned int flags )`  
[static]
- 4.2.3.94 `int virDomainDiskFindControllerModel ( virDomainDefPtr def, virDomainDiskDefPtr disk, int controllerType )`
- 4.2.3.95 `static void virDomainDiskGeometryDefFormat ( virBufferPtr buf, virDomainDiskDefPtr def )` [static]
- 4.2.3.96 `void virDomainDiskHostDefFree ( virDomainDiskHostDefPtr def )`
- 4.2.3.97 `int virDomainDiskIndexByName ( virDomainDefPtr def, const char * name, bool allow_ambiguous )`
- 4.2.3.98 `int virDomainDiskInsert ( virDomainDefPtr def, virDomainDiskDefPtr disk )`
- 4.2.3.99 `void virDomainDiskInsertPreAlloced ( virDomainDefPtr def, virDomainDiskDefPtr disk )`
- 4.2.3.100 `const char* virDomainDiskPathByName ( virDomainDefPtr def, const char * name )`
- 4.2.3.101 `virDomainDiskDefPtr virDomainDiskRemove ( virDomainDefPtr def, size_t i )`
- 4.2.3.102 `virDomainDiskDefPtr virDomainDiskRemoveByName ( virDomainDefPtr def, const char * name )`
- 4.2.3.103 `int virDomainEmulatorPinAdd ( virDomainDefPtr def, unsigned char * cpumap, int maplen )`
- 4.2.3.104 `int virDomainEmulatorPinDel ( virDomainDefPtr def )`
- 4.2.3.105 `virDomainObjPtr virDomainFindByID ( const virDomainObjListPtr doms, int id )`
- 4.2.3.106 `virDomainObjPtr virDomainFindByName ( const virDomainObjListPtr doms, const char * name )`

- 4.2.3.107 **virDomainObjPtr** virDomainFindByUUID ( *const virDomainObjListPtr doms*, *const unsigned char \* uuid* )
- 4.2.3.108 **static bool** virDomainFsDefCheckABIStability ( *virDomainFSDefPtr src*, *virDomainFSDefPtr dst* )  
[static]
- 4.2.3.109 **static int** virDomainFSDefFormat ( *virBufferPtr buf*, *virDomainFSDefPtr def*, *unsigned int flags* ) [static]
- 4.2.3.110 **void** virDomainFSDefFree ( *virDomainFSDefPtr def* )
- 4.2.3.111 **static virDomainFSDefPtr** virDomainFSDefParseXML ( *xmlNodePtr node*, *xmlXPathContextPtr ctxt*, *unsigned int flags* ) [static]
- 4.2.3.112 **int** virDomainFSIndexByName ( *virDomainDefPtr def*, *const char \* name* )
- 4.2.3.113 **virDomainFSDefPtr** virDomainGetRootFilesystem ( *virDomainDefPtr def* )
- 4.2.3.114 **static void** virDomainGraphicsAuthDefClear ( *virDomainGraphicsAuthDefPtr def* ) [static]
- 4.2.3.115 **static void** virDomainGraphicsAuthDefFormatAttr ( *virBufferPtr buf*, *virDomainGraphicsAuthDefPtr def*, *unsigned int flags* ) [static]
- 4.2.3.116 **static int** virDomainGraphicsAuthDefParseXML ( *xmlNodePtr node*, *virDomainGraphicsAuthDefPtr def*, *int type* ) [static]
- 4.2.3.117 **static int** virDomainGraphicsDefFormat ( *virBufferPtr buf*, *virDomainGraphicsDefPtr def*, *unsigned int flags* )  
[static]
- 4.2.3.118 **void** virDomainGraphicsDefFree ( *virDomainGraphicsDefPtr def* )
- 4.2.3.119 **static virDomainGraphicsDefPtr** virDomainGraphicsDefParseXML ( *xmlNodePtr node*, *xmlXPathContextPtr ctxt*, *unsigned int flags* ) [static]
- 4.2.3.120 **static virDomainGraphicsListenDefPtr** virDomainGraphicsGetListen ( *virDomainGraphicsDefPtr def*, *size\_t ii*, *bool force0* ) [static]
- 4.2.3.121 **static void** virDomainGraphicsListenDefClear ( *virDomainGraphicsListenDefPtr def* ) [static]
- 4.2.3.122 **static void** virDomainGraphicsListenDefFormat ( *virBufferPtr buf*, *virDomainGraphicsListenDefPtr def*, *unsigned int flags* ) [static]
- 4.2.3.123 **static int** virDomainGraphicsListenDefParseXML ( *virDomainGraphicsListenDefPtr def*, *xmlNodePtr node*, *unsigned int flags* ) [static]
- 4.2.3.124 **const char\*** virDomainGraphicsListenGetAddress ( *virDomainGraphicsDefPtr def*, *size\_t ii* )
- 4.2.3.125 **const char\*** virDomainGraphicsListenGetNetwork ( *virDomainGraphicsDefPtr def*, *size\_t ii* )
- 4.2.3.126 **int** virDomainGraphicsListenGetType ( *virDomainGraphicsDefPtr def*, *size\_t ii* )
- 4.2.3.127 **int** virDomainGraphicsListenSetAddress ( *virDomainGraphicsDefPtr def*, *size\_t ii*, *const char \* address*, *int len*, *bool setType* )
- 4.2.3.128 **int** virDomainGraphicsListenSetNetwork ( *virDomainGraphicsDefPtr def*, *size\_t ii*, *const char \* network*, *int len* )
- 4.2.3.129 **int** virDomainGraphicsListenSetType ( *virDomainGraphicsDefPtr def*, *size\_t ii*, *int val* )

- 4.2.3.130 `virDomainHostdevDefPtr virDomainHostdevDefAlloc ( void )`
- 4.2.3.131 `static bool virDomainHostdevDefCheckABIStability ( virDomainHostdevDefPtr src, virDomainHostdevDefPtr dst ) [static]`
- 4.2.3.132 `void virDomainHostdevDefClear ( virDomainHostdevDefPtr def )`
- 4.2.3.133 `static int virDomainHostdevDefFormat ( virBufferPtr buf, virDomainHostdevDefPtr def, unsigned int flags ) [static]`
- 4.2.3.134 `void virDomainHostdevDefFree ( virDomainHostdevDefPtr def )`
- 4.2.3.135 `static virDomainHostdevDefPtr virDomainHostdevDefParseXML ( const xmlNodePtr node, xmlXPathContextPtr ctxt, virBitmapPtr bootMap, unsigned int flags ) [static]`
- 4.2.3.136 `int virDomainHostdevFind ( virDomainDefPtr def, virDomainHostdevDefPtr match, virDomainHostdevDefPtr * found )`
- 4.2.3.137 `int virDomainHostdevInsert ( virDomainDefPtr def, virDomainHostdevDefPtr hostdev )`
- 4.2.3.138 `static int virDomainHostdevPartsParse ( xmlNodePtr node, xmlXPathContextPtr ctxt, const char * mode, const char * type, virDomainHostdevDefPtr def, unsigned int flags ) [static]`
- 4.2.3.139 `virDomainHostdevDefPtr virDomainHostdevRemove ( virDomainDefPtr def, size_t i )`
- 4.2.3.140 `static int virDomainHostdevSourceFormat ( virBufferPtr buf, virDomainHostdevDefPtr def, unsigned int flags, bool includeTypeInAddr ) [static]`
- 4.2.3.141 `static int virDomainHostdevSubsysPciDefParseXML ( const xmlNodePtr node, virDomainHostdevDefPtr def, unsigned int flags ) [static]`
- 4.2.3.142 `static int virDomainHostdevSubsysPciOrigStatesDefParseXML ( const xmlNodePtr node, virDomainHostdevOrigStatesPtr def ) [static]`
- 4.2.3.143 `static int virDomainHostdevSubsysUsbDefParseXML ( const xmlNodePtr node, virDomainHostdevDefPtr def ) [static]`
- 4.2.3.144 `static bool virDomainHubDefCheckABIStability ( virDomainHubDefPtr src, virDomainHubDefPtr dst ) [static]`
- 4.2.3.145 `static int virDomainHubDefFormat ( virBufferPtr buf, virDomainHubDefPtr def, unsigned int flags ) [static]`
- 4.2.3.146 `void virDomainHubDefFree ( virDomainHubDefPtr def )`
- 4.2.3.147 `static virDomainHubDefPtr virDomainHubDefParseXML ( xmlNodePtr node, unsigned int flags ) [static]`
- 4.2.3.148 `static bool virDomainInputDefCheckABIStability ( virDomainInputDefPtr src, virDomainInputDefPtr dst ) [static]`
- 4.2.3.149 `static int virDomainInputDefFormat ( virBufferPtr buf, virDomainInputDefPtr def, unsigned int flags ) [static]`
- 4.2.3.150 `void virDomainInputDefFree ( virDomainInputDefPtr def )`

- 4.2.3.151 `static virDomainInputDefPtr virDomainInputDefParseXML ( const char * ostype, xmlNodePtr node, unsigned int flags ) [static]`
- 4.2.3.152 `static int virDomainLeaseDefFormat ( virBufferPtr buf, virDomainLeaseDefPtr def ) [static]`
- 4.2.3.153 `void virDomainLeaseDefFree ( virDomainLeaseDefPtr def )`
- 4.2.3.154 `static virDomainLeaseDefPtr virDomainLeaseDefParseXML ( xmlNodePtr node ) [static]`
- 4.2.3.155 `int virDomainLeaseIndex ( virDomainDefPtr def, virDomainLeaseDefPtr lease )`
- 4.2.3.156 `int virDomainLeaseInsert ( virDomainDefPtr def, virDomainLeaseDefPtr lease )`
- 4.2.3.157 `int virDomainLeaseInsertPreAlloc ( virDomainDefPtr def )`
- 4.2.3.158 `void virDomainLeaseInsertPreAlloced ( virDomainDefPtr def, virDomainLeaseDefPtr lease )`
- 4.2.3.159 `virDomainLeaseDefPtr virDomainLeaseRemove ( virDomainDefPtr def, virDomainLeaseDefPtr lease )`
- 4.2.3.160 `virDomainLeaseDefPtr virDomainLeaseRemoveAt ( virDomainDefPtr def, size_t i )`
- 4.2.3.161 `static int virDomainLifecycleDefFormat ( virBufferPtr buf, int type, const char * name, virLifecycleToStringFunc convFunc ) [static]`
- 4.2.3.162 `static int virDomainLifecycleParseXML ( xmlXPathContextPtr ctxt, const char * xpath, int * val, int defaultVal, virLifecycleFromStringFunc convFunc ) [static]`
- 4.2.3.163 `int virDomainList ( virConnectPtr conn, virHashTablePtr domobjs, virDomainPtr ** domains, unsigned int flags )`
- 4.2.3.164 `static void virDomainListPopulate ( void * payload, const void *name ATTRIBUTE_UNUSED, void * opaque ) [static]`
- 4.2.3.165 `int virDomainLiveConfigHelperMethod ( virCapsPtr caps, virDomainObjPtr dom, unsigned int * flags, virDomainDefPtr * persistentDef )`
- 4.2.3.166 `int virDomainLoadAllConfigs ( virCapsPtr caps, virDomainObjListPtr doms, const char * configDir, const char * autostartDir, int liveStatus, unsigned int expectedVirtTypes, virDomainLoadConfigNotify notify, void * opaque )`
- 4.2.3.167 `static virDomainObjPtr virDomainLoadConfig ( virCapsPtr caps, virDomainObjListPtr doms, const char * configDir, const char * autostartDir, const char * name, unsigned int expectedVirtTypes, virDomainLoadConfigNotify notify, void * opaque ) [static]`
- 4.2.3.168 `static virDomainObjPtr virDomainLoadStatus ( virCapsPtr caps, virDomainObjListPtr doms, const char * statusDir, const char * name, unsigned int expectedVirtTypes, virDomainLoadConfigNotify notify, void * opaque ) [static]`
- 4.2.3.169 `static bool virDomainMemballoonDefCheckABIStability ( virDomainMemballoonDefPtr src, virDomainMemballoonDefPtr dst ) [static]`
- 4.2.3.170 `static int virDomainMemballoonDefFormat ( virBufferPtr buf, virDomainMemballoonDefPtr def, unsigned int flags ) [static]`
- 4.2.3.171 `void virDomainMemballoonDefFree ( virDomainMemballoonDefPtr def )`

4.2.3.172 **static virDomainMemballoonDefPtr** virDomainMemballoonDefParseXML ( **const xmlNodePtr** *node*, **unsigned int** *flags* ) [static]

4.2.3.173 **static bool** virDomainNetDefCheckABIStability ( **virDomainNetDefPtr** *src*, **virDomainNetDefPtr** *dst* ) [static]

4.2.3.174 **static int** virDomainNetDefFormat ( **virBufferPtr** *buf*, **virDomainNetDefPtr** *def*, **unsigned int** *flags* ) [static]

virDomainNetDefFormat

#### Parameters

<i>buf</i>	pointer to virBuffer structure
<i>def</i>	pointer to virDomainNetDef structure
<i>flags</i>	flags

Display the Domain network configuration

POL modification:

- Display the bridgetype attribute in the 'source' block

4.2.3.175 **POL Mod** **void** virDomainNetDefFree ( **virDomainNetDefPtr** *def* )

virDomainNetDefFree

#### Parameters

<i>def</i>	pointer to virDomainNetDef structure
------------	--------------------------------------

Free a virDomainNetDef structure.

POL modification:

- Free also def->data.bridge.brtype

4.2.3.176 **POL Mod** **static virDomainNetDefPtr** virDomainNetDefParseXML ( **virCapsPtr** *caps*, **xmlNodePtr** *node*, **xmlXPathContextPtr** *ctxt*, **virBitmapPtr** *bootMap*, **unsigned int** *flags* ) [static]

virDomainNetDefParseXML Parse the XML definition for a network interface

#### Parameters

<i>node</i>	XML nodeset to parse for net definition
-------------	---

#### Returns

0 on success, -1 on failure

POL modification:

- Parse the bridge type from bridgetype attribute of the 'source' block

4.2.3.177 **virDomainNetDefPtr** virDomainNetFind ( **virDomainDefPtr** *def*, **const char \*** *device* )

4.2.3.178 `virNetDevBandwidthPtr virDomainNetGetActualBandwidth ( virDomainNetDefPtr iface )`

4.2.3.179 `const char* virDomainNetGetActualBridgeName ( virDomainNetDefPtr iface )`

4.2.3.180 **POL New** `const char* virDomainNetGetActualBridgeType ( virDomainNetDefPtr iface )`

`virDomainNetGetActualBridgeType`

#### Parameters

<i>iface</i>	pointer to virDomainNetDef structure
--------------	--------------------------------------

Return the bridge type from a virDomainNetDef structure

4.2.3.181 `const char* virDomainNetGetActualDirectDev ( virDomainNetDefPtr iface )`

4.2.3.182 `int virDomainNetGetActualDirectMode ( virDomainNetDefPtr iface )`

4.2.3.183 `virDomainHostdevDefPtr virDomainNetGetActualHostdev ( virDomainNetDefPtr iface )`

4.2.3.184 `int virDomainNetGetActualType ( virDomainNetDefPtr iface )`

4.2.3.185 **POL Mod** `virNetDevVPortProfilePtr virDomainNetGetActualVirtPortProfile ( virDomainNetDefPtr iface )`

`virDomainNetGetActualVirtPortProfile`

#### Parameters

<i>def</i>	pointer to virDomainNetDef structure
------------	--------------------------------------

Return the actual virtual port profile from a virDomainNetDef structure

POL modification:

- Added support for interfaces with type 'network'

4.2.3.186 `virNetDevVlanPtr virDomainNetGetActualVlan ( virDomainNetDefPtr iface )`

4.2.3.187 `int virDomainNetIndexByMac ( virDomainDefPtr def, const virMacAddrPtr mac )`

4.2.3.188 `int virDomainNetInsert ( virDomainDefPtr def, virDomainNetDefPtr net )`

4.2.3.189 `virDomainNetDefPtr virDomainNetRemove ( virDomainDefPtr def, size_t i )`

4.2.3.190 `virDomainNetDefPtr virDomainNetRemoveByMac ( virDomainDefPtr def, const virMacAddrPtr mac )`

4.2.3.191 `void virDomainObjAssignDef ( virDomainObjPtr domain, const virDomainDefPtr def, bool live )`

4.2.3.192 `virDomainDefPtr virDomainObjCopyPersistentDef ( virCapsPtr caps, virDomainObjPtr dom )`

4.2.3.193 `static void virDomainObjDispose ( void * obj )` [static]

4.2.3.194 `static char* virDomainObjFormat ( virCapsPtr caps, virDomainObjPtr obj, unsigned int flags )` [static]

4.2.3.195 `virDomainDefPtr virDomainObjGetPersistentDef ( virCapsPtr caps, virDomainObjPtr domain )`

- 4.2.3.196 `virDomainState virDomainObjGetState ( virDomainObjPtr dom, int * reason )`
- 4.2.3.197 `int virDomainObjsDuplicate ( virDomainObjListPtr doms, virDomainDefPtr def, unsigned int check_active )`
- 4.2.3.198 `static void virDomainObjListCopyActiveIDs ( void * payload, const void *name ATTRIBUTE_UNUSED, void * opaque ) [static]`
- 4.2.3.199 `static void virDomainObjListCopyInactiveNames ( void * payload, const void *name ATTRIBUTE_UNUSED, void * opaque ) [static]`
- 4.2.3.200 `static void virDomainObjListCountActive ( void * payload, const void *name ATTRIBUTE_UNUSED, void * data ) [static]`
- 4.2.3.201 `static void virDomainObjListCountInactive ( void * payload, const void *name ATTRIBUTE_UNUSED, void * data ) [static]`
- 4.2.3.202 `static void virDomainObjListDataFree ( void * payload, const void *name ATTRIBUTE_UNUSED ) [static]`
- 4.2.3.203 `void virDomainObjListDeinit ( virDomainObjListPtr doms )`
- 4.2.3.204 `int virDomainObjListGetActiveIDs ( virDomainObjListPtr doms, int * ids, int maxids )`
- 4.2.3.205 `int virDomainObjListGetInactiveNames ( virDomainObjListPtr doms, char **const names, int maxnames )`
- 4.2.3.206 `int virDomainObjListInit ( virDomainObjListPtr doms )`
- 4.2.3.207 `int virDomainObjListNumOfDomains ( virDomainObjListPtr doms, int active )`
- 4.2.3.208 `static int virDomainObjListSearchID ( const void * payload, const void *name ATTRIBUTE_UNUSED, const void * data ) [static]`
- 4.2.3.209 `static int virDomainObjListSearchName ( const void * payload, const void *name ATTRIBUTE_UNUSED, const void * data ) [static]`
- 4.2.3.210 `void virDomainObjLock ( virDomainObjPtr obj )`
- 4.2.3.211 `virDomainObjPtr virDomainObjNew ( virCapsPtr caps )`
- 4.2.3.212 `static virDomainObjPtr virDomainObjParseFile ( virCapsPtr caps, const char * filename, unsigned int expectedVirtTypes, unsigned int flags ) [static]`
- 4.2.3.213 `static virDomainObjPtr virDomainObjParseNode ( virCapsPtr caps, xmlDocPtr xml, xmlNodePtr root, unsigned int expectedVirtTypes, unsigned int flags ) [static]`
- 4.2.3.214 `static virDomainObjPtr virDomainObjParseXML ( virCapsPtr caps, xmlDocPtr xml, xmlXPathContextPtr ctxt, unsigned int expectedVirtTypes, unsigned int flags ) [static]`
- 4.2.3.215 `int virDomainObjSetDefTransient ( virCapsPtr caps, virDomainObjPtr domain, bool live )`
- 4.2.3.216 `void virDomainObjSetState ( virDomainObjPtr dom, virDomainState state, int reason )`
- 4.2.3.217 `bool virDomainObjTaint ( virDomainObjPtr obj, enum virDomainTaintFlags taint )`
- 4.2.3.218 `void virDomainObjUnlock ( virDomainObjPtr obj )`

- 4.2.3.219 `static bool virDomainParallelDefCheckABIStability ( virDomainChrDefPtr src, virDomainChrDefPtr dst )`  
[static]
- 4.2.3.220 `static int virDomainParseLegacyDeviceAddress ( char * devaddr, virDevicePCIAddressPtr pci )` [static]
- 4.2.3.221 `static int virDomainParseMemory ( const char * xpath, xmlXPathContextPtr ctxt, unsigned long long * mem, bool required )` [static]
- 4.2.3.222 `static int virDomainParseScaledValue ( const char * xpath, xmlXPathContextPtr ctxt, unsigned long long * val, unsigned long long scale, unsigned long long max, bool required )` [static]
- 4.2.3.223 `static int virDomainPMStateParseXML ( xmlXPathContextPtr ctxt, const char * xpath, int * val )` [static]
- 4.2.3.224 `static int virDomainRedirdevDefFormat ( virBufferPtr buf, virDomainRedirdevDefPtr def, unsigned int flags )`  
[static]
- 4.2.3.225 `void virDomainRedirdevDefFree ( virDomainRedirdevDefPtr def )`
- 4.2.3.226 `static virDomainRedirdevDefPtr virDomainRedirdevDefParseXML ( const xmlNodePtr node, virBitmapPtr bootMap, unsigned int flags )` [static]
- 4.2.3.227 `static bool virDomainRedirFilterDefCheckABIStability ( virDomainRedirFilterDefPtr src, virDomainRedirFilterDefPtr dst )` [static]
- 4.2.3.228 `static int virDomainRedirFilterDefFormat ( virBufferPtr buf, virDomainRedirFilterDefPtr filter )` [static]
- 4.2.3.229 `void virDomainRedirFilterDefFree ( virDomainRedirFilterDefPtr def )`
- 4.2.3.230 `static virDomainRedirFilterDefPtr virDomainRedirFilterDefParseXML ( const xmlNodePtr node, xmlXPathContextPtr ctxt )` [static]
- 4.2.3.231 `static virDomainRedirFilterUsbDevDefPtr virDomainRedirFilterUsbDevDefParseXML ( const xmlNodePtr node )` [static]
- 4.2.3.232 `static int virDomainRedirFilterUsbVersionHelper ( const char * version, virDomainRedirFilterUsbDevDefPtr def )` [static]
- 4.2.3.233 `void virDomainRemoveInactive ( virDomainObjListPtr doms, virDomainObjPtr dom )`
- 4.2.3.234 `int virDomainSaveConfig ( const char * configDir, virDomainDefPtr def )`
- 4.2.3.235 `int virDomainSaveStatus ( virCapsPtr caps, const char * statusDir, virDomainObjPtr obj )`
- 4.2.3.236 `int virDomainSaveXML ( const char * configDir, virDomainDefPtr def, const char * xml )`
- 4.2.3.237 `static bool virDomainSerialDefCheckABIStability ( virDomainChrDefPtr src, virDomainChrDefPtr dst )`  
[static]
- 4.2.3.238 `static bool virDomainSmartcardDefCheckABIStability ( virDomainSmartcardDefPtr src, virDomainSmartcardDefPtr dst )` [static]
- 4.2.3.239 `int virDomainSmartcardDefForeach ( virDomainDefPtr def, bool abortOnError, virDomainSmartcardDefIterator iter, void * opaque )`
- 4.2.3.240 `static int virDomainSmartcardDefFormat ( virBufferPtr buf, virDomainSmartcardDefPtr def, unsigned int flags )`  
[static]



- 4.2.3.241 void virDomainSmartcardDefFree ( virDomainSmartcardDefPtr *def* )
- 4.2.3.242 static virDomainSmartcardDefPtr virDomainSmartcardDefParseXML ( xmlNodePtr *node*, unsigned int *flags* )  
[static]
- 4.2.3.243 static int virDomainSoundCodecDefFormat ( virBufferPtr *buf*, virDomainSoundCodecDefPtr *def* )  
[static]
- 4.2.3.244 void virDomainSoundCodecDefFree ( virDomainSoundCodecDefPtr *def* )
- 4.2.3.245 static virDomainSoundCodecDefPtr virDomainSoundCodecDefParseXML ( const xmlNodePtr *node* )  
[static]
- 4.2.3.246 static bool virDomainSoundDefCheckABIStability ( virDomainSoundDefPtr *src*, virDomainSoundDefPtr *dst* )  
[static]
- 4.2.3.247 static int virDomainSoundDefFormat ( virBufferPtr *buf*, virDomainSoundDefPtr *def*, unsigned int *flags* )  
[static]
- 4.2.3.248 void virDomainSoundDefFree ( virDomainSoundDefPtr *def* )
- 4.2.3.249 static virDomainSoundDefPtr virDomainSoundDefParseXML ( const xmlNodePtr *node*, xmlXPathContextPtr *ctxt*, unsigned int *flags* ) [static]
- 4.2.3.250 int virDomainStateReasonFromString ( virDomainState *state*, const char \* *reason* )
- 4.2.3.251 const char\* virDomainStateReasonToString ( virDomainState *state*, int *reason* )
- 4.2.3.252 static int virDomainSysinfoDefFormat ( virBufferPtr *buf*, virSysinfoDefPtr *def* ) [static]
- 4.2.3.253 static bool virDomainTimerDefCheckABIStability ( virDomainTimerDefPtr *src*, virDomainTimerDefPtr *dst* )  
[static]
- 4.2.3.254 static int virDomainTimerDefFormat ( virBufferPtr *buf*, virDomainTimerDefPtr *def* ) [static]
- 4.2.3.255 static virDomainTimerDefPtr virDomainTimerDefParseXML ( const xmlNodePtr *node*, xmlXPathContextPtr *ctxt* )  
[static]
- 4.2.3.256 int virDomainVcpuPinAdd ( virDomainVcpuPinDefPtr \*\* *vcpupin\_list*, int \* *nvcpupin*, unsigned char \* *cpumap*, int *maplen*, int *vcpu* )
- 4.2.3.257 void virDomainVcpuPinDefArrayFree ( virDomainVcpuPinDefPtr \* *def*, int *nvcpupin* )
- 4.2.3.258 virDomainVcpuPinDefPtr\* virDomainVcpuPinDefCopy ( virDomainVcpuPinDefPtr \* *src*, int *nvcpupin* )
- 4.2.3.259 void virDomainVcpuPinDefFree ( virDomainVcpuPinDefPtr *def* )
- 4.2.3.260 static virDomainVcpuPinDefPtr virDomainVcpuPinDefParseXML ( const xmlNodePtr *node*, xmlXPathContextPtr *ctxt*, int *maxvcpus*, int *emulator* ) [static]
- 4.2.3.261 int virDomainVcpuPinDel ( virDomainDefPtr *def*, int *vcpu* )
- 4.2.3.262 virDomainVcpuPinDefPtr virDomainVcpuPinFindByVcpu ( virDomainVcpuPinDefPtr \* *def*, int *nvcpupin*, int *vcpu* )
- 4.2.3.263 int virDomainVcpuPinsDuplicate ( virDomainVcpuPinDefPtr \* *def*, int *nvcpupin*, int *vcpu* )

- 4.2.3.264 `static void virDomainVideoAccelDefFormat ( virBufferPtr buf, virDomainVideoAccelDefPtr def )` `[static]`
- 4.2.3.265 `static virDomainVideoAccelDefPtr virDomainVideoAccelDefParseXML ( const xmlNodePtr node )`  
`[static]`
- 4.2.3.266 `int virDomainVideoDefaultRAM ( virDomainDefPtr def, int type )`
- 4.2.3.267 `int virDomainVideoDefaultType ( virDomainDefPtr def )`
- 4.2.3.268 `static bool virDomainVideoDefCheckABIStability ( virDomainVideoDefPtr src, virDomainVideoDefPtr dst )`  
`[static]`
- 4.2.3.269 `static int virDomainVideoDefFormat ( virBufferPtr buf, virDomainVideoDefPtr def, unsigned int flags )`  
`[static]`
- 4.2.3.270 `void virDomainVideoDefFree ( virDomainVideoDefPtr def )`
- 4.2.3.271 `static virDomainVideoDefPtr virDomainVideoDefParseXML ( const xmlNodePtr node, virDomainDefPtr dom, unsigned int flags )` `[static]`
- 4.2.3.272 `static bool virDomainWatchdogDefCheckABIStability ( virDomainWatchdogDefPtr src, virDomainWatchdogDefPtr dst )` `[static]`
- 4.2.3.273 `static int virDomainWatchdogDefFormat ( virBufferPtr buf, virDomainWatchdogDefPtr def, unsigned int flags )`  
`[static]`
- 4.2.3.274 `void virDomainWatchdogDefFree ( virDomainWatchdogDefPtr def )`
- 4.2.3.275 `static virDomainWatchdogDefPtr virDomainWatchdogDefParseXML ( const xmlNodePtr node, unsigned int flags )` `[static]`
- 4.2.3.276 `static void virSecurityDeviceLabelDefFormat ( virBufferPtr buf, virSecurityDeviceLabelDefPtr def )`  
`[static]`
- 4.2.3.277 `static void virSecurityDeviceLabelDefFree ( virSecurityDeviceLabelDefPtr def )` `[static]`
- 4.2.3.278 `static int virSecurityDeviceLabelDefParseXML ( virSecurityDeviceLabelDefPtr ** seclabels_rtn, size_t * nseclabels_rtn, virSecurityLabelDefPtr * vmSeclabels, int nvmSeclabels, xmlXPathContextPtr ctxt )`  
`[static]`
- 4.2.3.279 `static void virSecurityLabelDefFormat ( virBufferPtr buf, virSecurityLabelDefPtr def )` `[static]`
- 4.2.3.280 `static void virSecurityLabelDefFree ( virSecurityLabelDefPtr def )` `[static]`
- 4.2.3.281 `static virSecurityLabelDefPtr virSecurityLabelDefParseXML ( xmlXPathContextPtr ctxt, unsigned int flags )`  
`[static]`
- 4.2.3.282 `static int virSecurityLabelDefsParseXML ( virDomainDefPtr def, xmlXPathContextPtr ctxt, virCapsPtr caps, unsigned int flags )` `[static]`
- 4.2.3.283 `static virSysinfoDefPtr virSysinfoParseXML ( const xmlNodePtr node, xmlXPathContextPtr ctxt )` `[static]`

## 4.3 src/conf/domain\_conf.h File Reference

```
#include <libxml/parser.h>
#include <libxml/tree.h>
#include <libxml/xpath.h>
#include "internal.h"
#include "capabilities.h"
#include "storage_encryption_conf.h"
#include "cpu_conf.h"
#include "util.h"
#include "threads.h"
#include "virhash.h"
#include "virsocketaddr.h"
#include "nwfilter_params.h"
#include "nwfilter_conf.h"
#include "virnetdevmacvlan.h"
#include "sysinfo.h"
#include "virnetdevvportprofile.h"
#include "virnetdevopenvswitch.h"
#include "virnetdevbandwidth.h"
#include "virnetdevvlan.h"
#include "virobject.h"
#include "device_conf.h"
#include "bitmap.h"
```

### Data Structures

- struct [\\_virDomainDeviceDef](#)
- struct [\\_virDomainDeviceDriveAddress](#)
- struct [\\_virDomainDeviceVirtioSerialAddress](#)
- struct [\\_virDomainDeviceCcidAddress](#)
- struct [\\_virDomainDeviceUSBAddress](#)
- struct [\\_virDomainDeviceSpaprVioAddress](#)
- struct [\\_virDomainDeviceUSBMaster](#)
- struct [\\_virDomainDeviceInfo](#)
- struct [\\_virSecurityLabelDef](#)
- struct [\\_virSecurityDeviceLabelDef](#)
- struct [\\_virDomainHostdevOrigStates](#)
- struct [\\_virDomainLeaseDef](#)
- struct [\\_virDomainHostdevSubsys](#)
- struct [\\_virDomainHostdevDef](#)
- struct [\\_virDomainDiskHostDef](#)
- struct [\\_virDomainBlockIoTuneInfo](#)
- struct [\\_virDomainDiskDef](#)
- struct [\\_virDomainVirtioSerialOpts](#)
- struct [\\_virDomainControllerDef](#)
- struct [\\_virDomainFSDef](#)
- struct [\\_virDomainActualNetDef](#)
- struct [\\_virDomainNetDef](#)
- struct [\\_virDomainChrSourceDef](#)
- struct [\\_virDomainChrDef](#)
- struct [\\_virDomainSmartcardDef](#)
- struct [\\_virDomainHubDef](#)
- struct [\\_virDomainInputDef](#)

- struct [\\_virDomainSoundCodecDef](#)
- struct [\\_virDomainSoundDef](#)
- struct [\\_virDomainWatchdogDef](#)
- struct [\\_virDomainVideoAccelDef](#)
- struct [\\_virDomainVideoDef](#)
- struct [\\_virDomainGraphicsAuthDef](#)
- struct [\\_virDomainGraphicsListenDef](#)
- struct [\\_virDomainGraphicsDef](#)
- struct [\\_virDomainRedirdevDef](#)
- struct [\\_virDomainRedirFilterUsbDevDef](#)
- struct [\\_virDomainRedirFilterDef](#)
- struct [\\_virDomainMemballoonDef](#)
- struct [\\_virDomainBIOSDef](#)
- struct [\\_virDomainOSDef](#)
- struct [\\_virDomainTimerCatchupDef](#)
- struct [\\_virDomainTimerDef](#)
- struct [\\_virDomainClockDef](#)
- struct [\\_virDomainVcpuPinDef](#)
- struct [\\_virDomainNumatuneDef](#)
- struct [\\_virBlkioDeviceWeight](#)
- struct [\\_virDomainDef](#)
- struct [\\_virDomainStateReason](#)
- struct [\\_virDomainObj](#)
- struct [\\_virDomainObjList](#)

## Macros

- [#define VIR\\_DOMAIN\\_FS\\_RAM\\_DEFAULT\\_USAGE](#) (1024 \* 2)
- [#define VIR\\_NET\\_GENERATED\\_PREFIX](#) "vnet"
- [#define VIR\\_DOMAIN\\_SMARTCARD\\_NUM\\_CERTIFICATES](#) 3
- [#define VIR\\_DOMAIN\\_SMARTCARD\\_DEFAULT\\_DATABASE](#) "/etc/pki/nssdb"
- [#define VIR\\_DOMAIN\\_MAX\\_BOOT\\_DEVS](#) 4
- [#define VIR\\_DOMAIN\\_CPUMASK\\_LEN](#) 1024
- [#define VIR\\_CONNECT\\_LIST\\_DOMAINS\\_FILTERS\\_ACTIVE](#)
- [#define VIR\\_CONNECT\\_LIST\\_DOMAINS\\_FILTERS\\_PERSISTENT](#)
- [#define VIR\\_CONNECT\\_LIST\\_DOMAINS\\_FILTERS\\_STATE](#)
- [#define VIR\\_CONNECT\\_LIST\\_DOMAINS\\_FILTERS\\_MANAGEDSAVE](#)
- [#define VIR\\_CONNECT\\_LIST\\_DOMAINS\\_FILTERS\\_AUTOSTART](#)
- [#define VIR\\_CONNECT\\_LIST\\_DOMAINS\\_FILTERS\\_SNAPSHOT](#)
- [#define VIR\\_CONNECT\\_LIST\\_DOMAINS\\_FILTERS\\_ALL](#)

## Typedefs

- typedef struct [\\_virDomainDiskDef](#) [virDomainDiskDef](#)
- typedef [virDomainDiskDef](#) \* [virDomainDiskDefPtr](#)
- typedef struct [\\_virDomainControllerDef](#) [virDomainControllerDef](#)
- typedef [virDomainControllerDef](#) \* [virDomainControllerDefPtr](#)
- typedef struct [\\_virDomainLeaseDef](#) [virDomainLeaseDef](#)
- typedef [virDomainLeaseDef](#) \* [virDomainLeaseDefPtr](#)
- typedef struct [\\_virDomainFSDef](#) [virDomainFSDef](#)
- typedef [virDomainFSDef](#) \* [virDomainFSDefPtr](#)
- typedef struct [\\_virDomainNetDef](#) [virDomainNetDef](#)
- typedef [virDomainNetDef](#) \* [virDomainNetDefPtr](#)

- typedef struct [\\_virDomainInputDef](#) [virDomainInputDef](#)
- typedef [virDomainInputDef](#) \* [virDomainInputDefPtr](#)
- typedef struct [\\_virDomainSoundCodecDef](#) [virDomainSoundCodecDef](#)
- typedef [virDomainSoundCodecDef](#) \* [virDomainSoundCodecDefPtr](#)
- typedef struct [\\_virDomainSoundDef](#) [virDomainSoundDef](#)
- typedef [virDomainSoundDef](#) \* [virDomainSoundDefPtr](#)
- typedef struct [\\_virDomainVideoDef](#) [virDomainVideoDef](#)
- typedef [virDomainVideoDef](#) \* [virDomainVideoDefPtr](#)
- typedef struct [\\_virDomainHostdevDef](#) [virDomainHostdevDef](#)
- typedef [virDomainHostdevDef](#) \* [virDomainHostdevDefPtr](#)
- typedef struct [\\_virDomainWatchdogDef](#) [virDomainWatchdogDef](#)
- typedef [virDomainWatchdogDef](#) \* [virDomainWatchdogDefPtr](#)
- typedef struct [\\_virDomainGraphicsDef](#) [virDomainGraphicsDef](#)
- typedef [virDomainGraphicsDef](#) \* [virDomainGraphicsDefPtr](#)
- typedef struct [\\_virDomainHubDef](#) [virDomainHubDef](#)
- typedef [virDomainHubDef](#) \* [virDomainHubDefPtr](#)
- typedef struct [\\_virDomainRedirdevDef](#) [virDomainRedirdevDef](#)
- typedef [virDomainRedirdevDef](#) \* [virDomainRedirdevDefPtr](#)
- typedef struct [\\_virDomainRedirFilterUsbDevDef](#) [virDomainRedirFilterUsbDevDef](#)
- typedef [virDomainRedirFilterUsbDevDef](#) \* [virDomainRedirFilterUsbDevDefPtr](#)
- typedef struct [\\_virDomainRedirFilterDef](#) [virDomainRedirFilterDef](#)
- typedef [virDomainRedirFilterDef](#) \* [virDomainRedirFilterDefPtr](#)
- typedef struct [\\_virDomainSmartcardDef](#) [virDomainSmartcardDef](#)
- typedef [virDomainSmartcardDef](#) \* [virDomainSmartcardDefPtr](#)
- typedef struct [\\_virDomainChrDef](#) [virDomainChrDef](#)
- typedef [virDomainChrDef](#) \* [virDomainChrDefPtr](#)
- typedef struct [\\_virDomainMemballoonDef](#) [virDomainMemballoonDef](#)
- typedef [virDomainMemballoonDef](#) \* [virDomainMemballoonDefPtr](#)
- typedef struct [\\_virDomainSnapshotObj](#) [virDomainSnapshotObj](#)
- typedef [virDomainSnapshotObj](#) \* [virDomainSnapshotObjPtr](#)
- typedef struct [\\_virDomainSnapshotObjList](#) [virDomainSnapshotObjList](#)
- typedef [virDomainSnapshotObjList](#) \* [virDomainSnapshotObjListPtr](#)
- typedef struct [\\_virDomainDeviceDef](#) [virDomainDeviceDef](#)
- typedef [virDomainDeviceDef](#) \* [virDomainDeviceDefPtr](#)
- typedef struct [\\_virDomainDeviceDriveAddress](#) [virDomainDeviceDriveAddress](#)
- typedef [virDomainDeviceDriveAddress](#) \* [virDomainDeviceDriveAddressPtr](#)
- typedef struct [\\_virDomainDeviceVirtioSerialAddress](#) [virDomainDeviceVirtioSerialAddress](#)
- typedef [virDomainDeviceVirtioSerialAddress](#) \* [virDomainDeviceVirtioSerialAddressPtr](#)
- typedef struct [\\_virDomainDeviceCcidAddress](#) [virDomainDeviceCcidAddress](#)

- typedef  
    [virDomainDeviceCcidAddress](#) \* [virDomainDeviceCcidAddressPtr](#)
- typedef struct  
    [\\_virDomainDeviceUSBAddress](#) [virDomainDeviceUSBAddress](#)
- typedef [virDomainDeviceUSBAddress](#) \* [virDomainDeviceUSBAddressPtr](#)
- typedef struct  
    [\\_virDomainDeviceSpaprVioAddress](#) [virDomainDeviceSpaprVioAddress](#)
- typedef  
    [virDomainDeviceSpaprVioAddress](#) \* [virDomainDeviceSpaprVioAddressPtr](#)
- typedef struct  
    [\\_virDomainDeviceUSBMaster](#) [virDomainDeviceUSBMaster](#)
- typedef [virDomainDeviceUSBMaster](#) \* [virDomainDeviceUSBMasterPtr](#)
- typedef struct [\\_virDomainDeviceInfo](#) [virDomainDeviceInfo](#)
- typedef [virDomainDeviceInfo](#) \* [virDomainDeviceInfoPtr](#)
- typedef struct [\\_virSecurityLabelDef](#) [virSecurityLabelDef](#)
- typedef [virSecurityLabelDef](#) \* [virSecurityLabelDefPtr](#)
- typedef struct  
    [\\_virSecurityDeviceLabelDef](#) [virSecurityDeviceLabelDef](#)
- typedef [virSecurityDeviceLabelDef](#) \* [virSecurityDeviceLabelDefPtr](#)
- typedef struct  
    [\\_virDomainHostdevOrigStates](#) [virDomainHostdevOrigStates](#)
- typedef  
    [virDomainHostdevOrigStates](#) \* [virDomainHostdevOrigStatesPtr](#)
- typedef struct  
    [\\_virDomainHostdevSubsys](#) [virDomainHostdevSubsys](#)
- typedef [virDomainHostdevSubsys](#) \* [virDomainHostdevSubsysPtr](#)
- typedef struct  
    [\\_virDomainDiskHostDef](#) [virDomainDiskHostDef](#)
- typedef [virDomainDiskHostDef](#) \* [virDomainDiskHostDefPtr](#)
- typedef struct  
    [\\_virDomainBlockIoTuneInfo](#) [virDomainBlockIoTuneInfo](#)
- typedef [virDomainBlockIoTuneInfo](#) \* [virDomainBlockIoTuneInfoPtr](#)
- typedef struct  
    [\\_virDomainVirtioSerialOpts](#) [virDomainVirtioSerialOpts](#)
- typedef [virDomainVirtioSerialOpts](#) \* [virDomainVirtioSerialOptsPtr](#)
- typedef struct  
    [\\_virDomainActualNetDef](#) [virDomainActualNetDef](#)
- typedef [virDomainActualNetDef](#) \* [virDomainActualNetDefPtr](#)
- typedef struct  
    [\\_virDomainChrSourceDef](#) [virDomainChrSourceDef](#)
- typedef [virDomainChrSourceDef](#) \* [virDomainChrSourceDefPtr](#)
- typedef struct  
    [\\_virDomainVideoAccelDef](#) [virDomainVideoAccelDef](#)
- typedef [virDomainVideoAccelDef](#) \* [virDomainVideoAccelDefPtr](#)
- typedef struct  
    [\\_virDomainGraphicsAuthDef](#) [virDomainGraphicsAuthDef](#)
- typedef [virDomainGraphicsAuthDef](#) \* [virDomainGraphicsAuthDefPtr](#)
- typedef struct  
    [\\_virDomainGraphicsListenDef](#) [virDomainGraphicsListenDef](#)
- typedef  
    [virDomainGraphicsListenDef](#) \* [virDomainGraphicsListenDefPtr](#)
- typedef struct [\\_virDomainBIOSDef](#) [virDomainBIOSDef](#)
- typedef [virDomainBIOSDef](#) \* [virDomainBIOSDefPtr](#)
- typedef struct [\\_virDomainOSDef](#) [virDomainOSDef](#)
- typedef [virDomainOSDef](#) \* [virDomainOSDefPtr](#)

- typedef struct  
    \_virDomainTimerCatchupDef virDomainTimerCatchupDef
- typedef virDomainTimerCatchupDef \* virDomainTimerCatchupDefPtr
- typedef struct \_virDomainTimerDef virDomainTimerDef
- typedef virDomainTimerDef \* virDomainTimerDefPtr
- typedef struct \_virDomainClockDef virDomainClockDef
- typedef virDomainClockDef \* virDomainClockDefPtr
- typedef struct \_virDomainVcpuPinDef virDomainVcpuPinDef
- typedef virDomainVcpuPinDef \* virDomainVcpuPinDefPtr
- typedef struct  
    \_virDomainNumatuneDef virDomainNumatuneDef
- typedef virDomainNumatuneDef \* virDomainNumatuneDefPtr
- typedef struct  
    \_virBlkioDeviceWeight virBlkioDeviceWeight
- typedef virBlkioDeviceWeight \* virBlkioDeviceWeightPtr
- typedef struct \_virDomainDef virDomainDef
- typedef virDomainDef \* virDomainDefPtr
- typedef struct  
    \_virDomainStateReason virDomainStateReason
- typedef struct \_virDomainObj virDomainObj
- typedef virDomainObj \* virDomainObjPtr
- typedef struct \_virDomainObjList virDomainObjList
- typedef virDomainObjList \* virDomainObjListPtr
- typedef int(\* virDomainDeviceInfoCallback )(virDomainDefPtr def, virDomainDeviceDefPtr dev, virDomain-  
    DeviceInfoPtr info, void \*opaque)
- typedef void(\* virDomainLoadConfigNotify )(virDomainObjPtr dom, int newDomain, void \*opaque)
- typedef int(\* virDomainSmartcardDefIterator )(virDomainDefPtr def, virDomainSmartcardDefPtr dev, void  
    \*opaque)
- typedef int(\* virDomainChrDefIterator )(virDomainDefPtr def, virDomainChrDefPtr dev, void \*opaque)
- typedef int(\* virDomainDiskDefPathIterator )(virDomainDiskDefPtr disk, const char \*path, size\_t depth, void  
    \*opaque)
- typedef const char \*(\* virLifecycleToStringFunc )(int type)
- typedef int(\* virLifecycleFromStringFunc )(const char \*type)

## Enumerations

- enum virDomainDeviceType {  
    VIR\_DOMAIN\_DEVICE\_NONE = 0, VIR\_DOMAIN\_DEVICE\_DISK, VIR\_DOMAIN\_DEVICE\_LEASE, VIR\_ -  
    DOMAIN\_DEVICE\_FS,  
    VIR\_DOMAIN\_DEVICE\_NET, VIR\_DOMAIN\_DEVICE\_INPUT, VIR\_DOMAIN\_DEVICE\_SOUND, VIR\_DO-  
    MAIN\_DEVICE\_VIDEO,  
    VIR\_DOMAIN\_DEVICE\_HOSTDEV, VIR\_DOMAIN\_DEVICE\_WATCHDOG, VIR\_DOMAIN\_DEVICE\_CON-  
    TROLLER, VIR\_DOMAIN\_DEVICE\_GRAPHICS,  
    VIR\_DOMAIN\_DEVICE\_HUB, VIR\_DOMAIN\_DEVICE\_REDIRDEV, VIR\_DOMAIN\_DEVICE\_SMARTCAR-  
    D, VIR\_DOMAIN\_DEVICE\_CHR,  
    VIR\_DOMAIN\_DEVICE\_MEMBALLOON, VIR\_DOMAIN\_DEVICE\_LAST }
- enum virDomainVirtType {  
    VIR\_DOMAIN\_VIRT\_QEMU, VIR\_DOMAIN\_VIRT\_KQEMU, VIR\_DOMAIN\_VIRT\_KVM, VIR\_DOMAIN\_VI-  
    RT\_XEN,  
    VIR\_DOMAIN\_VIRT\_LXC, VIR\_DOMAIN\_VIRT\_UML, VIR\_DOMAIN\_VIRT\_OPENVZ, VIR\_DOMAIN\_VIR-  
    T\_TEST,  
    VIR\_DOMAIN\_VIRT\_VMWARE, VIR\_DOMAIN\_VIRT\_HYPERV, VIR\_DOMAIN\_VIRT\_VBOX, VIR\_DOMAI-  
    N\_VIRT\_PHYP,  
    VIR\_DOMAIN\_VIRT\_PARALLELS, VIR\_DOMAIN\_VIRT\_LAST }

- enum `virDomainDeviceAddressType` {  
`VIR_DOMAIN_DEVICE_ADDRESS_TYPE_NONE`, `VIR_DOMAIN_DEVICE_ADDRESS_TYPE_PCI`, `VIR_DOMAIN_DEVICE_ADDRESS_TYPE_DRIVE`, `VIR_DOMAIN_DEVICE_ADDRESS_TYPE_VIRTIO_SERIAL`,  
`VIR_DOMAIN_DEVICE_ADDRESS_TYPE_CCID`, `VIR_DOMAIN_DEVICE_ADDRESS_TYPE_USB`, `VIR_DOMAIN_DEVICE_ADDRESS_TYPE_SPAPRVIO`, `VIR_DOMAIN_DEVICE_ADDRESS_TYPE_VIRTIO_S390`,  
`VIR_DOMAIN_DEVICE_ADDRESS_TYPE_LAST` }
- enum `virDomainPciRombarMode` { `VIR_DOMAIN_PCI_ROMBAR_DEFAULT` = 0, `VIR_DOMAIN_PCI_ROMBAR_ON`, `VIR_DOMAIN_PCI_ROMBAR_OFF`, `VIR_DOMAIN_PCI_ROMBAR_LAST` }
- enum `virDomainControllerMaster` { `VIR_DOMAIN_CONTROLLER_MASTER_NONE`, `VIR_DOMAIN_CONTROLLER_MASTER_USB`, `VIR_DOMAIN_CONTROLLER_MASTER_LAST` }
- enum `virDomainSeclabelType` {  
`VIR_DOMAIN_SECLABEL_DEFAULT`, `VIR_DOMAIN_SECLABEL_NONE`, `VIR_DOMAIN_SECLABEL_DYNAMIC`, `VIR_DOMAIN_SECLABEL_STATIC`,  
`VIR_DOMAIN_SECLABEL_LAST` }
- enum `virDomainHostdevMode` { `VIR_DOMAIN_HOSTDEV_MODE_SUBSYS`, `VIR_DOMAIN_HOSTDEV_MODE_CAPABILITIES`, `VIR_DOMAIN_HOSTDEV_MODE_LAST` }
- enum `virDomainHostdevSubsysType` { `VIR_DOMAIN_HOSTDEV_SUBSYS_TYPE_USB`, `VIR_DOMAIN_HOSTDEV_SUBSYS_TYPE_PCI`, `VIR_DOMAIN_HOSTDEV_SUBSYS_TYPE_LAST` }
- enum `virDomainDiskType` {  
`VIR_DOMAIN_DISK_TYPE_BLOCK`, `VIR_DOMAIN_DISK_TYPE_FILE`, `VIR_DOMAIN_DISK_TYPE_DIR`,  
`VIR_DOMAIN_DISK_TYPE_NETWORK`,  
`VIR_DOMAIN_DISK_TYPE_LAST` }
- enum `virDomainDiskDevice` {  
`VIR_DOMAIN_DISK_DEVICE_DISK`, `VIR_DOMAIN_DISK_DEVICE_CDROM`, `VIR_DOMAIN_DISK_DEVICE_FLOPPY`, `VIR_DOMAIN_DISK_DEVICE_LUN`,  
`VIR_DOMAIN_DISK_DEVICE_LAST` }
- enum `virDomainDiskBus` {  
`VIR_DOMAIN_DISK_BUS_IDE`, `VIR_DOMAIN_DISK_BUS_FDC`, `VIR_DOMAIN_DISK_BUS_SCSI`, `VIR_DOMAIN_DISK_BUS_VIRTIO`,  
`VIR_DOMAIN_DISK_BUS_XEN`, `VIR_DOMAIN_DISK_BUS_USB`, `VIR_DOMAIN_DISK_BUS_UML`, `VIR_DOMAIN_DISK_BUS_SATA`,  
`VIR_DOMAIN_DISK_BUS_LAST` }
- enum `virDomainDiskCache` {  
`VIR_DOMAIN_DISK_CACHE_DEFAULT`, `VIR_DOMAIN_DISK_CACHE_DISABLE`, `VIR_DOMAIN_DISK_CACHE_WRITETHRU`, `VIR_DOMAIN_DISK_CACHE_WRITEBACK`,  
`VIR_DOMAIN_DISK_CACHE_DIRECTSYNC`, `VIR_DOMAIN_DISK_CACHE_UNSAFE`, `VIR_DOMAIN_DISK_CACHE_LAST` }
- enum `virDomainDiskErrorPolicy` {  
`VIR_DOMAIN_DISK_ERROR_POLICY_DEFAULT`, `VIR_DOMAIN_DISK_ERROR_POLICY_STOP`, `VIR_DOMAIN_DISK_ERROR_POLICY_REPORT`, `VIR_DOMAIN_DISK_ERROR_POLICY_IGNORE`,  
`VIR_DOMAIN_DISK_ERROR_POLICY_ENOSPACE`, `VIR_DOMAIN_DISK_ERROR_POLICY_LAST` }
- enum `virDomainDiskProtocol` { `VIR_DOMAIN_DISK_PROTOCOL_NBD`, `VIR_DOMAIN_DISK_PROTOCOL_RBD`, `VIR_DOMAIN_DISK_PROTOCOL_SHEEPDOG`, `VIR_DOMAIN_DISK_PROTOCOL_LAST` }
- enum `virDomainDiskTray` { `VIR_DOMAIN_DISK_TRAY_CLOSED`, `VIR_DOMAIN_DISK_TRAY_OPEN`, `VIR_DOMAIN_DISK_TRAY_LAST` }
- enum `virDomainDiskGeometryTrans` {  
`VIR_DOMAIN_DISK_TRANS_DEFAULT` = 0, `VIR_DOMAIN_DISK_TRANS_NONE`, `VIR_DOMAIN_DISK_TRANS_AUTO`, `VIR_DOMAIN_DISK_TRANS_LBA`,  
`VIR_DOMAIN_DISK_TRANS_LAST` }
- enum `virDomainDiskIo` { `VIR_DOMAIN_DISK_IO_DEFAULT`, `VIR_DOMAIN_DISK_IO_NATIVE`, `VIR_DOMAIN_DISK_IO_THREADS`, `VIR_DOMAIN_DISK_IO_LAST` }
- enum `virDomainIoEventFd` { `VIR_DOMAIN_IO_EVENT_FD_DEFAULT` = 0, `VIR_DOMAIN_IO_EVENT_FD_ON`, `VIR_DOMAIN_IO_EVENT_FD_OFF`, `VIR_DOMAIN_IO_EVENT_FD_LAST` }
- enum `virDomainVirtioEventIdx` { `VIR_DOMAIN_VIRTIO_EVENT_IDX_DEFAULT` = 0, `VIR_DOMAIN_VIRTIO_EVENT_IDX_ON`, `VIR_DOMAIN_VIRTIO_EVENT_IDX_OFF`, `VIR_DOMAIN_VIRTIO_EVENT_IDX_LAST` }



- enum `virDomainDiskCopyOnRead` { `VIR_DOMAIN_DISK_COPY_ON_READ_DEFAULT` = 0, `VIR_DOMAIN_DISK_COPY_ON_READ_ON`, `VIR_DOMAIN_DISK_COPY_ON_READ_OFF`, `VIR_DOMAIN_DISK_COPY_ON_READ_LAST` }
- enum `virDomainStartupPolicy` { `VIR_DOMAIN_STARTUP_POLICY_DEFAULT` = 0, `VIR_DOMAIN_STARTUP_POLICY_MANDATORY`, `VIR_DOMAIN_STARTUP_POLICY_REQUISITE`, `VIR_DOMAIN_STARTUP_POLICY_OPTIONAL`, `VIR_DOMAIN_STARTUP_POLICY_LAST` }
- enum `virDomainDiskSecretType` { `VIR_DOMAIN_DISK_SECRET_TYPE_NONE`, `VIR_DOMAIN_DISK_SECRET_TYPE_UUID`, `VIR_DOMAIN_DISK_SECRET_TYPE_USAGE`, `VIR_DOMAIN_DISK_SECRET_TYPE_LAST` }
- enum `virDomainControllerType` { `VIR_DOMAIN_CONTROLLER_TYPE_IDE`, `VIR_DOMAIN_CONTROLLER_TYPE_FDC`, `VIR_DOMAIN_CONTROLLER_TYPE_SCSI`, `VIR_DOMAIN_CONTROLLER_TYPE_SATA`, `VIR_DOMAIN_CONTROLLER_TYPE_VIRTIO_SERIAL`, `VIR_DOMAIN_CONTROLLER_TYPE_CCID`, `VIR_DOMAIN_CONTROLLER_TYPE_USB`, `VIR_DOMAIN_CONTROLLER_TYPE_LAST` }
- enum `virDomainControllerModelSCSI` { `VIR_DOMAIN_CONTROLLER_MODEL_SCSI_AUTO`, `VIR_DOMAIN_CONTROLLER_MODEL_SCSI_BUSLOGIC`, `VIR_DOMAIN_CONTROLLER_MODEL_SCSI_LSILOGIC`, `VIR_DOMAIN_CONTROLLER_MODEL_SCSI_LSIAS1068`, `VIR_DOMAIN_CONTROLLER_MODEL_SCSI_VMPVSCSI`, `VIR_DOMAIN_CONTROLLER_MODEL_SCSI_IBMVSCSI`, `VIR_DOMAIN_CONTROLLER_MODEL_SCSI_VIRTIO_SCSI`, `VIR_DOMAIN_CONTROLLER_MODEL_SCSI_LAST` }
- enum `virDomainControllerModelUSB` { `VIR_DOMAIN_CONTROLLER_MODEL_USB_PIIX3_UHCI`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_PIIX4_UHCI`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_EHCI`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_ICH9_EHCI1`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_ICH9_UHCI1`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_ICH9_UHCI2`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_ICH9_UHCI3`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_VT82C686B_UHCI`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_PCI_OHCI`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_NEC_XHCI`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_NONE`, `VIR_DOMAIN_CONTROLLER_MODEL_USB_LAST` }
- enum `virDomainFSType` { `VIR_DOMAIN_FS_TYPE_MOUNT`, `VIR_DOMAIN_FS_TYPE_BLOCK`, `VIR_DOMAIN_FS_TYPE_FILE`, `VIR_DOMAIN_FS_TYPE_TEMPLATE`, `VIR_DOMAIN_FS_TYPE_RAM`, `VIR_DOMAIN_FS_TYPE_BIND`, `VIR_DOMAIN_FS_TYPE_LAST` }
- enum `virDomainFSDriverType` { `VIR_DOMAIN_FS_DRIVER_TYPE_DEFAULT` = 0, `VIR_DOMAIN_FS_DRIVER_TYPE_PATH`, `VIR_DOMAIN_FS_DRIVER_TYPE_HANDLE`, `VIR_DOMAIN_FS_DRIVER_TYPE_LAST` }
- enum `virDomainFSAccessMode` { `VIR_DOMAIN_FS_ACCESSMODE_PASSTHROUGH`, `VIR_DOMAIN_FS_ACCESSMODE_MAPPED`, `VIR_DOMAIN_FS_ACCESSMODE_SQUASH`, `VIR_DOMAIN_FS_ACCESSMODE_LAST` }
- enum `virDomainFSWrpolicy` { `VIR_DOMAIN_FS_WRPOLICY_DEFAULT` = 0, `VIR_DOMAIN_FS_WRPOLICY_IMMEDIATE`, `VIR_DOMAIN_FS_WRPOLICY_LAST` }
- enum `virDomainNetType` { `VIR_DOMAIN_NET_TYPE_USER`, `VIR_DOMAIN_NET_TYPE_ETHERNET`, `VIR_DOMAIN_NET_TYPE_SERVER`, `VIR_DOMAIN_NET_TYPE_CLIENT`, `VIR_DOMAIN_NET_TYPE_MCAST`, `VIR_DOMAIN_NET_TYPE_NETWORK`, `VIR_DOMAIN_NET_TYPE_BRIDGE`, `VIR_DOMAIN_NET_TYPE_INTERNAL`, `VIR_DOMAIN_NET_TYPE_DIRECT`, `VIR_DOMAIN_NET_TYPE_HOSTDEV`, `VIR_DOMAIN_NET_TYPE_LAST` }
- enum `virDomainNetBackendType` { `VIR_DOMAIN_NET_BACKEND_TYPE_DEFAULT`, `VIR_DOMAIN_NET_BACKEND_TYPE_QEMU`, `VIR_DOMAIN_NET_BACKEND_TYPE_VHOST`, `VIR_DOMAIN_NET_BACKEND_TYPE_LAST` }
- enum `virDomainNetVirtioTxModeType` { `VIR_DOMAIN_NET_VIRTIO_TX_MODE_DEFAULT`, `VIR_DOMAIN_NET_VIRTIO_TX_MODE_IOTHREAD`, `VIR_DOMAIN_NET_VIRTIO_TX_MODE_TIMER`, `VIR_DOMAIN_NET_VIRTIO_TX_MODE_LAST` }

- enum `virDomainNetInterfaceLinkState` { `VIR_DOMAIN_NET_INTERFACE_LINK_STATE_DEFAULT` = 0, `VIR_DOMAIN_NET_INTERFACE_LINK_STATE_UP`, `VIR_DOMAIN_NET_INTERFACE_LINK_STATE_DOWN`, `VIR_DOMAIN_NET_INTERFACE_LINK_STATE_LAST` }
- enum `virDomainChrDeviceType` { `VIR_DOMAIN_CHR_DEVICE_TYPE_PARALLEL` = 0, `VIR_DOMAIN_CHR_DEVICE_TYPE_SERIAL`, `VIR_DOMAIN_CHR_DEVICE_TYPE_CONSOLE`, `VIR_DOMAIN_CHR_DEVICE_TYPE_CHANNEL`, `VIR_DOMAIN_CHR_DEVICE_TYPE_LAST` }
- enum `virDomainChrChannelTargetType` { `VIR_DOMAIN_CHR_CHANNEL_TARGET_TYPE_NONE` = 0, `VIR_DOMAIN_CHR_CHANNEL_TARGET_TYPE_GUESTFWD`, `VIR_DOMAIN_CHR_CHANNEL_TARGET_TYPE_VIRTIO`, `VIR_DOMAIN_CHR_CHANNEL_TARGET_TYPE_LAST` }
- enum `virDomainChrConsoleTargetType` { `VIR_DOMAIN_CHR_CONSOLE_TARGET_TYPE_SERIAL` = 0, `VIR_DOMAIN_CHR_CONSOLE_TARGET_TYPE_XEN`, `VIR_DOMAIN_CHR_CONSOLE_TARGET_TYPE_UML`, `VIR_DOMAIN_CHR_CONSOLE_TARGET_TYPE_VIRTIO`, `VIR_DOMAIN_CHR_CONSOLE_TARGET_TYPE_LXC`, `VIR_DOMAIN_CHR_CONSOLE_TARGET_TYPE_OPENVZ`, `VIR_DOMAIN_CHR_CONSOLE_TARGET_TYPE_LAST` }
- enum `virDomainChrType` { `VIR_DOMAIN_CHR_TYPE_NULL`, `VIR_DOMAIN_CHR_TYPE_VC`, `VIR_DOMAIN_CHR_TYPE_PTY`, `VIR_DOMAIN_CHR_TYPE_DEV`, `VIR_DOMAIN_CHR_TYPE_FILE`, `VIR_DOMAIN_CHR_TYPE_PIPE`, `VIR_DOMAIN_CHR_TYPE_STDIO`, `VIR_DOMAIN_CHR_TYPE_UDP`, `VIR_DOMAIN_CHR_TYPE_TCP`, `VIR_DOMAIN_CHR_TYPE_UNIX`, `VIR_DOMAIN_CHR_TYPE_SPICEVMC`, `VIR_DOMAIN_CHR_TYPE_LAST` }
- enum `virDomainChrTcpProtocol` { `VIR_DOMAIN_CHR_TCP_PROTOCOL_RAW`, `VIR_DOMAIN_CHR_TCP_PROTOCOL_TELNET`, `VIR_DOMAIN_CHR_TCP_PROTOCOL_TELNETS`, `VIR_DOMAIN_CHR_TCP_PROTOCOL_TLS`, `VIR_DOMAIN_CHR_TCP_PROTOCOL_LAST` }
- enum `virDomainChrSpicevmcName` { `VIR_DOMAIN_CHR_SPICEVMC_VDAGENT`, `VIR_DOMAIN_CHR_SPICEVMC_SMARTCARD`, `VIR_DOMAIN_CHR_SPICEVMC_USBREDIR`, `VIR_DOMAIN_CHR_SPICEVMC_LAST` }
- enum `virDomainSmartcardType` { `VIR_DOMAIN_SMARTCARD_TYPE_HOST`, `VIR_DOMAIN_SMARTCARD_TYPE_HOST_CERTIFICATES`, `VIR_DOMAIN_SMARTCARD_TYPE_PASSTHROUGH`, `VIR_DOMAIN_SMARTCARD_TYPE_LAST` }
- enum `virDomainInputType` { `VIR_DOMAIN_INPUT_TYPE_MOUSE`, `VIR_DOMAIN_INPUT_TYPE_TABLET`, `VIR_DOMAIN_INPUT_TYPE_LAST` }
- enum `virDomainInputBus` { `VIR_DOMAIN_INPUT_BUS_PS2`, `VIR_DOMAIN_INPUT_BUS_USB`, `VIR_DOMAIN_INPUT_BUS_XEN`, `VIR_DOMAIN_INPUT_BUS_LAST` }
- enum `virDomainSoundCodecType` { `VIR_DOMAIN_SOUND_CODEC_TYPE_DUPLEX`, `VIR_DOMAIN_SOUND_CODEC_TYPE_MICRO`, `VIR_DOMAIN_SOUND_CODEC_TYPE_LAST` }
- enum `virDomainSoundModel` { `VIR_DOMAIN_SOUND_MODEL_SB16`, `VIR_DOMAIN_SOUND_MODEL_ES1370`, `VIR_DOMAIN_SOUND_MODEL_PCSPK`, `VIR_DOMAIN_SOUND_MODEL_AC97`, `VIR_DOMAIN_SOUND_MODEL_ICH6`, `VIR_DOMAIN_SOUND_MODEL_LAST` }
- enum `virDomainWatchdogModel` { `VIR_DOMAIN_WATCHDOG_MODEL_I6300ESB`, `VIR_DOMAIN_WATCHDOG_MODEL_IB700`, `VIR_DOMAIN_WATCHDOG_MODEL_LAST` }
- enum `virDomainWatchdogAction` { `VIR_DOMAIN_WATCHDOG_ACTION_RESET`, `VIR_DOMAIN_WATCHDOG_ACTION_SHUTDOWN`, `VIR_DOMAIN_WATCHDOG_ACTION_POWEROFF`, `VIR_DOMAIN_WATCHDOG_ACTION_PAUSE`, `VIR_DOMAIN_WATCHDOG_ACTION_DUMP`, `VIR_DOMAIN_WATCHDOG_ACTION_NONE`, `VIR_DOMAIN_WATCHDOG_ACTION_LAST` }
- enum `virDomainVideoType` { `VIR_DOMAIN_VIDEO_TYPE_VGA`, `VIR_DOMAIN_VIDEO_TYPE_CIRRUS`, `VIR_DOMAIN_VIDEO_TYPE_VMVGA`, `VIR_DOMAIN_VIDEO_TYPE_XEN`, `VIR_DOMAIN_VIDEO_TYPE_VBOX`, `VIR_DOMAIN_VIDEO_TYPE_QXL`, `VIR_DOMAIN_VIDEO_TYPE_LAST` }

- enum `virDomainGraphicsType` {  
`VIR_DOMAIN_GRAPHICS_TYPE_SDL`, `VIR_DOMAIN_GRAPHICS_TYPE_VNC`, `VIR_DOMAIN_GRAPHICS_TYPE_RDP`, `VIR_DOMAIN_GRAPHICS_TYPE_DESKTOP`,  
`VIR_DOMAIN_GRAPHICS_TYPE_SPICE`, `VIR_DOMAIN_GRAPHICS_TYPE_LAST` }
- enum `virDomainGraphicsAuthConnectedType` {  
`VIR_DOMAIN_GRAPHICS_AUTH_CONNECTED_DEFAULT` = 0, `VIR_DOMAIN_GRAPHICS_AUTH_CONNECTED_FAIL`, `VIR_DOMAIN_GRAPHICS_AUTH_CONNECTED_DISCONNECT`, `VIR_DOMAIN_GRAPHICS_AUTH_CONNECTED_KEEP`,  
`VIR_DOMAIN_GRAPHICS_AUTH_CONNECTED_LAST` }
- enum `virDomainGraphicsSpiceChannelName` {  
`VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_MAIN`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_DISPLAY`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_INPUT`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_CURSOR`,  
`VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_PLAYBACK`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_RECORD`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_SMARTCARD`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_USBREDIR`,  
`VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_LAST` }
- enum `virDomainGraphicsSpiceChannelMode` { `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_MODE_ANY`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_MODE_SECURE`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_MODE_INSECURE`, `VIR_DOMAIN_GRAPHICS_SPICE_CHANNEL_MODE_LAST` }
- enum `virDomainGraphicsSpiceImageCompression` {  
`VIR_DOMAIN_GRAPHICS_SPICE_IMAGE_COMPRESSION_DEFAULT` = 0, `VIR_DOMAIN_GRAPHICS_SPICE_IMAGE_COMPRESSION_AUTO_GLZ`, `VIR_DOMAIN_GRAPHICS_SPICE_IMAGE_COMPRESSION_AUTO_LZ`, `VIR_DOMAIN_GRAPHICS_SPICE_IMAGE_COMPRESSION_QUIC`,  
`VIR_DOMAIN_GRAPHICS_SPICE_IMAGE_COMPRESSION_GLZ`, `VIR_DOMAIN_GRAPHICS_SPICE_IMAGE_COMPRESSION_LZ`, `VIR_DOMAIN_GRAPHICS_SPICE_IMAGE_COMPRESSION_OFF`, `VIR_DOMAIN_GRAPHICS_SPICE_IMAGE_COMPRESSION_LAST` }
- enum `virDomainGraphicsSpiceJpegCompression` {  
`VIR_DOMAIN_GRAPHICS_SPICE_JPEG_COMPRESSION_DEFAULT` = 0, `VIR_DOMAIN_GRAPHICS_SPICE_JPEG_COMPRESSION_AUTO`, `VIR_DOMAIN_GRAPHICS_SPICE_JPEG_COMPRESSION_NEVER`, `VIR_DOMAIN_GRAPHICS_SPICE_JPEG_COMPRESSION_ALWAYS`,  
`VIR_DOMAIN_GRAPHICS_SPICE_JPEG_COMPRESSION_LAST` }
- enum `virDomainGraphicsSpiceZlibCompression` {  
`VIR_DOMAIN_GRAPHICS_SPICE_ZLIB_COMPRESSION_DEFAULT` = 0, `VIR_DOMAIN_GRAPHICS_SPICE_ZLIB_COMPRESSION_AUTO`, `VIR_DOMAIN_GRAPHICS_SPICE_ZLIB_COMPRESSION_NEVER`, `VIR_DOMAIN_GRAPHICS_SPICE_ZLIB_COMPRESSION_ALWAYS`,  
`VIR_DOMAIN_GRAPHICS_SPICE_ZLIB_COMPRESSION_LAST` }
- enum `virDomainGraphicsSpicePlaybackCompression` { `VIR_DOMAIN_GRAPHICS_SPICE_PLAYBACK_COMPRESSION_DEFAULT` = 0, `VIR_DOMAIN_GRAPHICS_SPICE_PLAYBACK_COMPRESSION_ON`, `VIR_DOMAIN_GRAPHICS_SPICE_PLAYBACK_COMPRESSION_OFF`, `VIR_DOMAIN_GRAPHICS_SPICE_PLAYBACK_COMPRESSION_LAST` }
- enum `virDomainGraphicsSpiceMouseMode` { `VIR_DOMAIN_GRAPHICS_SPICE_MOUSE_MODE_DEFAULT` = 0, `VIR_DOMAIN_GRAPHICS_SPICE_MOUSE_MODE_SERVER`, `VIR_DOMAIN_GRAPHICS_SPICE_MOUSE_MODE_CLIENT`, `VIR_DOMAIN_GRAPHICS_SPICE_MOUSE_MODE_LAST` }
- enum `virDomainGraphicsSpiceStreamingMode` {  
`VIR_DOMAIN_GRAPHICS_SPICE_STREAMING_MODE_DEFAULT` = 0, `VIR_DOMAIN_GRAPHICS_SPICE_STREAMING_MODE_FILTER`, `VIR_DOMAIN_GRAPHICS_SPICE_STREAMING_MODE_ALL`, `VIR_DOMAIN_GRAPHICS_SPICE_STREAMING_MODE_OFF`,  
`VIR_DOMAIN_GRAPHICS_SPICE_STREAMING_MODE_LAST` }
- enum `virDomainGraphicsSpiceClipboardCypaste` { `VIR_DOMAIN_GRAPHICS_SPICE_CLIPBOARD_COPYPASTE_DEFAULT` = 0, `VIR_DOMAIN_GRAPHICS_SPICE_CLIPBOARD_COPYPASTE_YES`, `VIR_DOMAIN_GRAPHICS_SPICE_CLIPBOARD_COPYPASTE_NO`, `VIR_DOMAIN_GRAPHICS_SPICE_CLIPBOARD_COPYPASTE_LAST` }
- enum `virDomainGraphicsListenType` { `VIR_DOMAIN_GRAPHICS_LISTEN_TYPE_NONE` = 0, `VIR_DOMAIN_GRAPHICS_LISTEN_TYPE_ADDRESS`, `VIR_DOMAIN_GRAPHICS_LISTEN_TYPE_NETWORK`, `VIR_DOMAIN_GRAPHICS_LISTEN_TYPE_LAST` }
- enum `virDomainHubType` { `VIR_DOMAIN_HUB_TYPE_USB`, `VIR_DOMAIN_HUB_TYPE_LAST` }

- enum `virDomainRedirdevBus` { `VIR_DOMAIN_REDIRDEV_BUS_USB`, `VIR_DOMAIN_REDIRDEV_BUS_LAST` }
- enum `virDomainMemDump` { `VIR_DOMAIN_MEM_DUMP_DEFAULT` = 0, `VIR_DOMAIN_MEM_DUMP_ON`, `VIR_DOMAIN_MEM_DUMP_OFF`, `VIR_DOMAIN_MEM_DUMP_LAST` }
- enum { `VIR_DOMAIN_MEMBALLOON_MODEL_VIRTIO`, `VIR_DOMAIN_MEMBALLOON_MODEL_XEN`, `VIR_DOMAIN_MEMBALLOON_MODEL_NONE`, `VIR_DOMAIN_MEMBALLOON_MODEL_LAST` }
- enum `virDomainSmbiosMode` { `VIR_DOMAIN_SMBIOS_NONE`, `VIR_DOMAIN_SMBIOS_EMULATE`, `VIR_DOMAIN_SMBIOS_HOST`, `VIR_DOMAIN_SMBIOS_SYSINFO`, `VIR_DOMAIN_SMBIOS_LAST` }
- enum `virDomainBootOrder` { `VIR_DOMAIN_BOOT_FLOPPY`, `VIR_DOMAIN_BOOT_CDROM`, `VIR_DOMAIN_BOOT_DISK`, `VIR_DOMAIN_BOOT_NET`, `VIR_DOMAIN_BOOT_LAST` }
- enum `virDomainBootMenu` { `VIR_DOMAIN_BOOT_MENU_DEFAULT` = 0, `VIR_DOMAIN_BOOT_MENU_ENABLED`, `VIR_DOMAIN_BOOT_MENU_DISABLED`, `VIR_DOMAIN_BOOT_MENU_LAST` }
- enum `virDomainFeature` { `VIR_DOMAIN_FEATURE_ACPI`, `VIR_DOMAIN_FEATURE_APIC`, `VIR_DOMAIN_FEATURE_PAE`, `VIR_DOMAIN_FEATURE_HAP`, `VIR_DOMAIN_FEATURE_VIRIDIAN`, `VIR_DOMAIN_FEATURE_PRIVNET`, `VIR_DOMAIN_FEATURE_LAST` }
- enum `virDomainApicEoi` { `VIR_DOMAIN_APIC_EOI_DEFAULT` = 0, `VIR_DOMAIN_APIC_EOI_ON`, `VIR_DOMAIN_APIC_EOI_OFF`, `VIR_DOMAIN_APIC_EOI_LAST` }
- enum `virDomainLifecycleAction` { `VIR_DOMAIN_LIFECYCLE_DESTROY`, `VIR_DOMAIN_LIFECYCLE_RESTART`, `VIR_DOMAIN_LIFECYCLE_RESTART_RENAME`, `VIR_DOMAIN_LIFECYCLE_PRESERVE`, `VIR_DOMAIN_LIFECYCLE_LAST` }
- enum `virDomainLifecycleCrashAction` { `VIR_DOMAIN_LIFECYCLE_CRASH_DESTROY`, `VIR_DOMAIN_LIFECYCLE_CRASH_RESTART`, `VIR_DOMAIN_LIFECYCLE_CRASH_RESTART_RENAME`, `VIR_DOMAIN_LIFECYCLE_CRASH_PRESERVE`, `VIR_DOMAIN_LIFECYCLE_CRASH_COREDUMP_DESTROY`, `VIR_DOMAIN_LIFECYCLE_CRASH_COREDUMP_RESTART`, `VIR_DOMAIN_LIFECYCLE_CRASH_LAST` }
- enum `virDomainPMState` { `VIR_DOMAIN_PM_STATE_DEFAULT` = 0, `VIR_DOMAIN_PM_STATE_ENABLED`, `VIR_DOMAIN_PM_STATE_DISABLED`, `VIR_DOMAIN_PM_STATE_LAST` }
- enum `virDomainBIOSUseSerial` { `VIR_DOMAIN_BIOS_USESERIAL_DEFAULT` = 0, `VIR_DOMAIN_BIOS_USESERIAL_YES`, `VIR_DOMAIN_BIOS_USESERIAL_NO` }
- enum `virDomainTimerNameType` { `VIR_DOMAIN_TIMER_NAME_PLATFORM` = 0, `VIR_DOMAIN_TIMER_NAME_PIT`, `VIR_DOMAIN_TIMER_NAME_RTC`, `VIR_DOMAIN_TIMER_NAME_HPET`, `VIR_DOMAIN_TIMER_NAME_TSC`, `VIR_DOMAIN_TIMER_NAME_KVMCLOCK`, `VIR_DOMAIN_TIMER_NAME_LAST` }
- enum `virDomainTimerTrackType` { `VIR_DOMAIN_TIMER_TRACK_BOOT` = 0, `VIR_DOMAIN_TIMER_TRACK_GUEST`, `VIR_DOMAIN_TIMER_TRACK_WALL`, `VIR_DOMAIN_TIMER_TRACK_LAST` }
- enum `virDomainTimerTickpolicyType` { `VIR_DOMAIN_TIMER_TICKPOLICY_DELAY` = 0, `VIR_DOMAIN_TIMER_TICKPOLICY_CATCHUP`, `VIR_DOMAIN_TIMER_TICKPOLICY_MERGE`, `VIR_DOMAIN_TIMER_TICKPOLICY_DISCARD`, `VIR_DOMAIN_TIMER_TICKPOLICY_LAST` }
- enum `virDomainTimerModeType` { `VIR_DOMAIN_TIMER_MODE_AUTO` = 0, `VIR_DOMAIN_TIMER_MODE_NATIVE`, `VIR_DOMAIN_TIMER_MODE_EMULATE`, `VIR_DOMAIN_TIMER_MODE_PARAVIRT`, `VIR_DOMAIN_TIMER_MODE_SMPSAFE`, `VIR_DOMAIN_TIMER_MODE_LAST` }
- enum `virDomainCpuPlacementMode` { `VIR_DOMAIN_CPU_PLACEMENT_MODE_STATIC` = 0, `VIR_DOMAIN_CPU_PLACEMENT_MODE_AUTO`, `VIR_DOMAIN_CPU_PLACEMENT_MODE_LAST` }
- enum `virDomainNumatuneMemPlacementMode` { `VIR_DOMAIN_NUMATUNE_MEM_PLACEMENT_MODE_DEFAULT` = 0, `VIR_DOMAIN_NUMATUNE_MEM_PLACEMENT_MODE_STATIC`, `VIR_DOMAIN_NUMATUNE_MEM_PLACEMENT_MODE_AUTO`, `VIR_DOMAIN_NUMATUNE_MEM_PLACEMENT_MODE_LAST` }

- enum `virDomainClockOffsetType` {  
`VIR_DOMAIN_CLOCK_OFFSET_UTC` = 0, `VIR_DOMAIN_CLOCK_OFFSET_LOCALTIME` = 1, `VIR_DOMAIN_CLOCK_OFFSET_VARIABLE` = 2, `VIR_DOMAIN_CLOCK_OFFSET_TIMEZONE` = 3,  
`VIR_DOMAIN_CLOCK_OFFSET_LAST` }
- enum `virDomainClockBasis` { `VIR_DOMAIN_CLOCK_BASIS_UTC` = 0, `VIR_DOMAIN_CLOCK_BASIS_LOCALTIME` = 1, `VIR_DOMAIN_CLOCK_BASIS_LAST` }
- enum `virDomainTaintFlags` {  
`VIR_DOMAIN_TAINT_CUSTOM_ARGV`, `VIR_DOMAIN_TAINT_CUSTOM_MONITOR`, `VIR_DOMAIN_TAINT_HIGH_PRIVILEGES`, `VIR_DOMAIN_TAINT_SHELL_SCRIPTS`,  
`VIR_DOMAIN_TAINT_DISK_PROBING`, `VIR_DOMAIN_TAINT_EXTERNAL_LAUNCH`, `VIR_DOMAIN_TAINT_HOST_CPU`, `VIR_DOMAIN_TAINT_LAST` }

## Functions

- void `virDomainVcpuPinDefFree` (`virDomainVcpuPinDefPtr` def)
- void `virDomainVcpuPinDefArrayFree` (`virDomainVcpuPinDefPtr` \*def, int nvcupin)
- `virDomainVcpuPinDefPtr` \* `virDomainVcpuPinDefCopy` (`virDomainVcpuPinDefPtr` \*src, int nvcupin)
- int `virDomainVcpuPinIsDuplicate` (`virDomainVcpuPinDefPtr` \*def, int nvcupin, int vcpu)
- `virDomainVcpuPinDefPtr` `virDomainVcpuPinFindByVcpu` (`virDomainVcpuPinDefPtr` \*def, int nvcupin, int vcpu)
- void `virBlkioDeviceWeightArrayClear` (`virBlkioDeviceWeightPtr` deviceWeights, int ndevices)
- static bool `virDomainObjsActive` (`virDomainObjPtr` dom)
- `virDomainObjPtr` `virDomainObjNew` (`virCapsPtr` caps)
- int `virDomainObjListInit` (`virDomainObjListPtr` objs)
- void `virDomainObjListDeinit` (`virDomainObjListPtr` objs)
- `virDomainObjPtr` `virDomainFindByID` (const `virDomainObjListPtr` doms, int id)
- `virDomainObjPtr` `virDomainFindByUUID` (const `virDomainObjListPtr` doms, const unsigned char \*uuid)
- `virDomainObjPtr` `virDomainFindByName` (const `virDomainObjListPtr` doms, const char \*name)
- bool `virDomainObjTaint` (`virDomainObjPtr` obj, enum `virDomainTaintFlags` taint)
- void `virDomainGraphicsDefFree` (`virDomainGraphicsDefPtr` def)
- void `virDomainInputDefFree` (`virDomainInputDefPtr` def)
- void `virDomainDiskDefFree` (`virDomainDiskDefPtr` def)
- void `virDomainLeaseDefFree` (`virDomainLeaseDefPtr` def)
- void `virDomainDiskHostDefFree` (`virDomainDiskHostDefPtr` def)
- int `virDomainDiskFindControllerModel` (`virDomainDefPtr` def, `virDomainDiskDefPtr` disk, int controllerType)
- void `virDomainControllerDefFree` (`virDomainControllerDefPtr` def)
- void `virDomainFSDefFree` (`virDomainFSDefPtr` def)
- POL Mod void `virDomainActualNetDefFree` (`virDomainActualNetDefPtr` def)
- POL Mod void `virDomainNetDefFree` (`virDomainNetDefPtr` def)
- void `virDomainSmartcardDefFree` (`virDomainSmartcardDefPtr` def)
- void `virDomainChrDefFree` (`virDomainChrDefPtr` def)
- void `virDomainChrSourceDefFree` (`virDomainChrSourceDefPtr` def)
- int `virDomainChrSourceDefCopy` (`virDomainChrSourceDefPtr` src, `virDomainChrSourceDefPtr` dest)
- void `virDomainSoundCodecDefFree` (`virDomainSoundCodecDefPtr` def)
- void `virDomainSoundDefFree` (`virDomainSoundDefPtr` def)
- void `virDomainMemballoonDefFree` (`virDomainMemballoonDefPtr` def)
- void `virDomainWatchdogDefFree` (`virDomainWatchdogDefPtr` def)
- void `virDomainVideoDefFree` (`virDomainVideoDefPtr` def)
- `virDomainHostdevDefPtr` `virDomainHostdevDefAlloc` (void)
- void `virDomainHostdevDefClear` (`virDomainHostdevDefPtr` def)
- void `virDomainHostdevDefFree` (`virDomainHostdevDefPtr` def)
- void `virDomainHubDefFree` (`virDomainHubDefPtr` def)
- void `virDomainRedirdevDefFree` (`virDomainRedirdevDefPtr` def)
- void `virDomainRedirFilterDefFree` (`virDomainRedirFilterDefPtr` def)

- void [virDomainDeviceDefFree](#) (virDomainDeviceDefPtr def)
- [virDomainDeviceDefPtr virDomainDeviceDefCopy](#) (virCapsPtr caps, const [virDomainDefPtr](#) def, [virDomainDeviceDefPtr](#) src)
- int [virDomainDeviceAddressIsValid](#) (virDomainDeviceInfoPtr info, int type)
- void [virDomainDeviceInfoClear](#) (virDomainDeviceInfoPtr info)
- void [virDomainDefClearPCIAddresses](#) (virDomainDefPtr def)
- void [virDomainDefClearDeviceAliases](#) (virDomainDefPtr def)
- int [virDomainDeviceInfolterate](#) (virDomainDefPtr def, [virDomainDeviceInfoCallback](#) cb, void \*opaque)
- void [virDomainDefFree](#) (virDomainDefPtr vm)
- [virDomainChrDefPtr virDomainChrDefNew](#) (void)
- [virDomainObjPtr virDomainAssignDef](#) (virCapsPtr caps, [virDomainObjListPtr](#) doms, const [virDomainDefPtr](#) def, bool live)
- void [virDomainObjAssignDef](#) (virDomainObjPtr domain, const [virDomainDefPtr](#) def, bool live)
- int [virDomainObjSetDefTransient](#) (virCapsPtr caps, [virDomainObjPtr](#) domain, bool live)
- [virDomainDefPtr virDomainObjGetPersistentDef](#) (virCapsPtr caps, [virDomainObjPtr](#) domain)
- int [virDomainLiveConfigHelperMethod](#) (virCapsPtr caps, [virDomainObjPtr](#) dom, unsigned int \*flags, [virDomainDefPtr](#) \*persistentDef)
- [virDomainDefPtr virDomainObjCopyPersistentDef](#) (virCapsPtr caps, [virDomainObjPtr](#) dom)
- void [virDomainRemoveInactive](#) (virDomainObjListPtr doms, [virDomainObjPtr](#) dom)
- [virDomainDeviceDefPtr virDomainDeviceDefParse](#) (virCapsPtr caps, [virDomainDefPtr](#) def, const char \*xmlStr, unsigned int flags)
- [virDomainDefPtr virDomainDefParseString](#) (virCapsPtr caps, const char \*xmlStr, unsigned int expectedVirtTypes, unsigned int flags)
- [virDomainDefPtr virDomainDefParseFile](#) (virCapsPtr caps, const char \*filename, unsigned int expectedVirtTypes, unsigned int flags)
- [virDomainDefPtr virDomainDefParseNode](#) (virCapsPtr caps, xmlDocPtr doc, xmlNodePtr root, unsigned int expectedVirtTypes, unsigned int flags)
- bool [virDomainDefCheckABIStability](#) (virDomainDefPtr src, [virDomainDefPtr](#) dst)
- int [virDomainDefAddImplicitControllers](#) (virDomainDefPtr def)
- char \* [virDomainDefFormat](#) (virDomainDefPtr def, unsigned int flags)
- int [virDomainDefFormatInternal](#) (virDomainDefPtr def, unsigned int flags, virBufferPtr buf)
- int [virDomainDefCompatibleDevice](#) (virDomainDefPtr def, [virDomainDeviceDefPtr](#) dev)
- int [virDomainVcpuPinAdd](#) (virDomainVcpuPinDefPtr \*\*vcupin\_list, int \*nvcupin, unsigned char \*cpumap, int maplen, int vcpu)
- int [virDomainVcpuPinDel](#) (virDomainDefPtr def, int vcpu)
- int [virDomainEmulatorPinAdd](#) (virDomainDefPtr def, unsigned char \*cpumap, int maplen)
- int [virDomainEmulatorPinDel](#) (virDomainDefPtr def)
- int [virDomainDiskIndexByName](#) (virDomainDefPtr def, const char \*name, bool allow\_ambiguous)
- const char \* [virDomainDiskPathByName](#) (virDomainDefPtr, const char \*name)
- int [virDomainDiskInsert](#) (virDomainDefPtr def, [virDomainDiskDefPtr](#) disk)
- void [virDomainDiskInsertPreAlloced](#) (virDomainDefPtr def, [virDomainDiskDefPtr](#) disk)
- int [virDomainDiskDefAssignAddress](#) (virCapsPtr caps, [virDomainDiskDefPtr](#) def)
- [virDomainDiskDefPtr virDomainDiskRemove](#) (virDomainDefPtr def, size\_t i)
- [virDomainDiskDefPtr virDomainDiskRemoveByName](#) (virDomainDefPtr def, const char \*name)
- int [virDomainNetIndexByMac](#) (virDomainDefPtr def, const virMacAddrPtr mac)
- int [virDomainNetInsert](#) (virDomainDefPtr def, [virDomainNetDefPtr](#) net)
- [virDomainNetDefPtr virDomainNetRemove](#) (virDomainDefPtr def, size\_t i)
- [virDomainNetDefPtr virDomainNetRemoveByMac](#) (virDomainDefPtr def, const virMacAddrPtr mac)
- int [virDomainHostdevInsert](#) (virDomainDefPtr def, [virDomainHostdevDefPtr](#) hostdev)
- [virDomainHostdevDefPtr virDomainHostdevRemove](#) (virDomainDefPtr def, size\_t i)
- int [virDomainHostdevFind](#) (virDomainDefPtr def, [virDomainHostdevDefPtr](#) match, [virDomainHostdevDefPtr](#) \*found)
- int [virDomainGraphicsListenGetType](#) (virDomainGraphicsDefPtr def, size\_t ii) [ATTRIBUTE\\_NONNULL\(1\)](#)
- int [virDomainGraphicsListenSetType](#) (virDomainGraphicsDefPtr def, size\_t ii, int val) [ATTRIBUTE\\_NONNULL\(1\)](#)



- const char \* [virDomainGraphicsListenGetAddress](#) (virDomainGraphicsDefPtr def, size\_t ii) [ATTRIBUTE\\_NONNULL\(1\)](#)
- int [virDomainGraphicsListenSetAddress](#) (virDomainGraphicsDefPtr def, size\_t ii, const char \*address, int len, bool setType) [ATTRIBUTE\\_NONNULL\(1\)](#)
- const char \* [virDomainGraphicsListenGetNetwork](#) (virDomainGraphicsDefPtr def, size\_t ii) [ATTRIBUTE\\_NONNULL\(1\)](#)
- int [virDomainGraphicsListenSetNetwork](#) (virDomainGraphicsDefPtr def, size\_t ii, const char \*network, int len) [ATTRIBUTE\\_NONNULL\(1\)](#)
- int [virDomainNetGetActualType](#) (virDomainNetDefPtr iface)
- const char \* [virDomainNetGetActualBridgeName](#) (virDomainNetDefPtr iface)
- **POL New** const char \* [virDomainNetGetActualBridgeType](#) (virDomainNetDefPtr iface)
- const char \* [virDomainNetGetActualDirectDev](#) (virDomainNetDefPtr iface)
- int [virDomainNetGetActualDirectMode](#) (virDomainNetDefPtr iface)
- [virDomainHostdevDefPtr](#) [virDomainNetGetActualHostdev](#) (virDomainNetDefPtr iface)
- **POL Mod** virNetDevVPortProfilePtr [virDomainNetGetActualVirtPortProfile](#) (virDomainNetDefPtr iface)
- virNetDevBandwidthPtr [virDomainNetGetActualBandwidth](#) (virDomainNetDefPtr iface)
- virNetDevVlanPtr [virDomainNetGetActualVlan](#) (virDomainNetDefPtr iface)
- int [virDomainControllerInsert](#) (virDomainDefPtr def, virDomainControllerDefPtr controller)
- void [virDomainControllerInsertPreAlloced](#) (virDomainDefPtr def, virDomainControllerDefPtr controller)
- int [virDomainControllerFind](#) (virDomainDefPtr def, int type, int idx)
- [virDomainControllerDefPtr](#) [virDomainControllerRemove](#) (virDomainDefPtr def, size\_t i)
- int [virDomainLeaseIndex](#) (virDomainDefPtr def, virDomainLeaseDefPtr lease)
- int [virDomainLeaseInsert](#) (virDomainDefPtr def, virDomainLeaseDefPtr lease)
- int [virDomainLeaseInsertPreAlloc](#) (virDomainDefPtr def)
- void [virDomainLeaseInsertPreAlloced](#) (virDomainDefPtr def, virDomainLeaseDefPtr lease)
- [virDomainLeaseDefPtr](#) [virDomainLeaseRemoveAt](#) (virDomainDefPtr def, size\_t i)
- [virDomainLeaseDefPtr](#) [virDomainLeaseRemove](#) (virDomainDefPtr def, virDomainLeaseDefPtr lease)
- int [virDomainSaveXML](#) (const char \*configDir, virDomainDefPtr def, const char \*xml)
- int [virDomainSaveConfig](#) (const char \*configDir, virDomainDefPtr def)
- int [virDomainSaveStatus](#) (virCapsPtr caps, const char \*statusDir, virDomainObjPtr obj) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- int [virDomainLoadAllConfigs](#) (virCapsPtr caps, virDomainObjListPtr doms, const char \*configDir, const char \*autostartDir, int liveStatus, unsigned int expectedVirtTypes, virDomainLoadConfigNotify notify, void \*opaque)
- int [virDomainDeleteConfig](#) (const char \*configDir, const char \*autostartDir, virDomainObjPtr dom)
- char \* [virDomainConfigFile](#) (const char \*dir, const char \*name)
- int [virDiskNameToBusDeviceIndex](#) (virDomainDiskDefPtr disk, int \*busIdx, int \*devIdx)
- [virDomainFSDefPtr](#) [virDomainGetRootFilesystem](#) (virDomainDefPtr def)
- int [virDomainFSIndexByName](#) (virDomainDefPtr def, const char \*name)
- int [virDomainVideoDefaultType](#) (virDomainDefPtr def)
- int [virDomainVideoDefaultRAM](#) (virDomainDefPtr def, int type)
- int [virDomainObjsDuplicate](#) (virDomainObjListPtr doms, virDomainDefPtr def, unsigned int check\_active)
- void [virDomainObjLock](#) (virDomainObjPtr obj)
- void [virDomainObjUnlock](#) (virDomainObjPtr obj)
- int [virDomainObjListNumOfDomains](#) (virDomainObjListPtr doms, int active)
- int [virDomainObjListGetActiveIDs](#) (virDomainObjListPtr doms, int \*ids, int maxids)
- int [virDomainObjListGetInactiveNames](#) (virDomainObjListPtr doms, char \*\*const names, int maxnames)
- int [virDomainSmartcardDefForeach](#) (virDomainDefPtr def, bool abortOnError, virDomainSmartcardDefIterator iter, void \*opaque)
- int [virDomainChrDefForeach](#) (virDomainDefPtr def, bool abortOnError, virDomainChrDefIterator iter, void \*opaque)
- int [virDomainDiskDefForeachPath](#) (virDomainDiskDefPtr disk, bool allowProbing, bool ignoreOpenFailure, uid\_t uid, gid\_t gid, virDomainDiskDefPathIterator iter, void \*opaque)
- void [virDomainObjSetState](#) (virDomainObjPtr obj, virDomainState state, int reason) [ATTRIBUTE\\_NONNULL\(1\)](#)

- `virDomainState` `virDomainObjGetState` (`virDomainObjPtr` obj, int \*reason) `ATTRIBUTE_NONNULL(1)`
- `virSecurityLabelDefPtr` `virDomainDefGetSecurityLabelDef` (`virDomainDefPtr` def, const char \*model)
- `virSecurityDeviceLabelDefPtr` `virDomainDiskDefGetSecurityLabelDef` (`virDomainDiskDefPtr` def, const char \*model)
- `virSecurityDeviceLabelDefPtr` `virDomainChrDefGetSecurityLabelDef` (`virDomainChrDefPtr` def, const char \*model)
- `virSecurityLabelDefPtr` `virDomainDefAddSecurityLabelDef` (`virDomainDefPtr` def, const char \*model)
- const char \* `virDomainStateReasonToString` (`virDomainState` state, int reason)
- int `virDomainStateReasonFromString` (`virDomainState` state, const char \*reason)
- `virDomainNetDefPtr` `virDomainNetFind` (`virDomainDefPtr` def, const char \*device)
- int `virDomainList` (`virConnectPtr` conn, `virHashTablePtr` domobjs, `virDomainPtr` \*\*domains, unsigned int flags)

### 4.3.1 Macro Definition Documentation

#### 4.3.1.1 #define VIR\_CONNECT\_LIST\_DOMAINS\_FILTERS\_ACTIVE

**Value:**

```
(VIR_CONNECT_LIST_DOMAINS_ACTIVE | \
    VIR_CONNECT_LIST_DOMAINS_INACTIVE)
```

#### 4.3.1.2 #define VIR\_CONNECT\_LIST\_DOMAINS\_FILTERS\_ALL

**Value:**

```
(VIR_CONNECT_LIST_DOMAINS_FILTERS_ACTIVE | \
    VIR_CONNECT_LIST_DOMAINS_FILTERS_PERSISTENT | \
    VIR_CONNECT_LIST_DOMAINS_FILTERS_STATE | \
    VIR_CONNECT_LIST_DOMAINS_FILTERS_MANAGEDSAVE | \
    VIR_CONNECT_LIST_DOMAINS_FILTERS_AUTOSTART | \
    VIR_CONNECT_LIST_DOMAINS_FILTERS_SNAPSHOT)
```

#### 4.3.1.3 #define VIR\_CONNECT\_LIST\_DOMAINS\_FILTERS\_AUTOSTART

**Value:**

```
(VIR_CONNECT_LIST_DOMAINS_AUTOSTART | \
    VIR_CONNECT_LIST_DOMAINS_NO_AUTOSTART)
```

#### 4.3.1.4 #define VIR\_CONNECT\_LIST\_DOMAINS\_FILTERS\_MANAGEDSAVE

**Value:**

```
(VIR_CONNECT_LIST_DOMAINS_MANAGEDSAVE | \
    VIR_CONNECT_LIST_DOMAINS_NO_MANAGEDSAVE)
```

#### 4.3.1.5 #define VIR\_CONNECT\_LIST\_DOMAINS\_FILTERS\_PERSISTENT

**Value:**

```
(VIR_CONNECT_LIST_DOMAINS_PERSISTENT | \
    VIR_CONNECT_LIST_DOMAINS_TRANSIENT)
```



## 4.3.1.6 #define VIR\_CONNECT\_LIST\_DOMAINS\_FILTERS\_SNAPSHOT

**Value:**

```
(VIR_CONNECT_LIST_DOMAINS_HAS_SNAPSHOT | \
    VIR_CONNECT_LIST_DOMAINS_NO_SNAPSHOT)
```

## 4.3.1.7 #define VIR\_CONNECT\_LIST\_DOMAINS\_FILTERS\_STATE

**Value:**

```
(VIR_CONNECT_LIST_DOMAINS_RUNNING | \
    VIR_CONNECT_LIST_DOMAINS_PAUSED | \
    VIR_CONNECT_LIST_DOMAINS_SHUTOFF | \
    VIR_CONNECT_LIST_DOMAINS_OTHER)
```

## 4.3.1.8 #define VIR\_DOMAIN\_CPUMASK\_LEN 1024

## 4.3.1.9 #define VIR\_DOMAIN\_FS\_RAM\_DEFAULT\_USAGE (1024 \* 2)

## 4.3.1.10 #define VIR\_DOMAIN\_MAX\_BOOT\_DEVS 4

## 4.3.1.11 #define VIR\_DOMAIN\_SMARTCARD\_DEFAULT\_DATABASE "/etc/pki/nssdb"

## 4.3.1.12 #define VIR\_DOMAIN\_SMARTCARD\_NUM\_CERTIFICATES 3

## 4.3.1.13 #define VIR\_NET\_GENERATED\_PREFIX "vnet"

## 4.3.2 Typedef Documentation

## 4.3.2.1 typedef struct \_virBlkioDeviceWeight virBlkioDeviceWeight

## 4.3.2.2 typedef virBlkioDeviceWeight\* virBlkioDeviceWeightPtr

## 4.3.2.3 typedef struct \_virDomainActualNetDef virDomainActualNetDef

## 4.3.2.4 typedef virDomainActualNetDef\* virDomainActualNetDefPtr

## 4.3.2.5 typedef struct \_virDomainBIOSDef virDomainBIOSDef

## 4.3.2.6 typedef virDomainBIOSDef\* virDomainBIOSDefPtr

## 4.3.2.7 typedef struct \_virDomainBlockIoTuneInfo virDomainBlockIoTuneInfo

## 4.3.2.8 typedef virDomainBlockIoTuneInfo\* virDomainBlockIoTuneInfoPtr

## 4.3.2.9 typedef struct \_virDomainChrDef virDomainChrDef

## 4.3.2.10 typedef int(\* virDomainChrDefIterator)(virDomainDefPtr def, virDomainChrDefPtr dev, void \*opaque)

## 4.3.2.11 typedef virDomainChrDef\* virDomainChrDefPtr

## 4.3.2.12 typedef struct \_virDomainChrSourceDef virDomainChrSourceDef

## 4.3.2.13 typedef virDomainChrSourceDef\* virDomainChrSourceDefPtr

- 4.3.2.14 `typedef struct _virDomainClockDef virDomainClockDef`
- 4.3.2.15 `typedef virDomainClockDef* virDomainClockDefPtr`
- 4.3.2.16 `typedef struct _virDomainControllerDef virDomainControllerDef`
- 4.3.2.17 `typedef virDomainControllerDef* virDomainControllerDefPtr`
- 4.3.2.18 `typedef struct _virDomainDef virDomainDef`
- 4.3.2.19 `typedef virDomainDef* virDomainDefPtr`
- 4.3.2.20 `typedef struct _virDomainDeviceCcidAddress virDomainDeviceCcidAddress`
- 4.3.2.21 `typedef virDomainDeviceCcidAddress* virDomainDeviceCcidAddressPtr`
- 4.3.2.22 `typedef struct _virDomainDeviceDef virDomainDeviceDef`
- 4.3.2.23 `typedef virDomainDeviceDef* virDomainDeviceDefPtr`
- 4.3.2.24 `typedef struct _virDomainDeviceDriveAddress virDomainDeviceDriveAddress`
- 4.3.2.25 `typedef virDomainDeviceDriveAddress* virDomainDeviceDriveAddressPtr`
- 4.3.2.26 `typedef struct _virDomainDeviceInfo virDomainDeviceInfo`
- 4.3.2.27 `typedef int(* virDomainDeviceInfoCallback)(virDomainDefPtr def, virDomainDeviceDefPtr dev, virDomainDeviceInfoPtr info, void *opaque)`
- 4.3.2.28 `typedef virDomainDeviceInfo* virDomainDeviceInfoPtr`
- 4.3.2.29 `typedef struct _virDomainDeviceSpaprVioAddress virDomainDeviceSpaprVioAddress`
- 4.3.2.30 `typedef virDomainDeviceSpaprVioAddress* virDomainDeviceSpaprVioAddressPtr`
- 4.3.2.31 `typedef struct _virDomainDeviceUSBAddress virDomainDeviceUSBAddress`
- 4.3.2.32 `typedef virDomainDeviceUSBAddress* virDomainDeviceUSBAddressPtr`
- 4.3.2.33 `typedef struct _virDomainDeviceUSBMaster virDomainDeviceUSBMaster`
- 4.3.2.34 `typedef virDomainDeviceUSBMaster* virDomainDeviceUSBMasterPtr`
- 4.3.2.35 `typedef struct _virDomainDeviceVirtioSerialAddress virDomainDeviceVirtioSerialAddress`
- 4.3.2.36 `typedef virDomainDeviceVirtioSerialAddress* virDomainDeviceVirtioSerialAddressPtr`
- 4.3.2.37 `typedef struct _virDomainDiskDef virDomainDiskDef`
- 4.3.2.38 `typedef int(* virDomainDiskDefPathIterator)(virDomainDiskDefPtr disk, const char *path, size_t depth, void *opaque)`
- 4.3.2.39 `typedef virDomainDiskDef* virDomainDiskDefPtr`
- 4.3.2.40 `typedef struct _virDomainDiskHostDef virDomainDiskHostDef`

- 4.3.2.41 `typedef virDomainDiskHostDef* virDomainDiskHostDefPtr`
- 4.3.2.42 `typedef struct _virDomainFSDef virDomainFSDef`
- 4.3.2.43 `typedef virDomainFSDef* virDomainFSDefPtr`
- 4.3.2.44 `typedef struct _virDomainGraphicsAuthDef virDomainGraphicsAuthDef`
- 4.3.2.45 `typedef virDomainGraphicsAuthDef* virDomainGraphicsAuthDefPtr`
- 4.3.2.46 `typedef struct _virDomainGraphicsDef virDomainGraphicsDef`
- 4.3.2.47 `typedef virDomainGraphicsDef* virDomainGraphicsDefPtr`
- 4.3.2.48 `typedef struct _virDomainGraphicsListenDef virDomainGraphicsListenDef`
- 4.3.2.49 `typedef virDomainGraphicsListenDef* virDomainGraphicsListenDefPtr`
- 4.3.2.50 `typedef struct _virDomainHostdevDef virDomainHostdevDef`
- 4.3.2.51 `typedef virDomainHostdevDef* virDomainHostdevDefPtr`
- 4.3.2.52 `typedef struct _virDomainHostdevOrigStates virDomainHostdevOrigStates`
- 4.3.2.53 `typedef virDomainHostdevOrigStates* virDomainHostdevOrigStatesPtr`
- 4.3.2.54 `typedef struct _virDomainHostdevSubsys virDomainHostdevSubsys`
- 4.3.2.55 `typedef virDomainHostdevSubsys* virDomainHostdevSubsysPtr`
- 4.3.2.56 `typedef struct _virDomainHubDef virDomainHubDef`
- 4.3.2.57 `typedef virDomainHubDef* virDomainHubDefPtr`
- 4.3.2.58 `typedef struct _virDomainInputDef virDomainInputDef`
- 4.3.2.59 `typedef virDomainInputDef* virDomainInputDefPtr`
- 4.3.2.60 `typedef struct _virDomainLeaseDef virDomainLeaseDef`
- 4.3.2.61 `typedef virDomainLeaseDef* virDomainLeaseDefPtr`
- 4.3.2.62 `typedef void(* virDomainLoadConfigNotify)(virDomainObjPtr dom, int newDomain, void *opaque)`
- 4.3.2.63 `typedef struct _virDomainMemballoonDef virDomainMemballoonDef`
- 4.3.2.64 `typedef virDomainMemballoonDef* virDomainMemballoonDefPtr`
- 4.3.2.65 `typedef struct _virDomainNetDef virDomainNetDef`
- 4.3.2.66 `typedef virDomainNetDef* virDomainNetDefPtr`
- 4.3.2.67 `typedef struct _virDomainNumatuneDef virDomainNumatuneDef`
- 4.3.2.68 `typedef virDomainNumatuneDef* virDomainNumatuneDefPtr`

- 4.3.2.69 `typedef struct _virDomainObj virDomainObj`
- 4.3.2.70 `typedef struct _virDomainObjList virDomainObjList`
- 4.3.2.71 `typedef virDomainObjList* virDomainObjListPtr`
- 4.3.2.72 `typedef virDomainObj* virDomainObjPtr`
- 4.3.2.73 `typedef struct _virDomainOSDef virDomainOSDef`
- 4.3.2.74 `typedef virDomainOSDef* virDomainOSDefPtr`
- 4.3.2.75 `typedef struct _virDomainRedirdevDef virDomainRedirdevDef`
- 4.3.2.76 `typedef virDomainRedirdevDef* virDomainRedirdevDefPtr`
- 4.3.2.77 `typedef struct _virDomainRedirFilterDef virDomainRedirFilterDef`
- 4.3.2.78 `typedef virDomainRedirFilterDef* virDomainRedirFilterDefPtr`
- 4.3.2.79 `typedef struct _virDomainRedirFilterUsbDevDef virDomainRedirFilterUsbDevDef`
- 4.3.2.80 `typedef virDomainRedirFilterUsbDevDef* virDomainRedirFilterUsbDevDefPtr`
- 4.3.2.81 `typedef struct _virDomainSmartcardDef virDomainSmartcardDef`
- 4.3.2.82 `typedef int(* virDomainSmartcardDefIterator)(virDomainDefPtr def, virDomainSmartcardDefPtr dev, void *opaque)`
- 4.3.2.83 `typedef virDomainSmartcardDef* virDomainSmartcardDefPtr`
- 4.3.2.84 `typedef struct _virDomainSnapshotObj virDomainSnapshotObj`
- 4.3.2.85 `typedef struct _virDomainSnapshotObjList virDomainSnapshotObjList`
- 4.3.2.86 `typedef virDomainSnapshotObjList* virDomainSnapshotObjListPtr`
- 4.3.2.87 `typedef virDomainSnapshotObj* virDomainSnapshotObjPtr`
- 4.3.2.88 `typedef struct _virDomainSoundCodecDef virDomainSoundCodecDef`
- 4.3.2.89 `typedef virDomainSoundCodecDef* virDomainSoundCodecDefPtr`
- 4.3.2.90 `typedef struct _virDomainSoundDef virDomainSoundDef`
- 4.3.2.91 `typedef virDomainSoundDef* virDomainSoundDefPtr`
- 4.3.2.92 `typedef struct _virDomainStateReason virDomainStateReason`
- 4.3.2.93 `typedef struct _virDomainTimerCatchupDef virDomainTimerCatchupDef`
- 4.3.2.94 `typedef virDomainTimerCatchupDef* virDomainTimerCatchupDefPtr`
- 4.3.2.95 `typedef struct _virDomainTimerDef virDomainTimerDef`
- 4.3.2.96 `typedef virDomainTimerDef* virDomainTimerDefPtr`

- 4.3.2.97 `typedef struct _virDomainVcpuPinDef virDomainVcpuPinDef`
- 4.3.2.98 `typedef virDomainVcpuPinDef* virDomainVcpuPinDefPtr`
- 4.3.2.99 `typedef struct _virDomainVideoAccelDef virDomainVideoAccelDef`
- 4.3.2.100 `typedef virDomainVideoAccelDef* virDomainVideoAccelDefPtr`
- 4.3.2.101 `typedef struct _virDomainVideoDef virDomainVideoDef`
- 4.3.2.102 `typedef virDomainVideoDef* virDomainVideoDefPtr`
- 4.3.2.103 `typedef struct _virDomainVirtioSerialOpts virDomainVirtioSerialOpts`
- 4.3.2.104 `typedef virDomainVirtioSerialOpts* virDomainVirtioSerialOptsPtr`
- 4.3.2.105 `typedef struct _virDomainWatchdogDef virDomainWatchdogDef`
- 4.3.2.106 `typedef virDomainWatchdogDef* virDomainWatchdogDefPtr`
- 4.3.2.107 `typedef int(* virLifecycleFromStringFunc)(const char *type)`
- 4.3.2.108 `typedef const char*(* virLifecycleToStringFunc)(int type)`
- 4.3.2.109 `typedef struct _virSecurityDeviceLabelDef virSecurityDeviceLabelDef`
- 4.3.2.110 `typedef virSecurityDeviceLabelDef* virSecurityDeviceLabelDefPtr`
- 4.3.2.111 `typedef struct _virSecurityLabelDef virSecurityLabelDef`
- 4.3.2.112 `typedef virSecurityLabelDef* virSecurityLabelDefPtr`

### 4.3.3 Enumeration Type Documentation

#### 4.3.3.1 anonymous enum

Enumerator

***VIR\_DOMAIN\_MEMBALLOON\_MODEL\_VIRTIO***  
***VIR\_DOMAIN\_MEMBALLOON\_MODEL\_XEN***  
***VIR\_DOMAIN\_MEMBALLOON\_MODEL\_NONE***  
***VIR\_DOMAIN\_MEMBALLOON\_MODEL\_LAST***

#### 4.3.3.2 enum virDomainApicEoi

Enumerator

***VIR\_DOMAIN\_APIC\_EOI\_DEFAULT***  
***VIR\_DOMAIN\_APIC\_EOI\_ON***  
***VIR\_DOMAIN\_APIC\_EOI\_OFF***  
***VIR\_DOMAIN\_APIC\_EOI\_LAST***

#### 4.3.3.3 enum virDomainBIOSUseserial

Enumerator

***VIR\_DOMAIN\_BIOS\_USESERIAL\_DEFAULT***  
***VIR\_DOMAIN\_BIOS\_USESERIAL\_YES***  
***VIR\_DOMAIN\_BIOS\_USESERIAL\_NO***

#### 4.3.3.4 enum virDomainBootMenu

Enumerator

***VIR\_DOMAIN\_BOOT\_MENU\_DEFAULT***  
***VIR\_DOMAIN\_BOOT\_MENU\_ENABLED***  
***VIR\_DOMAIN\_BOOT\_MENU\_DISABLED***  
***VIR\_DOMAIN\_BOOT\_MENU\_LAST***

#### 4.3.3.5 enum virDomainBootOrder

Enumerator

***VIR\_DOMAIN\_BOOT\_FLOPPY***  
***VIR\_DOMAIN\_BOOT\_CDROM***  
***VIR\_DOMAIN\_BOOT\_DISK***  
***VIR\_DOMAIN\_BOOT\_NET***  
***VIR\_DOMAIN\_BOOT\_LAST***

#### 4.3.3.6 enum virDomainChrChannelTargetType

Enumerator

***VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE\_NONE***  
***VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE\_GUESTFWD***  
***VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE\_VIRTIO***  
***VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE\_LAST***

#### 4.3.3.7 enum virDomainChrConsoleTargetType

Enumerator

***VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_SERIAL***  
***VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_XEN***  
***VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_UML***  
***VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_VIRTIO***  
***VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_LXC***  
***VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_OPENVZ***  
***VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_LAST***

## 4.3.3.8 enum virDomainChrDeviceType

Enumerator

*VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_PARALLEL*  
*VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_SERIAL*  
*VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_CONSOLE*  
*VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_CHANNEL*  
*VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_LAST*

## 4.3.3.9 enum virDomainChrSpicevmcName

Enumerator

*VIR\_DOMAIN\_CHR\_SPICEVMC\_VDAGENT*  
*VIR\_DOMAIN\_CHR\_SPICEVMC\_SMARTCARD*  
*VIR\_DOMAIN\_CHR\_SPICEVMC\_USBREDIR*  
*VIR\_DOMAIN\_CHR\_SPICEVMC\_LAST*

## 4.3.3.10 enum virDomainChrTcpProtocol

Enumerator

*VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_RAW*  
*VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TELNET*  
*VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TELNETS*  
*VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TLS*  
*VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_LAST*

## 4.3.3.11 enum virDomainChrType

Enumerator

*VIR\_DOMAIN\_CHR\_TYPE\_NULL*  
*VIR\_DOMAIN\_CHR\_TYPE\_VC*  
*VIR\_DOMAIN\_CHR\_TYPE\_PTY*  
*VIR\_DOMAIN\_CHR\_TYPE\_DEV*  
*VIR\_DOMAIN\_CHR\_TYPE\_FILE*  
*VIR\_DOMAIN\_CHR\_TYPE\_PIPE*  
*VIR\_DOMAIN\_CHR\_TYPE\_STDIO*  
*VIR\_DOMAIN\_CHR\_TYPE\_UDP*  
*VIR\_DOMAIN\_CHR\_TYPE\_TCP*  
*VIR\_DOMAIN\_CHR\_TYPE\_UNIX*  
*VIR\_DOMAIN\_CHR\_TYPE\_SPICEVMC*  
*VIR\_DOMAIN\_CHR\_TYPE\_LAST*

## 4.3.3.12 enum virDomainClockBasis

Enumerator

*VIR\_DOMAIN\_CLOCK\_BASIS\_UTC*  
*VIR\_DOMAIN\_CLOCK\_BASIS\_LOCALTIME*  
*VIR\_DOMAIN\_CLOCK\_BASIS\_LAST*

## 4.3.3.13 enum virDomainClockOffsetType

Enumerator

*VIR\_DOMAIN\_CLOCK\_OFFSET\_UTC*  
*VIR\_DOMAIN\_CLOCK\_OFFSET\_LOCALTIME*  
*VIR\_DOMAIN\_CLOCK\_OFFSET\_VARIABLE*  
*VIR\_DOMAIN\_CLOCK\_OFFSET\_TIMEZONE*  
*VIR\_DOMAIN\_CLOCK\_OFFSET\_LAST*

## 4.3.3.14 enum virDomainControllerMaster

Enumerator

*VIR\_DOMAIN\_CONTROLLER\_MASTER\_NONE*  
*VIR\_DOMAIN\_CONTROLLER\_MASTER\_USB*  
*VIR\_DOMAIN\_CONTROLLER\_MASTER\_LAST*

## 4.3.3.15 enum virDomainControllerModelSCSI

Enumerator

*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_AUTO*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_BUSLOGIC*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_LSILOGIC*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_LSILOGIC*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_LSILOGIC*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_VMPVSCSI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_IBMVSCSI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_VIRTIO\_SCSI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_LAST*

## 4.3.3.16 enum virDomainControllerModelUSB

Enumerator

*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_PIIX3\_UHCI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_PIIX4\_UHCI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_EHCI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_ICH9\_EHCI1*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_ICH9\_UHCI1*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_ICH9\_UHCI2*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_ICH9\_UHCI3*



*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_VT82C686B\_UHCI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_PCI\_OHCI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_NEC\_XHCI*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_NONE*  
*VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_LAST*

#### 4.3.3.17 enum virDomainControllerType

Enumerator

*VIR\_DOMAIN\_CONTROLLER\_TYPE\_IDE*  
*VIR\_DOMAIN\_CONTROLLER\_TYPE\_FDC*  
*VIR\_DOMAIN\_CONTROLLER\_TYPE\_SCSI*  
*VIR\_DOMAIN\_CONTROLLER\_TYPE\_SATA*  
*VIR\_DOMAIN\_CONTROLLER\_TYPE\_VIRTIO\_SERIAL*  
*VIR\_DOMAIN\_CONTROLLER\_TYPE\_CCID*  
*VIR\_DOMAIN\_CONTROLLER\_TYPE\_USB*  
*VIR\_DOMAIN\_CONTROLLER\_TYPE\_LAST*

#### 4.3.3.18 enum virDomainCpuPlacementMode

Enumerator

*VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_STATIC*  
*VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_AUTO*  
*VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_LAST*

#### 4.3.3.19 enum virDomainDeviceAddressType

Enumerator

*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_NONE*  
*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_PCI*  
*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_DRIVE*  
*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_VIRTIO\_SERIAL*  
*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_CCID*  
*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_USB*  
*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_SPAPRVIO*  
*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_VIRTIO\_S390*  
*VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_LAST*

#### 4.3.3.20 enum virDomainDeviceType

Enumerator

*VIR\_DOMAIN\_DEVICE\_NONE*  
*VIR\_DOMAIN\_DEVICE\_DISK*  
*VIR\_DOMAIN\_DEVICE\_LEASE*

*VIR\_DOMAIN\_DEVICE\_FS*  
*VIR\_DOMAIN\_DEVICE\_NET*  
*VIR\_DOMAIN\_DEVICE\_INPUT*  
*VIR\_DOMAIN\_DEVICE\_SOUND*  
*VIR\_DOMAIN\_DEVICE\_VIDEO*  
*VIR\_DOMAIN\_DEVICE\_HOSTDEV*  
*VIR\_DOMAIN\_DEVICE\_WATCHDOG*  
*VIR\_DOMAIN\_DEVICE\_CONTROLLER*  
*VIR\_DOMAIN\_DEVICE\_GRAPHICS*  
*VIR\_DOMAIN\_DEVICE\_HUB*  
*VIR\_DOMAIN\_DEVICE\_REDIRDEV*  
*VIR\_DOMAIN\_DEVICE\_SMARTCARD*  
*VIR\_DOMAIN\_DEVICE\_CHR*  
*VIR\_DOMAIN\_DEVICE\_MEMBALLOON*  
*VIR\_DOMAIN\_DEVICE\_LAST*

#### 4.3.3.21 enum virDomainDiskBus

Enumerator

*VIR\_DOMAIN\_DISK\_BUS\_IDE*  
*VIR\_DOMAIN\_DISK\_BUS\_FDC*  
*VIR\_DOMAIN\_DISK\_BUS\_SCSI*  
*VIR\_DOMAIN\_DISK\_BUS\_VIRTIO*  
*VIR\_DOMAIN\_DISK\_BUS\_XEN*  
*VIR\_DOMAIN\_DISK\_BUS\_USB*  
*VIR\_DOMAIN\_DISK\_BUS\_UML*  
*VIR\_DOMAIN\_DISK\_BUS\_SATA*  
*VIR\_DOMAIN\_DISK\_BUS\_LAST*

#### 4.3.3.22 enum virDomainDiskCache

Enumerator

*VIR\_DOMAIN\_DISK\_CACHE\_DEFAULT*  
*VIR\_DOMAIN\_DISK\_CACHE\_DISABLE*  
*VIR\_DOMAIN\_DISK\_CACHE\_WRITETHRU*  
*VIR\_DOMAIN\_DISK\_CACHE\_WRITEBACK*  
*VIR\_DOMAIN\_DISK\_CACHE\_DIRECTSYNC*  
*VIR\_DOMAIN\_DISK\_CACHE\_UNSAFE*  
*VIR\_DOMAIN\_DISK\_CACHE\_LAST*

#### 4.3.3.23 enum virDomainDiskCopyOnRead

Enumerator

*VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_DEFAULT*  
*VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_ON*  
*VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_OFF*  
*VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_LAST*

## 4.3.3.24 enum virDomainDiskDevice

Enumerator

***VIR\_DOMAIN\_DISK\_DEVICE\_DISK***  
***VIR\_DOMAIN\_DISK\_DEVICE\_CDROM***  
***VIR\_DOMAIN\_DISK\_DEVICE\_FLOPPY***  
***VIR\_DOMAIN\_DISK\_DEVICE\_LUN***  
***VIR\_DOMAIN\_DISK\_DEVICE\_LAST***

## 4.3.3.25 enum virDomainDiskErrorPolicy

Enumerator

***VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_DEFAULT***  
***VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_STOP***  
***VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_REPORT***  
***VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_IGNORE***  
***VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_ENOSPACE***  
***VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_LAST***

## 4.3.3.26 enum virDomainDiskGeometryTrans

Enumerator

***VIR\_DOMAIN\_DISK\_TRANS\_DEFAULT***  
***VIR\_DOMAIN\_DISK\_TRANS\_NONE***  
***VIR\_DOMAIN\_DISK\_TRANS\_AUTO***  
***VIR\_DOMAIN\_DISK\_TRANS\_LBA***  
***VIR\_DOMAIN\_DISK\_TRANS\_LAST***

## 4.3.3.27 enum virDomainDiskIo

Enumerator

***VIR\_DOMAIN\_DISK\_IO\_DEFAULT***  
***VIR\_DOMAIN\_DISK\_IO\_NATIVE***  
***VIR\_DOMAIN\_DISK\_IO\_THREADS***  
***VIR\_DOMAIN\_DISK\_IO\_LAST***

## 4.3.3.28 enum virDomainDiskProtocol

Enumerator

***VIR\_DOMAIN\_DISK\_PROTOCOL\_NBD***  
***VIR\_DOMAIN\_DISK\_PROTOCOL\_RBD***  
***VIR\_DOMAIN\_DISK\_PROTOCOL\_SHEEPDOG***  
***VIR\_DOMAIN\_DISK\_PROTOCOL\_LAST***

## 4.3.3.29 enum virDomainDiskSecretType

Enumerator

*VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_NONE*  
*VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_UUID*  
*VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_USAGE*  
*VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_LAST*

## 4.3.3.30 enum virDomainDiskTray

Enumerator

*VIR\_DOMAIN\_DISK\_TRAY\_CLOSED*  
*VIR\_DOMAIN\_DISK\_TRAY\_OPEN*  
*VIR\_DOMAIN\_DISK\_TRAY\_LAST*

## 4.3.3.31 enum virDomainDiskType

Enumerator

*VIR\_DOMAIN\_DISK\_TYPE\_BLOCK*  
*VIR\_DOMAIN\_DISK\_TYPE\_FILE*  
*VIR\_DOMAIN\_DISK\_TYPE\_DIR*  
*VIR\_DOMAIN\_DISK\_TYPE\_NETWORK*  
*VIR\_DOMAIN\_DISK\_TYPE\_LAST*

## 4.3.3.32 enum virDomainFeature

Enumerator

*VIR\_DOMAIN\_FEATURE\_ACPI*  
*VIR\_DOMAIN\_FEATURE\_APIC*  
*VIR\_DOMAIN\_FEATURE\_PAE*  
*VIR\_DOMAIN\_FEATURE\_HAP*  
*VIR\_DOMAIN\_FEATURE\_VIRIDIAN*  
*VIR\_DOMAIN\_FEATURE\_PRIVNET*  
*VIR\_DOMAIN\_FEATURE\_LAST*

## 4.3.3.33 enum virDomainFSAccessMode

Enumerator

*VIR\_DOMAIN\_FS\_ACCESSMODE\_PASSTHROUGH*  
*VIR\_DOMAIN\_FS\_ACCESSMODE\_MAPPED*  
*VIR\_DOMAIN\_FS\_ACCESSMODE\_SQUASH*  
*VIR\_DOMAIN\_FS\_ACCESSMODE\_LAST*

## 4.3.3.34 enum virDomainFSDriverType

Enumerator

*VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_DEFAULT*  
*VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_PATH*  
*VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_HANDLE*  
*VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_LAST*

## 4.3.3.35 enum virDomainFSType

Enumerator

*VIR\_DOMAIN\_FS\_TYPE\_MOUNT*  
*VIR\_DOMAIN\_FS\_TYPE\_BLOCK*  
*VIR\_DOMAIN\_FS\_TYPE\_FILE*  
*VIR\_DOMAIN\_FS\_TYPE\_TEMPLATE*  
*VIR\_DOMAIN\_FS\_TYPE\_RAM*  
*VIR\_DOMAIN\_FS\_TYPE\_BIND*  
*VIR\_DOMAIN\_FS\_TYPE\_LAST*

## 4.3.3.36 enum virDomainFSWrpolicy

Enumerator

*VIR\_DOMAIN\_FS\_WRPOLICY\_DEFAULT*  
*VIR\_DOMAIN\_FS\_WRPOLICY\_IMMEDIATE*  
*VIR\_DOMAIN\_FS\_WRPOLICY\_LAST*

## 4.3.3.37 enum virDomainGraphicsAuthConnectedType

Enumerator

*VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_DEFAULT*  
*VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_FAIL*  
*VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_DISCONNECT*  
*VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_KEEP*  
*VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_LAST*

## 4.3.3.38 enum virDomainGraphicsListenType

Enumerator

*VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_NONE*  
*VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_ADDRESS*  
*VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_NETWORK*  
*VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_LAST*

## 4.3.3.39 enum virDomainGraphicsSpiceChannelMode

Enumerator

***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_ANY***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_SECURE***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_INSECURE***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_LAST***

## 4.3.3.40 enum virDomainGraphicsSpiceChannelName

Enumerator

***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MAIN***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_DISPLAY***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_INPUT***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_CURSOR***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_PLAYBACK***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_RECORD***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_SMARTCARD***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_USBREDIR***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_LAST***

## 4.3.3.41 enum virDomainGraphicsSpiceClipboardCypypaste

Enumerator

***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_DEFAULT***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_YES***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_NO***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_LAST***

## 4.3.3.42 enum virDomainGraphicsSpiceImageCompression

Enumerator

***VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_DEFAULT***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_AUTO\_GLZ***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_AUTO\_LZ***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_QUIC***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_GLZ***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_LZ***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_OFF***  
***VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_LAST***

## 4.3.3.43 enum virDomainGraphicsSpiceJpegCompression

Enumerator

*VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_DEFAULT*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_AUTO*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_NEVER*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_ALWAYS*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_LAST*

## 4.3.3.44 enum virDomainGraphicsSpiceMouseMode

Enumerator

*VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_DEFAULT*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_SERVER*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_CLIENT*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_LAST*

## 4.3.3.45 enum virDomainGraphicsSpicePlaybackCompression

Enumerator

*VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_DEFAULT*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_ON*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_OFF*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_LAST*

## 4.3.3.46 enum virDomainGraphicsSpiceStreamingMode

Enumerator

*VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_DEFAULT*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_FILTER*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_ALL*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_OFF*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_LAST*

## 4.3.3.47 enum virDomainGraphicsSpiceZlibCompression

Enumerator

*VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_DEFAULT*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_AUTO*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_NEVER*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_ALWAYS*  
*VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_LAST*

## 4.3.3.48 enum virDomainGraphicsType

Enumerator

*VIR\_DOMAIN\_GRAPHICS\_TYPE\_SDL*  
*VIR\_DOMAIN\_GRAPHICS\_TYPE\_VNC*  
*VIR\_DOMAIN\_GRAPHICS\_TYPE\_RDP*  
*VIR\_DOMAIN\_GRAPHICS\_TYPE\_DESKTOP*  
*VIR\_DOMAIN\_GRAPHICS\_TYPE\_SPICE*  
*VIR\_DOMAIN\_GRAPHICS\_TYPE\_LAST*

## 4.3.3.49 enum virDomainHostdevMode

Enumerator

*VIR\_DOMAIN\_HOSTDEV\_MODE\_SUBSYS*  
*VIR\_DOMAIN\_HOSTDEV\_MODE\_CAPABILITIES*  
*VIR\_DOMAIN\_HOSTDEV\_MODE\_LAST*

## 4.3.3.50 enum virDomainHostdevSubsysType

Enumerator

*VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_USB*  
*VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_PCI*  
*VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_LAST*

## 4.3.3.51 enum virDomainHubType

Enumerator

*VIR\_DOMAIN\_HUB\_TYPE\_USB*  
*VIR\_DOMAIN\_HUB\_TYPE\_LAST*

## 4.3.3.52 enum virDomainInputBus

Enumerator

*VIR\_DOMAIN\_INPUT\_BUS\_PS2*  
*VIR\_DOMAIN\_INPUT\_BUS\_USB*  
*VIR\_DOMAIN\_INPUT\_BUS\_XEN*  
*VIR\_DOMAIN\_INPUT\_BUS\_LAST*

## 4.3.3.53 enum virDomainInputType

Enumerator

*VIR\_DOMAIN\_INPUT\_TYPE\_MOUSE*  
*VIR\_DOMAIN\_INPUT\_TYPE\_TABLET*  
*VIR\_DOMAIN\_INPUT\_TYPE\_LAST*



## 4.3.3.54 enum virDomainIoEventFd

Enumerator

*VIR\_DOMAIN\_IO\_EVENT\_FD\_DEFAULT*  
*VIR\_DOMAIN\_IO\_EVENT\_FD\_ON*  
*VIR\_DOMAIN\_IO\_EVENT\_FD\_OFF*  
*VIR\_DOMAIN\_IO\_EVENT\_FD\_LAST*

## 4.3.3.55 enum virDomainLifecycleAction

Enumerator

*VIR\_DOMAIN\_LIFECYCLE\_DESTROY*  
*VIR\_DOMAIN\_LIFECYCLE\_RESTART*  
*VIR\_DOMAIN\_LIFECYCLE\_RESTART\_RENAME*  
*VIR\_DOMAIN\_LIFECYCLE\_PRESERVE*  
*VIR\_DOMAIN\_LIFECYCLE\_LAST*

## 4.3.3.56 enum virDomainLifecycleCrashAction

Enumerator

*VIR\_DOMAIN\_LIFECYCLE\_CRASH\_DESTROY*  
*VIR\_DOMAIN\_LIFECYCLE\_CRASH\_RESTART*  
*VIR\_DOMAIN\_LIFECYCLE\_CRASH\_RESTART\_RENAME*  
*VIR\_DOMAIN\_LIFECYCLE\_CRASH\_PRESERVE*  
*VIR\_DOMAIN\_LIFECYCLE\_CRASH\_COREDUMP\_DESTROY*  
*VIR\_DOMAIN\_LIFECYCLE\_CRASH\_COREDUMP\_RESTART*  
*VIR\_DOMAIN\_LIFECYCLE\_CRASH\_LAST*

## 4.3.3.57 enum virDomainMemDump

Enumerator

*VIR\_DOMAIN\_MEM\_DUMP\_DEFAULT*  
*VIR\_DOMAIN\_MEM\_DUMP\_ON*  
*VIR\_DOMAIN\_MEM\_DUMP\_OFF*  
*VIR\_DOMAIN\_MEM\_DUMP\_LAST*

## 4.3.3.58 enum virDomainNetBackendType

Enumerator

*VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_DEFAULT*  
*VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_QEMU*  
*VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_VHOST*  
*VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_LAST*

## 4.3.3.59 enum virDomainNetInterfaceLinkState

Enumerator

***VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_DEFAULT***  
***VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_UP***  
***VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_DOWN***  
***VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_LAST***

## 4.3.3.60 enum virDomainNetType

Enumerator

***VIR\_DOMAIN\_NET\_TYPE\_USER***  
***VIR\_DOMAIN\_NET\_TYPE\_ETHERNET***  
***VIR\_DOMAIN\_NET\_TYPE\_SERVER***  
***VIR\_DOMAIN\_NET\_TYPE\_CLIENT***  
***VIR\_DOMAIN\_NET\_TYPE\_MCAST***  
***VIR\_DOMAIN\_NET\_TYPE\_NETWORK***  
***VIR\_DOMAIN\_NET\_TYPE\_BRIDGE***  
***VIR\_DOMAIN\_NET\_TYPE\_INTERNAL***  
***VIR\_DOMAIN\_NET\_TYPE\_DIRECT***  
***VIR\_DOMAIN\_NET\_TYPE\_HOSTDEV***  
***VIR\_DOMAIN\_NET\_TYPE\_LAST***

## 4.3.3.61 enum virDomainNetVirtioTxModeType

Enumerator

***VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_DEFAULT***  
***VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_IOTHREAD***  
***VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_TIMER***  
***VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_LAST***

## 4.3.3.62 enum virDomainNumatuneMemPlacementMode

Enumerator

***VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_DEFAULT***  
***VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_STATIC***  
***VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_AUTO***  
***VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_LAST***

## 4.3.3.63 enum virDomainPciRombarMode

Enumerator

***VIR\_DOMAIN\_PCI\_ROMBAR\_DEFAULT***  
***VIR\_DOMAIN\_PCI\_ROMBAR\_ON***  
***VIR\_DOMAIN\_PCI\_ROMBAR\_OFF***  
***VIR\_DOMAIN\_PCI\_ROMBAR\_LAST***

## 4.3.3.64 enum virDomainPMState

Enumerator

***VIR\_DOMAIN\_PM\_STATE\_DEFAULT***  
***VIR\_DOMAIN\_PM\_STATE\_ENABLED***  
***VIR\_DOMAIN\_PM\_STATE\_DISABLED***  
***VIR\_DOMAIN\_PM\_STATE\_LAST***

## 4.3.3.65 enum virDomainRedirdevBus

Enumerator

***VIR\_DOMAIN\_REDIRDEV\_BUS\_USB***  
***VIR\_DOMAIN\_REDIRDEV\_BUS\_LAST***

## 4.3.3.66 enum virDomainSeclabelType

Enumerator

***VIR\_DOMAIN\_SECLABEL\_DEFAULT***  
***VIR\_DOMAIN\_SECLABEL\_NONE***  
***VIR\_DOMAIN\_SECLABEL\_DYNAMIC***  
***VIR\_DOMAIN\_SECLABEL\_STATIC***  
***VIR\_DOMAIN\_SECLABEL\_LAST***

## 4.3.3.67 enum virDomainSmartcardType

Enumerator

***VIR\_DOMAIN\_SMARTCARD\_TYPE\_HOST***  
***VIR\_DOMAIN\_SMARTCARD\_TYPE\_HOST\_CERTIFICATES***  
***VIR\_DOMAIN\_SMARTCARD\_TYPE\_PASSTHROUGH***  
***VIR\_DOMAIN\_SMARTCARD\_TYPE\_LAST***

## 4.3.3.68 enum virDomainSmbiosMode

Enumerator

***VIR\_DOMAIN\_SMBIOS\_NONE***  
***VIR\_DOMAIN\_SMBIOS\_EMULATE***  
***VIR\_DOMAIN\_SMBIOS\_HOST***  
***VIR\_DOMAIN\_SMBIOS\_SYSINFO***  
***VIR\_DOMAIN\_SMBIOS\_LAST***

## 4.3.3.69 enum virDomainSoundCodecType

Enumerator

***VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_DUPLEX***  
***VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_MICRO***  
***VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_LAST***

## 4.3.3.70 enum virDomainSoundModel

Enumerator

*VIR\_DOMAIN\_SOUND\_MODEL\_SB16*  
*VIR\_DOMAIN\_SOUND\_MODEL\_ES1370*  
*VIR\_DOMAIN\_SOUND\_MODEL\_PCSPK*  
*VIR\_DOMAIN\_SOUND\_MODEL\_AC97*  
*VIR\_DOMAIN\_SOUND\_MODEL\_ICH6*  
*VIR\_DOMAIN\_SOUND\_MODEL\_LAST*

## 4.3.3.71 enum virDomainStartupPolicy

Enumerator

*VIR\_DOMAIN\_STARTUP\_POLICY\_DEFAULT*  
*VIR\_DOMAIN\_STARTUP\_POLICY\_MANDATORY*  
*VIR\_DOMAIN\_STARTUP\_POLICY\_REQUISITE*  
*VIR\_DOMAIN\_STARTUP\_POLICY\_OPTIONAL*  
*VIR\_DOMAIN\_STARTUP\_POLICY\_LAST*

## 4.3.3.72 enum virDomainTaintFlags

Enumerator

*VIR\_DOMAIN\_TAINT\_CUSTOM\_ARGV*  
*VIR\_DOMAIN\_TAINT\_CUSTOM\_MONITOR*  
*VIR\_DOMAIN\_TAINT\_HIGH\_PRIVILEGES*  
*VIR\_DOMAIN\_TAINT\_SHELL\_SCRIPTS*  
*VIR\_DOMAIN\_TAINT\_DISK\_PROBING*  
*VIR\_DOMAIN\_TAINT\_EXTERNAL\_LAUNCH*  
*VIR\_DOMAIN\_TAINT\_HOST\_CPU*  
*VIR\_DOMAIN\_TAINT\_LAST*

## 4.3.3.73 enum virDomainTimerModeType

Enumerator

*VIR\_DOMAIN\_TIMER\_MODE\_AUTO*  
*VIR\_DOMAIN\_TIMER\_MODE\_NATIVE*  
*VIR\_DOMAIN\_TIMER\_MODE\_EMULATE*  
*VIR\_DOMAIN\_TIMER\_MODE\_PARAVIRT*  
*VIR\_DOMAIN\_TIMER\_MODE\_SMPSAFE*  
*VIR\_DOMAIN\_TIMER\_MODE\_LAST*

## 4.3.3.74 enum virDomainTimerNameType

Enumerator

```
VIR_DOMAIN_TIMER_NAME_PLATFORM  
VIR_DOMAIN_TIMER_NAME_PIT  
VIR_DOMAIN_TIMER_NAME_RTC  
VIR_DOMAIN_TIMER_NAME_HPET  
VIR_DOMAIN_TIMER_NAME_TSC  
VIR_DOMAIN_TIMER_NAME_KVMCLOCK  
VIR_DOMAIN_TIMER_NAME_LAST
```

## 4.3.3.75 enum virDomainTimerTickpolicyType

Enumerator

```
VIR_DOMAIN_TIMER_TICKPOLICY_DELAY  
VIR_DOMAIN_TIMER_TICKPOLICY_CATCHUP  
VIR_DOMAIN_TIMER_TICKPOLICY_MERGE  
VIR_DOMAIN_TIMER_TICKPOLICY_DISCARD  
VIR_DOMAIN_TIMER_TICKPOLICY_LAST
```

## 4.3.3.76 enum virDomainTimerTrackType

Enumerator

```
VIR_DOMAIN_TIMER_TRACK_BOOT  
VIR_DOMAIN_TIMER_TRACK_GUEST  
VIR_DOMAIN_TIMER_TRACK_WALL  
VIR_DOMAIN_TIMER_TRACK_LAST
```

## 4.3.3.77 enum virDomainVideoType

Enumerator

```
VIR_DOMAIN_VIDEO_TYPE_VGA  
VIR_DOMAIN_VIDEO_TYPE_CIRRUS  
VIR_DOMAIN_VIDEO_TYPE_VMVGA  
VIR_DOMAIN_VIDEO_TYPE_XEN  
VIR_DOMAIN_VIDEO_TYPE_VBOX  
VIR_DOMAIN_VIDEO_TYPE_QXL  
VIR_DOMAIN_VIDEO_TYPE_LAST
```

## 4.3.3.78 enum virDomainVirtioEventIdx

Enumerator

```
VIR_DOMAIN_VIRTIO_EVENT_IDX_DEFAULT  
VIR_DOMAIN_VIRTIO_EVENT_IDX_ON  
VIR_DOMAIN_VIRTIO_EVENT_IDX_OFF  
VIR_DOMAIN_VIRTIO_EVENT_IDX_LAST
```

## 4.3.3.79 enum virDomainVirtType

Enumerator

***VIR\_DOMAIN\_VIRT\_QEMU***  
***VIR\_DOMAIN\_VIRT\_KQEMU***  
***VIR\_DOMAIN\_VIRT\_KVM***  
***VIR\_DOMAIN\_VIRT\_XEN***  
***VIR\_DOMAIN\_VIRT\_LXC***  
***VIR\_DOMAIN\_VIRT\_UML***  
***VIR\_DOMAIN\_VIRT\_OPENVZ***  
***VIR\_DOMAIN\_VIRT\_TEST***  
***VIR\_DOMAIN\_VIRT\_VMWARE***  
***VIR\_DOMAIN\_VIRT\_HYPERV***  
***VIR\_DOMAIN\_VIRT\_VBOX***  
***VIR\_DOMAIN\_VIRT\_PHYP***  
***VIR\_DOMAIN\_VIRT\_PARALLELS***  
***VIR\_DOMAIN\_VIRT\_LAST***

## 4.3.3.80 enum virDomainWatchdogAction

Enumerator

***VIR\_DOMAIN\_WATCHDOG\_ACTION\_RESET***  
***VIR\_DOMAIN\_WATCHDOG\_ACTION\_SHUTDOWN***  
***VIR\_DOMAIN\_WATCHDOG\_ACTION\_POWEROFF***  
***VIR\_DOMAIN\_WATCHDOG\_ACTION\_PAUSE***  
***VIR\_DOMAIN\_WATCHDOG\_ACTION\_DUMP***  
***VIR\_DOMAIN\_WATCHDOG\_ACTION\_NONE***  
***VIR\_DOMAIN\_WATCHDOG\_ACTION\_LAST***

## 4.3.3.81 enum virDomainWatchdogModel

Enumerator

***VIR\_DOMAIN\_WATCHDOG\_MODEL\_I6300ESB***  
***VIR\_DOMAIN\_WATCHDOG\_MODEL\_IB700***  
***VIR\_DOMAIN\_WATCHDOG\_MODEL\_LAST***

## 4.3.4 Function Documentation

4.3.4.1 void virBlkioDeviceWeightArrayClear ( virBlkioDeviceWeightPtr deviceWeights, int ndevices )

4.3.4.2 int virDiskNameToBusDeviceIndex ( virDomainDiskDefPtr disk, int \* busidx, int \* devidx )

4.3.4.3 **POL Mod** void virDomainActualNetDefFree ( virDomainActualNetDefPtr def )

virDomainActualNetDefFree

Parameters

<i>def</i>	pointer to virDomainActualNetDef structure
------------	--

Free a virDomainActualNetDef structure.

POL modification:

- Free also *def*->data.bridge.brtype

**4.3.4.4** `virDomainObjPtr virDomainAssignDef ( virCapsPtr caps, virDomainObjListPtr doms, const virDomainDefPtr def, bool live )`

**4.3.4.5** `int virDomainChrDefForeach ( virDomainDefPtr def, bool abortOnError, virDomainChrDefIterator iter, void * opaque )`

**4.3.4.6** `void virDomainChrDefFree ( virDomainChrDefPtr def )`

**4.3.4.7** `virSecurityDeviceLabelDefPtr virDomainChrDefGetSecurityLabelDef ( virDomainChrDefPtr def, const char * model )`

**4.3.4.8** `virDomainChrDefPtr virDomainChrDefNew ( void )`

**4.3.4.9** `int virDomainChrSourceDefCopy ( virDomainChrSourceDefPtr src, virDomainChrSourceDefPtr dest )`

**4.3.4.10** `void virDomainChrSourceDefFree ( virDomainChrSourceDefPtr def )`

**4.3.4.11** `char* virDomainConfigFile ( const char * dir, const char * name )`

**4.3.4.12** `void virDomainControllerDefFree ( virDomainControllerDefPtr def )`

**4.3.4.13** `int virDomainControllerFind ( virDomainDefPtr def, int type, int idx )`

**4.3.4.14** `int virDomainControllerInsert ( virDomainDefPtr def, virDomainControllerDefPtr controller )`

**4.3.4.15** `void virDomainControllerInsertPreAlloced ( virDomainDefPtr def, virDomainControllerDefPtr controller )`

**4.3.4.16** `virDomainControllerDefPtr virDomainControllerRemove ( virDomainDefPtr def, size_t i )`

**4.3.4.17** `int virDomainDefAddImplicitControllers ( virDomainDefPtr def )`

**4.3.4.18** `virSecurityLabelDefPtr virDomainDefAddSecurityLabelDef ( virDomainDefPtr def, const char * model )`

**4.3.4.19** `bool virDomainDefCheckABIStability ( virDomainDefPtr src, virDomainDefPtr dst )`

**4.3.4.20** `void virDomainDefClearDeviceAliases ( virDomainDefPtr def )`

**4.3.4.21** `void virDomainDefClearPCIAddresses ( virDomainDefPtr def )`

**4.3.4.22** `int virDomainDefCompatibleDevice ( virDomainDefPtr def, virDomainDeviceDefPtr dev )`

**4.3.4.23** `char* virDomainDefFormat ( virDomainDefPtr def, unsigned int flags )`

**4.3.4.24** `int virDomainDefFormatInternal ( virDomainDefPtr def, unsigned int flags, virBufferPtr buf )`

**4.3.4.25** `void virDomainDefFree ( virDomainDefPtr vm )`

- 4.3.4.26 **virSecurityLabelDefPtr** virDomainDefGetSecurityLabelDef ( **virDomainDefPtr** *def*, const char \* *model* )
- 4.3.4.27 **virDomainDefPtr** virDomainDefParseFile ( **virCapsPtr** *caps*, const char \* *filename*, unsigned int *expectedVirtTypes*, unsigned int *flags* )
- 4.3.4.28 **virDomainDefPtr** virDomainDefParseNode ( **virCapsPtr** *caps*, xmlDocPtr *doc*, xmlNodePtr *root*, unsigned int *expectedVirtTypes*, unsigned int *flags* )
- 4.3.4.29 **virDomainDefPtr** virDomainDefParseString ( **virCapsPtr** *caps*, const char \* *xmlStr*, unsigned int *expectedVirtTypes*, unsigned int *flags* )
- 4.3.4.30 int virDomainDeleteConfig ( const char \* *configDir*, const char \* *autostartDir*, **virDomainObjPtr** *dom* )
- 4.3.4.31 int virDomainDeviceAddressIsValid ( **virDomainDeviceInfoPtr** *info*, int *type* )
- 4.3.4.32 **virDomainDeviceDefPtr** virDomainDeviceDefCopy ( **virCapsPtr** *caps*, const **virDomainDefPtr** *def*, **virDomainDeviceDefPtr** *src* )
- 4.3.4.33 void virDomainDeviceDefFree ( **virDomainDeviceDefPtr** *def* )
- 4.3.4.34 **virDomainDeviceDefPtr** virDomainDeviceDefParse ( **virCapsPtr** *caps*, **virDomainDefPtr** *def*, const char \* *xmlStr*, unsigned int *flags* )
- 4.3.4.35 void virDomainDeviceInfoClear ( **virDomainDeviceInfoPtr** *info* )
- 4.3.4.36 int virDomainDeviceInfoIterate ( **virDomainDefPtr** *def*, **virDomainDeviceInfoCallback** *cb*, void \* *opaque* )
- 4.3.4.37 int virDomainDiskDefAssignAddress ( **virCapsPtr** *caps*, **virDomainDiskDefPtr** *def* )
- 4.3.4.38 int virDomainDiskDefForeachPath ( **virDomainDiskDefPtr** *disk*, bool *allowProbing*, bool *ignoreOpenFailure*, uid\_t *uid*, gid\_t *gid*, **virDomainDiskDefPathIterator** *iter*, void \* *opaque* )
- 4.3.4.39 void virDomainDiskDefFree ( **virDomainDiskDefPtr** *def* )
- 4.3.4.40 **virSecurityDeviceLabelDefPtr** virDomainDiskDefGetSecurityLabelDef ( **virDomainDiskDefPtr** *def*, const char \* *model* )
- 4.3.4.41 int virDomainDiskFindControllerModel ( **virDomainDefPtr** *def*, **virDomainDiskDefPtr** *disk*, int *controllerType* )
- 4.3.4.42 void virDomainDiskHostDefFree ( **virDomainDiskHostDefPtr** *def* )
- 4.3.4.43 int virDomainDiskIndexByName ( **virDomainDefPtr** *def*, const char \* *name*, bool *allow\_ambiguous* )
- 4.3.4.44 int virDomainDiskInsert ( **virDomainDefPtr** *def*, **virDomainDiskDefPtr** *disk* )
- 4.3.4.45 void virDomainDiskInsertPreAlloced ( **virDomainDefPtr** *def*, **virDomainDiskDefPtr** *disk* )
- 4.3.4.46 const char\* virDomainDiskPathByName ( **virDomainDefPtr** , const char \* *name* )
- 4.3.4.47 **virDomainDiskDefPtr** virDomainDiskRemove ( **virDomainDefPtr** *def*, size\_t *i* )
- 4.3.4.48 **virDomainDiskDefPtr** virDomainDiskRemoveByName ( **virDomainDefPtr** *def*, const char \* *name* )
- 4.3.4.49 int virDomainEmulatorPinAdd ( **virDomainDefPtr** *def*, unsigned char \* *cpumap*, int *maplen* )
- 4.3.4.50 int virDomainEmulatorPinDel ( **virDomainDefPtr** *def* )



- 4.3.4.51 `virDomainObjPtr virDomainFindByID ( const virDomainObjListPtr doms, int id )`
- 4.3.4.52 `virDomainObjPtr virDomainFindByName ( const virDomainObjListPtr doms, const char * name )`
- 4.3.4.53 `virDomainObjPtr virDomainFindByUUID ( const virDomainObjListPtr doms, const unsigned char * uuid )`
- 4.3.4.54 `void virDomainFSDefFree ( virDomainFSDefPtr def )`
- 4.3.4.55 `int virDomainFSIndexByName ( virDomainDefPtr def, const char * name )`
- 4.3.4.56 `virDomainFSDefPtr virDomainGetRootFilesystem ( virDomainDefPtr def )`
- 4.3.4.57 `void virDomainGraphicsDefFree ( virDomainGraphicsDefPtr def )`
- 4.3.4.58 `const char* virDomainGraphicsListenGetAddress ( virDomainGraphicsDefPtr def, size_t ii )`
- 4.3.4.59 `const char* virDomainGraphicsListenGetNetwork ( virDomainGraphicsDefPtr def, size_t ii )`
- 4.3.4.60 `int virDomainGraphicsListenGetType ( virDomainGraphicsDefPtr def, size_t ii )`
- 4.3.4.61 `int virDomainGraphicsListenSetAddress ( virDomainGraphicsDefPtr def, size_t ii, const char * address, int len, bool setType )`
- 4.3.4.62 `int virDomainGraphicsListenSetNetwork ( virDomainGraphicsDefPtr def, size_t ii, const char * network, int len )`
- 4.3.4.63 `int virDomainGraphicsListenSetType ( virDomainGraphicsDefPtr def, size_t ii, int val )`
- 4.3.4.64 `virDomainHostdevDefPtr virDomainHostdevDefAlloc ( void )`
- 4.3.4.65 `void virDomainHostdevDefClear ( virDomainHostdevDefPtr def )`
- 4.3.4.66 `void virDomainHostdevDefFree ( virDomainHostdevDefPtr def )`
- 4.3.4.67 `int virDomainHostdevFind ( virDomainDefPtr def, virDomainHostdevDefPtr match, virDomainHostdevDefPtr * found )`
- 4.3.4.68 `int virDomainHostdevInsert ( virDomainDefPtr def, virDomainHostdevDefPtr hostdev )`
- 4.3.4.69 `virDomainHostdevDefPtr virDomainHostdevRemove ( virDomainDefPtr def, size_t i )`
- 4.3.4.70 `void virDomainHubDefFree ( virDomainHubDefPtr def )`
- 4.3.4.71 `void virDomainInputDefFree ( virDomainInputDefPtr def )`
- 4.3.4.72 `void virDomainLeaseDefFree ( virDomainLeaseDefPtr def )`
- 4.3.4.73 `int virDomainLeaseIndex ( virDomainDefPtr def, virDomainLeaseDefPtr lease )`
- 4.3.4.74 `int virDomainLeaseInsert ( virDomainDefPtr def, virDomainLeaseDefPtr lease )`
- 4.3.4.75 `int virDomainLeaseInsertPreAlloc ( virDomainDefPtr def )`
- 4.3.4.76 `void virDomainLeaseInsertPreAlloced ( virDomainDefPtr def, virDomainLeaseDefPtr lease )`
- 4.3.4.77 `virDomainLeaseDefPtr virDomainLeaseRemove ( virDomainDefPtr def, virDomainLeaseDefPtr lease )`

4.3.4.78 **virDomainLeaseDefPtr** virDomainLeaseRemoveAt ( **virDomainDefPtr** *def*, **size\_t** *i* )

4.3.4.79 **int** virDomainList ( **virConnectPtr** *conn*, **virHashTablePtr** *domobjs*, **virDomainPtr** \*\* *domains*, **unsigned int** *flags* )

4.3.4.80 **int** virDomainLiveConfigHelperMethod ( **virCapsPtr** *caps*, **virDomainObjPtr** *dom*, **unsigned int** \* *flags*, **virDomainDefPtr** \* *persistentDef* )

4.3.4.81 **int** virDomainLoadAllConfigs ( **virCapsPtr** *caps*, **virDomainObjListPtr** *doms*, **const char** \* *configDir*, **const char** \* *autostartDir*, **int** *liveStatus*, **unsigned int** *expectedVirtTypes*, **virDomainLoadConfigNotify** *notify*, **void** \* *opaque* )

4.3.4.82 **void** virDomainMemballoonDefFree ( **virDomainMemballoonDefPtr** *def* )

4.3.4.83 **POL Mod** **void** virDomainNetDefFree ( **virDomainNetDefPtr** *def* )

virDomainNetDefFree

Parameters

<i>def</i>	pointer to virDomainNetDef structure
------------	--------------------------------------

Free a virDomainNetDef structure.

POL modification:

- Free also *def->data.bridge.brtype*

4.3.4.84 **virDomainNetDefPtr** virDomainNetFind ( **virDomainDefPtr** *def*, **const char** \* *device* )

4.3.4.85 **virNetDevBandwidthPtr** virDomainNetGetActualBandwidth ( **virDomainNetDefPtr** *iface* )

4.3.4.86 **const char\*** virDomainNetGetActualBridgeName ( **virDomainNetDefPtr** *iface* )

4.3.4.87 **POL New** **const char\*** virDomainNetGetActualBridgeType ( **virDomainNetDefPtr** *iface* )

virDomainNetGetActualBridgeType

Parameters

<i>iface</i>	pointer to virDomainNetDef structure
--------------	--------------------------------------

Return the bridge type from a virDomainNetDef structure

4.3.4.88 **const char\*** virDomainNetGetActualDirectDev ( **virDomainNetDefPtr** *iface* )

4.3.4.89 **int** virDomainNetGetActualDirectMode ( **virDomainNetDefPtr** *iface* )

4.3.4.90 **virDomainHostdevDefPtr** virDomainNetGetActualHostdev ( **virDomainNetDefPtr** *iface* )

4.3.4.91 **int** virDomainNetGetActualType ( **virDomainNetDefPtr** *iface* )

4.3.4.92 **POL Mod** **virNetDevVPortProfilePtr** virDomainNetGetActualVirtPortProfile ( **virDomainNetDefPtr** *iface* )

virDomainNetGetActualVirtPortProfile

## Parameters

<i>def</i>	pointer to virDomainNetDef structure
------------	--------------------------------------

Return the actual virtual port profile from a virDomainNetDef structure

POL modification:

- Added support for interfaces with type 'network'

- 4.3.4.93 virNetDevVlanPtr virDomainNetGetActualVlan ( virDomainNetDefPtr iface )
- 4.3.4.94 int virDomainNetIndexByMac ( virDomainDefPtr def, const virMacAddrPtr mac )
- 4.3.4.95 int virDomainNetInsert ( virDomainDefPtr def, virDomainNetDefPtr net )
- 4.3.4.96 virDomainNetDefPtr virDomainNetRemove ( virDomainDefPtr def, size\_t i )
- 4.3.4.97 virDomainNetDefPtr virDomainNetRemoveByMac ( virDomainDefPtr def, const virMacAddrPtr mac )
- 4.3.4.98 void virDomainObjAssignDef ( virDomainObjPtr domain, const virDomainDefPtr def, bool live )
- 4.3.4.99 virDomainDefPtr virDomainObjCopyPersistentDef ( virCapsPtr caps, virDomainObjPtr dom )
- 4.3.4.100 virDomainDefPtr virDomainObjGetPersistentDef ( virCapsPtr caps, virDomainObjPtr domain )
- 4.3.4.101 virDomainState virDomainObjGetState ( virDomainObjPtr obj, int \* reason )
- 4.3.4.102 static bool virDomainObjsActive ( virDomainObjPtr dom ) [inline],[static]
- 4.3.4.103 int virDomainObjsDuplicate ( virDomainObjListPtr doms, virDomainDefPtr def, unsigned int check\_active )
- 4.3.4.104 void virDomainObjListDeinit ( virDomainObjListPtr objs )
- 4.3.4.105 int virDomainObjListGetActiveIDs ( virDomainObjListPtr doms, int \* ids, int maxids )
- 4.3.4.106 int virDomainObjListGetInactiveNames ( virDomainObjListPtr doms, char \*\*const names, int maxnames )
- 4.3.4.107 int virDomainObjListInit ( virDomainObjListPtr objs )
- 4.3.4.108 int virDomainObjListNumOfDomains ( virDomainObjListPtr doms, int active )
- 4.3.4.109 void virDomainObjLock ( virDomainObjPtr obj )
- 4.3.4.110 virDomainObjPtr virDomainObjNew ( virCapsPtr caps )
- 4.3.4.111 int virDomainObjSetDefTransient ( virCapsPtr caps, virDomainObjPtr domain, bool live )
- 4.3.4.112 void virDomainObjSetState ( virDomainObjPtr obj, virDomainState state, int reason )
- 4.3.4.113 bool virDomainObjTaint ( virDomainObjPtr obj, enum virDomainTaintFlags taint )
- 4.3.4.114 void virDomainObjUnlock ( virDomainObjPtr obj )
- 4.3.4.115 void virDomainRedirdevDefFree ( virDomainRedirdevDefPtr def )

- 4.3.4.116 void virDomainRedirFilterDefFree ( virDomainRedirFilterDefPtr *def* )
- 4.3.4.117 void virDomainRemovelnactive ( virDomainObjListPtr *doms*, virDomainObjPtr *dom* )
- 4.3.4.118 int virDomainSaveConfig ( const char \* *configDir*, virDomainDefPtr *def* )
- 4.3.4.119 int virDomainSaveStatus ( virCapsPtr *caps*, const char \* *statusDir*, virDomainObjPtr *obj* )
- 4.3.4.120 int virDomainSaveXML ( const char \* *configDir*, virDomainDefPtr *def*, const char \* *xml* )
- 4.3.4.121 int virDomainSmartcardDefForeach ( virDomainDefPtr *def*, bool *abortOnError*,  
virDomainSmartcardDefIterator *iter*, void \* *opaque* )
- 4.3.4.122 void virDomainSmartcardDefFree ( virDomainSmartcardDefPtr *def* )
- 4.3.4.123 void virDomainSoundCodecDefFree ( virDomainSoundCodecDefPtr *def* )
- 4.3.4.124 void virDomainSoundDefFree ( virDomainSoundDefPtr *def* )
- 4.3.4.125 int virDomainStateReasonFromString ( virDomainState *state*, const char \* *reason* )
- 4.3.4.126 const char\* virDomainStateReasonToString ( virDomainState *state*, int *reason* )
- 4.3.4.127 int virDomainVcpuPinAdd ( virDomainVcpuPinDefPtr \*\* *vcupin\_list*, int \* *nvcupin*, unsigned char \* *cpumap*,  
int *maplen*, int *vcpu* )
- 4.3.4.128 void virDomainVcpuPinDefArrayFree ( virDomainVcpuPinDefPtr \* *def*, int *nvcupin* )
- 4.3.4.129 virDomainVcpuPinDefPtr\* virDomainVcpuPinDefCopy ( virDomainVcpuPinDefPtr \* *src*, int *nvcupin* )
- 4.3.4.130 void virDomainVcpuPinDefFree ( virDomainVcpuPinDefPtr *def* )
- 4.3.4.131 int virDomainVcpuPinDel ( virDomainDefPtr *def*, int *vcpu* )
- 4.3.4.132 virDomainVcpuPinDefPtr virDomainVcpuPinFindByVcpu ( virDomainVcpuPinDefPtr \* *def*, int *nvcupin*,  
int *vcpu* )
- 4.3.4.133 int virDomainVcpuPinsDuplicate ( virDomainVcpuPinDefPtr \* *def*, int *nvcupin*, int *vcpu* )
- 4.3.4.134 int virDomainVideoDefaultRAM ( virDomainDefPtr *def*, int *type* )
- 4.3.4.135 int virDomainVideoDefaultType ( virDomainDefPtr *def* )
- 4.3.4.136 void virDomainVideoDefFree ( virDomainVideoDefPtr *def* )
- 4.3.4.137 void virDomainWatchdogDefFree ( virDomainWatchdogDefPtr *def* )

## 4.4 src/conf/network\_conf.c File Reference

```
#include <config.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <dirent.h>
#include "virterror_internal.h"
#include "datatypes.h"
#include "network_conf.h"
#include "netdev_vport_profile_conf.h"
#include "netdev_bandwidth_conf.h"
#include "netdev_vlan_conf.h"
#include "memory.h"
#include "xml.h"
#include "uuid.h"
#include "util.h"
#include "buf.h"
#include "c-ctype.h"
#include "virfile.h"
#include "../util/virnetdevopenvswitch.h"
```

### Macros

- `#define MAX_BRIDGE_ID` 256
- `#define VIR_FROM_THIS` VIR\_FROM\_NETWORK
- `#define MATCH(FLAG)` (flags & (FLAG))

### Functions

- `VIR_ENUM_IMPL` (virNetworkForward, `VIR_NETWORK_FORWARD_LAST`, `VIR_ENUM_IMPL("none","nat","route","bridge","p`
- `virNetworkObjPtr virNetworkFindByName` (const `virNetworkObjListPtr` nets, const char \*name)
- static void `virPortGroupDefClear` (`virPortGroupDefPtr` def)
- static void `virNetworkForwardIfDefClear` (`virNetworkForwardIfDefPtr` def)
- static void `virNetworkForwardPfDefClear` (`virNetworkForwardPfDefPtr` def)
- static void `virNetworkDHCPHostDefClear` (`virNetworkDHCPHostDefPtr` def)
- static void `virNetworkIpDefClear` (`virNetworkIpDefPtr` def)
- static void `virNetworkDNSDefFree` (`virNetworkDNSDefPtr` def)
- **POL New** static void `virTunnelDefFree` (`virTunnelDefPtr` def)
- **POL Mod** void `virNetworkDefFree` (`virNetworkDefPtr` def)
- void `virNetworkObjFree` (`virNetworkObjPtr` net)
- void `virNetworkObjListFree` (`virNetworkObjListPtr` nets)
- int `virNetworkObjAssignDef` (`virNetworkObjPtr` network, const `virNetworkDefPtr` def, bool live)
- `virNetworkObjPtr virNetworkAssignDef` (`virNetworkObjListPtr` nets, const `virNetworkDefPtr` def, bool live)
- int `virNetworkObjSetDefTransient` (`virNetworkObjPtr` network, bool live)
- void `virNetworkObjUnsetDefTransient` (`virNetworkObjPtr` network)
- `virNetworkDefPtr virNetworkObjGetPersistentDef` (`virNetworkObjPtr` network)
- int `virNetworkObjReplacePersistentDef` (`virNetworkObjPtr` network, `virNetworkDefPtr` def)
- `virNetworkDefPtr virNetworkDefCopy` (`virNetworkDefPtr` def, unsigned int flags)
- int `virNetworkConfigChangeSetup` (`virNetworkObjPtr` network, unsigned int flags)
- void `virNetworkRemoveInactive` (`virNetworkObjListPtr` nets, const `virNetworkObjPtr` net)

- [virNetworkIpfDefPtr virNetworkDefGetIpfByIndex](#) (const [virNetworkDefPtr](#) def, int family, size\_t n)
- int [virNetworkIpfDefPrefix](#) (const [virNetworkIpfDefPtr](#) def)
- int [virNetworkIpfDefNetmask](#) (const [virNetworkIpfDefPtr](#) def, [virSocketAddrPtr](#) netmask)
- static int [virNetworkDHCPRangeDefParse](#) (const char \*networkName, [xmlNodePtr](#) node, [virNetworkDHCPRangeDefPtr](#) range)
- static int [virNetworkDHCPHostDefParse](#) (const char \*networkName, [xmlNodePtr](#) node, [virNetworkDHCPHostDefPtr](#) host, bool partialOkay)
- static int [virNetworkDHCPDefParse](#) (const char \*networkName, [virNetworkIpfDefPtr](#) def, [xmlNodePtr](#) node)
- static int [virNetworkDNSHostsDefParseXML](#) ([virNetworkDNSDefPtr](#) def, [xmlNodePtr](#) node)
- static int [virNetworkDNSSrvDefParseXML](#) ([virNetworkDNSDefPtr](#) def, [xmlNodePtr](#) cur, [xmlXPathContextPtr](#) ctxt)
- static int [virNetworkDNSDefParseXML](#) ([virNetworkDNSDefPtr](#) \*dnsdef, [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt)
- static int [virNetworkIPParseXML](#) (const char \*networkName, [virNetworkIpfDefPtr](#) def, [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt)
- static int [virNetworkPortGroupParseXML](#) ([virPortGroupDefPtr](#) def, [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt)
- **POL New** static int [virNetworkTunnelParseXML](#) ([virTunnelDefPtr](#) def, [xmlNodePtr](#) node, [xmlXPathContextPtr](#) ctxt)
- **POL Mod** static [virNetworkDefPtr](#) [virNetworkDefParseXML](#) ([xmlXPathContextPtr](#) ctxt)
- static [virNetworkDefPtr](#) [virNetworkDefParse](#) (const char \*xmlStr, const char \*filename)
- [virNetworkDefPtr](#) [virNetworkDefParseString](#) (const char \*xmlStr)
- [virNetworkDefPtr](#) [virNetworkDefParseFile](#) (const char \*filename)
- [virNetworkDefPtr](#) [virNetworkDefParseNode](#) ([xmlDocPtr](#) xml, [xmlNodePtr](#) root)
- static int [virNetworkDNSDefFormat](#) ([virBufferPtr](#) buf, [virNetworkDNSDefPtr](#) def)
- static int [virNetworkIpfDefFormat](#) ([virBufferPtr](#) buf, const [virNetworkIpfDefPtr](#) def)
- static int [virPortGroupDefFormat](#) ([virBufferPtr](#) buf, const [virPortGroupDefPtr](#) def)
- **POL New** static int [virTunnelDefFormat](#) ([virBufferPtr](#) buf, const [virTunnelDefPtr](#) def)
- **POL Mod** char \* [virNetworkDefFormat](#) (const [virNetworkDefPtr](#) def, unsigned int flags)
- [virPortGroupDefPtr](#) [virPortGroupFindByName](#) ([virNetworkDefPtr](#) net, const char \*portgroup)
- int [virNetworkSaveXML](#) (const char \*configDir, [virNetworkDefPtr](#) def, const char \*xml)
- int [virNetworkSaveConfig](#) (const char \*configDir, [virNetworkDefPtr](#) def)
- int [virNetworkSaveStatus](#) (const char \*statusDir, [virNetworkObjPtr](#) network)
- [virNetworkObjPtr](#) [virNetworkLoadConfig](#) ([virNetworkObjListPtr](#) nets, const char \*configDir, const char \*autostartDir, const char \*name)
- int [virNetworkLoadAllConfigs](#) ([virNetworkObjListPtr](#) nets, const char \*configDir, const char \*autostartDir)
- int [virNetworkDeleteConfig](#) (const char \*configDir, const char \*autostartDir, [virNetworkObjPtr](#) net)
- char \* [virNetworkConfigFile](#) (const char \*dir, const char \*name)
- int [virNetworkBridgeInUse](#) (const [virNetworkObjListPtr](#) nets, const char \*bridge, const char \*skipname)
- char \* [virNetworkAllocateBridge](#) (const [virNetworkObjListPtr](#) nets, const char \*template)
- int [virNetworkSetBridgeName](#) (const [virNetworkObjListPtr](#) nets, [virNetworkDefPtr](#) def, int check\_collision)
- void [virNetworkSetBridgeMacAddr](#) ([virNetworkDefPtr](#) def)
- static void [virNetworkDefUpdateNoSupport](#) ([virNetworkDefPtr](#) def, const char \*section)
- static void [virNetworkDefUpdateUnknownCommand](#) (unsigned int command)
- static int [virNetworkDefUpdateCheckElementName](#) ([virNetworkDefPtr](#) def, [xmlNodePtr](#) node, const char \*section)
- static int [virNetworkDefUpdateBridge](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, [xmlXPathContextPtr](#) ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateDomain](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, [xmlXPathContextPtr](#) ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateIP](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, [xmlXPathContextPtr](#) ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)

- static [virNetworkIpDefPtr](#) [virNetworkIpDefByIndex](#) ([virNetworkDefPtr](#) def, int parentIndex)
- static int [virNetworkDefUpdateIPDHCPHost](#) ([virNetworkDefPtr](#) def, unsigned int command, int parentIndex, xmlXPathContextPtr ctxt, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateIPDHCPRange](#) ([virNetworkDefPtr](#) def, unsigned int command, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateForward](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateForwardInterface](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateForwardPF](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdatePortGroup](#) ([virNetworkDefPtr](#) def, unsigned int command, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateDNSHost](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateDNSTxt](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)
- static int [virNetworkDefUpdateDNSSrv](#) ([virNetworkDefPtr](#) def, unsigned int command ATTRIBUTE\_UNUSED, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE\_UNUSED, unsigned int fflags ATTRIBUTE\_UNUSED)
- **POL New** static int [virNetworkDefUpdateTunnel](#) ([virNetworkDefPtr](#) def, unsigned int command, int parentIndex ATTRIBUTE\_UNUSED, xmlXPathContextPtr ctxt, unsigned int fflags ATTRIBUTE\_UNUSED)
- **POL Mod** static int [virNetworkDefUpdateSection](#) ([virNetworkDefPtr](#) def, unsigned int command, unsigned int section, int parentIndex, const char \*xml, unsigned int flags)
- int [virNetworkObjUpdate](#) ([virNetworkObjPtr](#) network, unsigned int command, unsigned int section, int parentIndex, const char \*xml, unsigned int flags)
- int [virNetworkObjsDuplicate](#) ([virNetworkObjListPtr](#) doms, [virNetworkDefPtr](#) def, unsigned int check\_active)
- void [virNetworkObjLock](#) ([virNetworkObjPtr](#) obj)
- void [virNetworkObjUnlock](#) ([virNetworkObjPtr](#) obj)
- static bool [virNetworkMatch](#) ([virNetworkObjPtr](#) netobj, unsigned int flags)
- int [virNetworkList](#) ([virConnectPtr](#) conn, [virNetworkObjList](#) netobjs, [virNetworkPtr](#) \*\*nets, unsigned int flags)

#### 4.4.1 Macro Definition Documentation

4.4.1.1 `#define MATCH( FLAG ) (flags & (FLAG))`

4.4.1.2 `#define MAX_BRIDGE_ID 256`

4.4.1.3 `#define VIR_FROM_THIS VIR_FROM_NETWORK`

#### 4.4.2 Function Documentation

4.4.2.1 `VIR_ENUM_IMPL ( virNetworkForward , VIR_NETWORK_FORWARD_LAST , VIR_ENUM_IMPL( "none", "nat", "route", "bridge", "private", "vepa", "passthrough", "hostdev" )`

4.4.2.2 `char* virNetworkAllocateBridge ( const virNetworkObjListPtr nets, const char * template )`

4.4.2.3 `virNetworkObjPtr virNetworkAssignDef ( virNetworkObjListPtr nets, const virNetworkDefPtr def, bool live )`

4.4.2.4 `int virNetworkBridgeInUse ( const virNetworkObjListPtr nets, const char * bridge, const char * skipname )`

4.4.2.5 `int virNetworkConfigChangeSetup ( virNetworkObjPtr network, unsigned int flags )`

4.4.2.6 `char* virNetworkConfigFile ( const char * dir, const char * name )`

4.4.2.7 `virNetworkDefPtr virNetworkDefCopy ( virNetworkDefPtr def, unsigned int flags )`

4.4.2.8 **POL Mod** `char* virNetworkDefFormat ( const virNetworkDefPtr def, unsigned int flags )`

virNetworkDefFormat

Parameters

<i>def</i>	pointer to virNetworkDef structure
<i>flags</i>	flags

Display network configuration

POL modification:

- Display new information (bridge type, source bridge, tunnels)

4.4.2.9 **POL Mod** `void virNetworkDefFree ( virNetworkDefPtr def )`

virNetworkDefFree

Parameters

<i>def</i>	pointer to virNetworkDef structure
------------	------------------------------------

Free a virNetworkDef structure.

POL modification:

- Free also `def->source_bridge`, `def->bridge_type`, `def->tunnels`

4.4.2.10 `virNetworkIplDefPtr virNetworkDefGetIplByIndex ( const virNetworkDefPtr def, int family, size_t n )`

4.4.2.11 `static virNetworkDefPtr virNetworkDefParse ( const char * xmlStr, const char * filename )` [static]

4.4.2.12 `virNetworkDefPtr virNetworkDefParseFile ( const char * filename )`

4.4.2.13 `virNetworkDefPtr virNetworkDefParseNode ( xmlDocPtr xml, xmlNodePtr root )`

4.4.2.14 `virNetworkDefPtr virNetworkDefParseString ( const char * xmlStr )`

4.4.2.15 **POL Mod** `static virNetworkDefPtr virNetworkDefParseXML ( xmlXPathContextPtr ctxt )` [static]

virNetworkDefParseXML

Parameters

<i>ctxt</i>	pointer to xmlXPathContext structure
-------------	--------------------------------------

Parse network configuration in XML format

POL modification:



- Parse new information (bridge type, source bridge and tunnels)

- 4.4.2.16 `static int virNetworkDefUpdateBridge ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.17 `static int virNetworkDefUpdateCheckElementName ( virNetworkDefPtr def, xmlNodePtr node, const char * section ) [static]`
- 4.4.2.18 `static int virNetworkDefUpdateDNSHost ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.19 `static int virNetworkDefUpdateDNSSrv ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.20 `static int virNetworkDefUpdateDNSTxt ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.21 `static int virNetworkDefUpdateDomain ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.22 `static int virNetworkDefUpdateForward ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.23 `static int virNetworkDefUpdateForwardInterface ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.24 `static int virNetworkDefUpdateForwardPF ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.25 `static int virNetworkDefUpdateIP ( virNetworkDefPtr def, unsigned int command ATTRIBUTE_UNUSED, int parentIndex ATTRIBUTE_UNUSED, xmlXPathContextPtr ctxt ATTRIBUTE_UNUSED, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.26 `static int virNetworkDefUpdateIPDHCPHost ( virNetworkDefPtr def, unsigned int command, int parentIndex, xmlXPathContextPtr ctxt, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.27 `static int virNetworkDefUpdateIPDHCPRange ( virNetworkDefPtr def, unsigned int command, int parentIndex, xmlXPathContextPtr ctxt, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.28 `static void virNetworkDefUpdateNoSupport ( virNetworkDefPtr def, const char * section ) [static]`
- 4.4.2.29 `static int virNetworkDefUpdatePortGroup ( virNetworkDefPtr def, unsigned int command, int parentIndex, xmlXPathContextPtr ctxt, unsigned int fflags ATTRIBUTE_UNUSED ) [static]`
- 4.4.2.30 **POL Mod** `static int virNetworkDefUpdateSection ( virNetworkDefPtr def, unsigned int command, unsigned int section, int parentIndex, const char * xml, unsigned int flags ) [static]`

virNetworkDefUpdateSection

## Parameters

<i>def</i>	pointer to virNetworkDef structure
<i>command</i>	command
<i>section</i>	section
<i>parentIndex</i>	index of the parent XML node
<i>xml</i>	new XML configuration
<i>flags</i>	flags

Update a network section.

POL modification:

- Add support for tunnels

4.4.2.31 **POL New** static int virNetworkDefUpdateTunnel ( virNetworkDefPtr *def*, unsigned int *command*, int *parentIndex* *ATTRIBUTE\_UNUSED*, xmlXPathContextPtr *ctxt*, unsigned int *fFlags* *ATTRIBUTE\_UNUSED* ) [static]

virNetworkDefUpdateTunnel

## Parameters

<i>def</i>	pointer to virNetworkDef structure
<i>command</i>	command
<i>parentIndex</i>	index of the parent XML node
<i>ctxt</i>	pointer to xmlXPathContext structure
<i>fFlags</i>	flags

Add/remove a tunnel from a network configuration

4.4.2.32 static void virNetworkDefUpdateUnknownCommand ( unsigned int *command* ) [static]

4.4.2.33 int virNetworkDeleteConfig ( const char \* *configDir*, const char \* *autostartDir*, virNetworkObjPtr *net* )

4.4.2.34 static int virNetworkDHCPDefParse ( const char \* *networkName*, virNetworkIpDefPtr *def*, xmlNodePtr *node* ) [static]

4.4.2.35 static void virNetworkDHCPHostDefClear ( virNetworkDHCPHostDefPtr *def* ) [static]

4.4.2.36 static int virNetworkDHCPHostDefParse ( const char \* *networkName*, xmlNodePtr *node*, virNetworkDHCPHostDefPtr *host*, bool *partialOkay* ) [static]

4.4.2.37 static int virNetworkDHCPRangeDefParse ( const char \* *networkName*, xmlNodePtr *node*, virNetworkDHCPRangeDefPtr *range* ) [static]

4.4.2.38 static int virNetworkDNSDefFormat ( virBufferPtr *buf*, virNetworkDNSDefPtr *def* ) [static]

4.4.2.39 static void virNetworkDNSDefFree ( virNetworkDNSDefPtr *def* ) [static]

4.4.2.40 static int virNetworkDNSDefParseXML ( virNetworkDNSDefPtr \* *dnsdef*, xmlNodePtr *node*, xmlXPathContextPtr *ctxt* ) [static]

4.4.2.41 static int virNetworkDNSHostsDefParseXML ( virNetworkDNSDefPtr *def*, xmlNodePtr *node* ) [static]

4.4.2.42 static int virNetworkDNSSrvDefParseXML ( virNetworkDNSDefPtr *def*, xmlNodePtr *cur*, xmlXPathContextPtr *ctxt* ) [static]

- 4.4.2.43 `virNetworkObjPtr virNetworkFindByName ( const virNetworkObjListPtr nets, const char * name )`
- 4.4.2.44 `static void virNetworkForwardIfDefClear ( virNetworkForwardIfDefPtr def ) [static]`
- 4.4.2.45 `static void virNetworkForwardPfDefClear ( virNetworkForwardPfDefPtr def ) [static]`
- 4.4.2.46 `static virNetworkIpDefPtr virNetworkIpDefByIndex ( virNetworkDefPtr def, int parentIndex ) [static]`
- 4.4.2.47 `static void virNetworkIpDefClear ( virNetworkIpDefPtr def ) [static]`
- 4.4.2.48 `static int virNetworkIpDefFormat ( virBufferPtr buf, const virNetworkIpDefPtr def ) [static]`
- 4.4.2.49 `int virNetworkIpDefNetmask ( const virNetworkIpDefPtr def, virSocketAddrPtr netmask )`
- 4.4.2.50 `int virNetworkIpDefPrefix ( const virNetworkIpDefPtr def )`
- 4.4.2.51 `static int virNetworkIPParseXML ( const char * networkName, virNetworkIpDefPtr def, xmlNodePtr node, xmlXPathContextPtr ctxt ) [static]`
- 4.4.2.52 `int virNetworkList ( virConnectPtr conn, virNetworkObjList netobjs, virNetworkPtr ** nets, unsigned int flags )`
- 4.4.2.53 `int virNetworkLoadAllConfigs ( virNetworkObjListPtr nets, const char * configDir, const char * autostartDir )`
- 4.4.2.54 `virNetworkObjPtr virNetworkLoadConfig ( virNetworkObjListPtr nets, const char * configDir, const char * autostartDir, const char * name )`
- 4.4.2.55 `static bool virNetworkMatch ( virNetworkObjPtr netobj, unsigned int flags ) [static]`
- 4.4.2.56 `int virNetworkObjAssignDef ( virNetworkObjPtr network, const virNetworkDefPtr def, bool live )`
- 4.4.2.57 `void virNetworkObjFree ( virNetworkObjPtr net )`
- 4.4.2.58 `virNetworkDefPtr virNetworkObjGetPersistentDef ( virNetworkObjPtr network )`
- 4.4.2.59 `int virNetworkObjsDuplicate ( virNetworkObjListPtr doms, virNetworkDefPtr def, unsigned int check_active )`
- 4.4.2.60 `void virNetworkObjListFree ( virNetworkObjListPtr nets )`
- 4.4.2.61 `void virNetworkObjLock ( virNetworkObjPtr obj )`
- 4.4.2.62 `int virNetworkObjReplacePersistentDef ( virNetworkObjPtr network, virNetworkDefPtr def )`
- 4.4.2.63 `int virNetworkObjSetDefTransient ( virNetworkObjPtr network, bool live )`
- 4.4.2.64 `void virNetworkObjUnlock ( virNetworkObjPtr obj )`
- 4.4.2.65 `void virNetworkObjUnsetDefTransient ( virNetworkObjPtr network )`
- 4.4.2.66 `int virNetworkObjUpdate ( virNetworkObjPtr network, unsigned int command, unsigned int section, int parentIndex, const char * xml, unsigned int flags )`
- 4.4.2.67 `static int virNetworkPortGroupParseXML ( virPortGroupDefPtr def, xmlNodePtr node, xmlXPathContextPtr ctxt ) [static]`
- 4.4.2.68 `void virNetworkRemoveInactive ( virNetworkObjListPtr nets, const virNetworkObjPtr net )`

4.4.2.69 `int virNetworkSaveConfig ( const char * configDir, virNetworkDefPtr def )`

4.4.2.70 `int virNetworkSaveStatus ( const char * statusDir, virNetworkObjPtr network )`

4.4.2.71 `int virNetworkSaveXML ( const char * configDir, virNetworkDefPtr def, const char * xml )`

4.4.2.72 `void virNetworkSetBridgeMacAddr ( virNetworkDefPtr def )`

4.4.2.73 `int virNetworkSetBridgeName ( const virNetworkObjListPtr nets, virNetworkDefPtr def, int check_collision )`

4.4.2.74 **POL New** `static int virNetworkTunnelParseXML ( virTunnelDefPtr def, xmlNodePtr node, xmlXPathContextPtr ctxt ) [static]`

virNetworkTunnelParseXML

Parameters

<i>def</i>	pointer to virTunnelDef structure
<i>node</i>	pointer to xmlNode structure
<i>ctxt</i>	pointer to xmlXPathContext structure

Parse tunnel configuration in XML format

4.4.2.75 `static void virPortGroupDefClear ( virPortGroupDefPtr def ) [static]`

4.4.2.76 `static int virPortGroupDefFormat ( virBufferPtr buf, const virPortGroupDefPtr def ) [static]`

4.4.2.77 `virPortGroupDefPtr virPortGroupFindByName ( virNetworkDefPtr net, const char * portgroup )`

4.4.2.78 **POL New** `static int virTunnelDefFormat ( virBufferPtr buf, const virTunnelDefPtr def ) [static]`

virTunnelDefFormat

Parameters

<i>buf</i>	pointer to virBuffer structure
<i>def</i>	pointer to virTunnelDef structure

Display tunnel configuration

4.4.2.79 **POL New** `static void virTunnelDefFree ( virTunnelDefPtr def ) [static]`

virTunnelDefFree

Parameters

<i>def</i>	pointer to virTunnelDef structure
------------	-----------------------------------

Free a virTunnelDef structure.

## 4.5 src/conf/network\_conf.h File Reference

```
#include <libxml/parser.h>
#include <libxml/tree.h>
#include <libxml/xpath.h>
#include "internal.h"
#include "threads.h"
#include "virsocketaddr.h"
#include "virnetdevbandwidth.h"
#include "virnetdevvportprofile.h"
#include "virnetdevvlan.h"
#include "virmacaddr.h"
#include "device_conf.h"
```

### Data Structures

- struct [\\_virNetworkDHCPRangeDef](#)
- struct [\\_virNetworkDHCPHostDef](#)
- struct [\\_virNetworkDNSTxtRecordsDef](#)
- struct [\\_virNetworkDNSSrvRecordsDef](#)
- struct [\\_virNetworkDNSHostsDef](#)
- struct [\\_virNetworkDNSDef](#)
- struct [\\_virNetworkIpDef](#)
- struct [\\_virNetworkForwardIfDef](#)
- struct [\\_virNetworkForwardPfDef](#)
- struct [\\_virPortGroupDef](#)
- struct [\\_virTunnelDef](#)
- struct [\\_virNetworkDef](#)
- struct [\\_virNetworkObj](#)
- struct [\\_virNetworkObjList](#)

### Macros

- #define [DNS\\_RECORD\\_LENGTH\\_SRV](#) (512 - 30) /\* Limit minus overhead as mentioned in RFC-2782 \*/
- #define [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_FILTERS\\_ACTIVE](#)
- #define [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_FILTERS\\_PERSISTENT](#)
- #define [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_FILTERS\\_AUTOSTART](#)
- #define [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_FILTERS\\_ALL](#)

### Typedefs

- typedef struct [\\_virNetworkDHCPRangeDef](#) [virNetworkDHCPRangeDef](#)
- typedef [virNetworkDHCPRangeDef](#) \* [virNetworkDHCPRangeDefPtr](#)
- typedef struct [\\_virNetworkDHCPHostDef](#) [virNetworkDHCPHostDef](#)
- typedef [virNetworkDHCPHostDef](#) \* [virNetworkDHCPHostDefPtr](#)
- typedef struct [\\_virNetworkDNSTxtRecordsDef](#) [virNetworkDNSTxtRecordsDef](#)
- typedef [virNetworkDNSTxtRecordsDef](#) \* [virNetworkDNSTxtRecordsDefPtr](#)
- typedef struct [\\_virNetworkDNSSrvRecordsDef](#) [virNetworkDNSSrvRecordsDef](#)

- typedef `virNetworkDNSSrvRecordsDef` \* `virNetworkDNSSrvRecordsDefPtr`
- typedef struct `_virNetworkDNSHostsDef` \* `virNetworkDNSHostsDefPtr`
- typedef struct `_virNetworkDNSDef` \* `virNetworkDNSDefPtr`
- typedef struct `_virNetworkIpDef` `virNetworkIpDef`
- typedef `virNetworkIpDef` \* `virNetworkIpDefPtr`
- typedef struct `_virNetworkForwardIfDef` `virNetworkForwardIfDef`
- typedef `virNetworkForwardIfDef` \* `virNetworkForwardIfDefPtr`
- typedef struct `_virNetworkForwardPfDef` `virNetworkForwardPfDef`
- typedef `virNetworkForwardPfDef` \* `virNetworkForwardPfDefPtr`
- typedef struct `_virPortGroupDef` `virPortGroupDef`
- typedef `virPortGroupDef` \* `virPortGroupDefPtr`
- typedef struct `_virTunnelDef` `virTunnelDef`
- typedef `virTunnelDef` \* `virTunnelDefPtr`
- typedef struct `_virNetworkDef` `virNetworkDef`
- typedef `virNetworkDef` \* `virNetworkDefPtr`
- typedef struct `_virNetworkObj` `virNetworkObj`
- typedef `virNetworkObj` \* `virNetworkObjPtr`
- typedef struct `_virNetworkObjList` `virNetworkObjList`
- typedef `virNetworkObjList` \* `virNetworkObjListPtr`

## Enumerations

- enum `virNetworkForwardType` {  
`VIR_NETWORK_FORWARD_NONE` = 0, `VIR_NETWORK_FORWARD_NAT`, `VIR_NETWORK_FORWARD_ROUTE`, `VIR_NETWORK_FORWARD_BRIDGE`,  
`VIR_NETWORK_FORWARD_PRIVATE`, `VIR_NETWORK_FORWARD_VEPA`, `VIR_NETWORK_FORWARD_PASSTHROUGH`, `VIR_NETWORK_FORWARD_HOSTDEV`,  
`VIR_NETWORK_FORWARD_LAST` }
- enum `virNetworkForwardHostdevDeviceType` { `VIR_NETWORK_FORWARD_HOSTDEV_DEVICE_NONE` = 0, `VIR_NETWORK_FORWARD_HOSTDEV_DEVICE_PCI`, `VIR_NETWORK_FORWARD_HOSTDEV_DEVICE_NETDEV`, `VIR_NETWORK_FORWARD_HOSTDEV_DEVICE_LAST` }

## Functions

- static int `virNetworkObjsActive` (const `virNetworkObjPtr` net)
- `virNetworkObjPtr` `virNetworkFindByUUID` (const `virNetworkObjListPtr` nets, const unsigned char \*uuid)
- `virNetworkObjPtr` `virNetworkFindByName` (const `virNetworkObjListPtr` nets, const char \*name)
- POL Mod void `virNetworkDefFree` (`virNetworkDefPtr` def)
- void `virNetworkObjFree` (`virNetworkObjPtr` net)
- void `virNetworkObjListFree` (`virNetworkObjListPtr` vms)
- `virNetworkObjPtr` `virNetworkAssignDef` (`virNetworkObjListPtr` nets, const `virNetworkDefPtr` def, bool live)
- int `virNetworkObjAssignDef` (`virNetworkObjPtr` network, const `virNetworkDefPtr` def, bool live)
- int `virNetworkObjSetDefTransient` (`virNetworkObjPtr` network, bool live)
- void `virNetworkObjUnsetDefTransient` (`virNetworkObjPtr` network)
- `virNetworkDefPtr` `virNetworkObjGetPersistentDef` (`virNetworkObjPtr` network)
- int `virNetworkObjReplacePersistentDef` (`virNetworkObjPtr` network, `virNetworkDefPtr` def)
- `virNetworkDefPtr` `virNetworkDefCopy` (`virNetworkDefPtr` def, unsigned int flags)
- int `virNetworkConfigChangeSetup` (`virNetworkObjPtr` dom, unsigned int flags)
- void `virNetworkRemoveInactive` (`virNetworkObjListPtr` nets, const `virNetworkObjPtr` net)
- `virNetworkDefPtr` `virNetworkDefParseString` (const char \*xmlStr)

- `virNetworkDefPtr virNetworkDefParseFile` (const char \*filename)
- `virNetworkDefPtr virNetworkDefParseNode` (xmlDocPtr xml, xmlNodePtr root)
- **POL Mod** `char * virNetworkDefFormat` (const `virNetworkDefPtr` def, unsigned int flags)
- static const char \* `virNetworkDefForwardIpf` (const `virNetworkDefPtr` def, size\_t n)
- `virPortGroupDefPtr virPortGroupFindByName` (`virNetworkDefPtr` net, const char \*portgroup)
- `virNetworkIpfDefPtr virNetworkDefGetIpfByIndex` (const `virNetworkDefPtr` def, int family, size\_t n)
- int `virNetworkIpfDefPrefix` (const `virNetworkIpfDefPtr` def)
- int `virNetworkIpfDefNetmask` (const `virNetworkIpfDefPtr` def, `virSocketAddrPtr` netmask)
- int `virNetworkSaveXML` (const char \*configDir, `virNetworkDefPtr` def, const char \*xml)
- int `virNetworkSaveConfig` (const char \*configDir, `virNetworkDefPtr` def)
- int `virNetworkSaveStatus` (const char \*statusDir, `virNetworkObjPtr` net) ATTRIBUTE\_RETURN\_CHECK
- `virNetworkObjPtr virNetworkLoadConfig` (`virNetworkObjListPtr` nets, const char \*configDir, const char \*autostartDir, const char \*file)
- int `virNetworkLoadAllConfigs` (`virNetworkObjListPtr` nets, const char \*configDir, const char \*autostartDir)
- int `virNetworkDeleteConfig` (const char \*configDir, const char \*autostartDir, `virNetworkObjPtr` net)
- char \* `virNetworkConfigFile` (const char \*dir, const char \*name)
- int `virNetworkBridgeInUse` (const `virNetworkObjListPtr` nets, const char \*bridge, const char \*skipname)
- char \* `virNetworkAllocateBridge` (const `virNetworkObjListPtr` nets, const char \*template)
- int `virNetworkSetBridgeName` (const `virNetworkObjListPtr` nets, `virNetworkDefPtr` def, int check\_collision)
- void `virNetworkSetBridgeMacAddr` (`virNetworkDefPtr` def)
- int `virNetworkObjUpdate` (`virNetworkObjPtr` obj, unsigned int command, unsigned int section, int parentIndex, const char \*xml, unsigned int flags)
- int `virNetworkObjIsDuplicate` (`virNetworkObjListPtr` doms, `virNetworkDefPtr` def, unsigned int check\_active)
- void `virNetworkObjLock` (`virNetworkObjPtr` obj)
- void `virNetworkObjUnlock` (`virNetworkObjPtr` obj)
- int `virNetworkList` (`virConnectPtr` conn, `virNetworkObjList` netobjs, `virNetworkPtr` \*\*nets, unsigned int flags)

#### 4.5.1 Macro Definition Documentation

4.5.1.1 `#define DNS_RECORD_LENGTH_SRV (512 - 30) /* Limit minus overhead as mentioned in RFC-2782 */`

4.5.1.2 `#define VIR_CONNECT_LIST_NETWORKS_FILTERS_ACTIVE`

**Value:**

```
(VIR_CONNECT_LIST_NETWORKS_ACTIVE | \
 VIR_CONNECT_LIST_NETWORKS_INACTIVE)
```

4.5.1.3 `#define VIR_CONNECT_LIST_NETWORKS_FILTERS_ALL`

**Value:**

```
(VIR_CONNECT_LIST_NETWORKS_FILTERS_ACTIVE | \
 VIR_CONNECT_LIST_NETWORKS_FILTERS_PERSISTENT | \
 VIR_CONNECT_LIST_NETWORKS_FILTERS_AUTOSTART)
```

4.5.1.4 `#define VIR_CONNECT_LIST_NETWORKS_FILTERS_AUTOSTART`

**Value:**

```
(VIR_CONNECT_LIST_NETWORKS_AUTOSTART | \
 VIR_CONNECT_LIST_NETWORKS_NO_AUTOSTART)
```

4.5.1.5 `#define VIR_CONNECT_LIST_NETWORKS_FILTERS_PERSISTENT`

**Value:**

```
(VIR_CONNECT_LIST_NETWORKS_PERSISTENT | \
    VIR_CONNECT_LIST_NETWORKS_TRANSIENT)
```

## 4.5.2 Typedef Documentation

4.5.2.1 `typedef struct _virNetworkDef virNetworkDef`

4.5.2.2 `typedef virNetworkDef* virNetworkDefPtr`

4.5.2.3 `typedef struct _virNetworkDHCPHostDef virNetworkDHCPHostDef`

4.5.2.4 `typedef virNetworkDHCPHostDef* virNetworkDHCPHostDefPtr`

4.5.2.5 `typedef struct _virNetworkDHCPRangeDef virNetworkDHCPRangeDef`

4.5.2.6 `typedef virNetworkDHCPRangeDef* virNetworkDHCPRangeDefPtr`

4.5.2.7 `typedef struct _virNetworkDNSDef* virNetworkDNSDefPtr`

4.5.2.8 `typedef struct _virNetworkDNSHostsDef* virNetworkDNSHostsDefPtr`

4.5.2.9 `typedef struct _virNetworkDNSSrvRecordsDef virNetworkDNSSrvRecordsDef`

4.5.2.10 `typedef virNetworkDNSSrvRecordsDef* virNetworkDNSSrvRecordsDefPtr`

4.5.2.11 `typedef struct _virNetworkDNSTxtRecordsDef virNetworkDNSTxtRecordsDef`

4.5.2.12 `typedef virNetworkDNSTxtRecordsDef* virNetworkDNSTxtRecordsDefPtr`

4.5.2.13 `typedef struct _virNetworkForwardIfDef virNetworkForwardIfDef`

4.5.2.14 `typedef virNetworkForwardIfDef* virNetworkForwardIfDefPtr`

4.5.2.15 `typedef struct _virNetworkForwardPfDef virNetworkForwardPfDef`

4.5.2.16 `typedef virNetworkForwardPfDef* virNetworkForwardPfDefPtr`

4.5.2.17 `typedef struct _virNetworkIpfDef virNetworkIpfDef`

4.5.2.18 `typedef virNetworkIpfDef* virNetworkIpfDefPtr`

4.5.2.19 `typedef struct _virNetworkObj virNetworkObj`

4.5.2.20 `typedef struct _virNetworkObjList virNetworkObjList`

4.5.2.21 `typedef virNetworkObjList* virNetworkObjListPtr`

4.5.2.22 `typedef virNetworkObj* virNetworkObjPtr`

4.5.2.23 `typedef struct _virPortGroupDef virPortGroupDef`

4.5.2.24 `typedef virPortGroupDef* virPortGroupDefPtr`



4.5.2.25 typedef struct \_virTunnelDef virTunnelDef

4.5.2.26 typedef virTunnelDef\* virTunnelDefPtr

### 4.5.3 Enumeration Type Documentation

4.5.3.1 enum virNetworkForwardHostdevDeviceType

Enumerator

***VIR\_NETWORK\_FORWARD\_HOSTDEV\_DEVICE\_NONE***  
***VIR\_NETWORK\_FORWARD\_HOSTDEV\_DEVICE\_PCI***  
***VIR\_NETWORK\_FORWARD\_HOSTDEV\_DEVICE\_NETDEV***  
***VIR\_NETWORK\_FORWARD\_HOSTDEV\_DEVICE\_LAST***

4.5.3.2 enum virNetworkForwardType

Enumerator

***VIR\_NETWORK\_FORWARD\_NONE***  
***VIR\_NETWORK\_FORWARD\_NAT***  
***VIR\_NETWORK\_FORWARD\_ROUTE***  
***VIR\_NETWORK\_FORWARD\_BRIDGE***  
***VIR\_NETWORK\_FORWARD\_PRIVATE***  
***VIR\_NETWORK\_FORWARD\_VEPA***  
***VIR\_NETWORK\_FORWARD\_PASSTHROUGH***  
***VIR\_NETWORK\_FORWARD\_HOSTDEV***  
***VIR\_NETWORK\_FORWARD\_LAST***

### 4.5.4 Function Documentation

4.5.4.1 char\* virNetworkAllocateBridge ( const virNetworkObjListPtr *nets*, const char \* *template* )

4.5.4.2 virNetworkObjPtr virNetworkAssignDef ( virNetworkObjListPtr *nets*, const virNetworkDefPtr *def*, bool *live* )

4.5.4.3 int virNetworkBridgeInUse ( const virNetworkObjListPtr *nets*, const char \* *bridge*, const char \* *skipname* )

4.5.4.4 int virNetworkConfigChangeSetup ( virNetworkObjPtr *dom*, unsigned int *flags* )

4.5.4.5 char\* virNetworkConfigFile ( const char \* *dir*, const char \* *name* )

4.5.4.6 virNetworkDefPtr virNetworkDefCopy ( virNetworkDefPtr *def*, unsigned int *flags* )

4.5.4.7 **POL Mod** char\* virNetworkDefFormat ( const virNetworkDefPtr *def*, unsigned int *flags* )

virNetworkDefFormat

Parameters

<i>def</i>	pointer to virNetworkDef structure
<i>flags</i>	flags

Display network configuration

POL modification:

- Display new information (bridge type, source bridge, tunnels)

4.5.4.8 `static const char* virNetworkDefForwardIf ( const virNetworkDefPtr def, size_t n )` `[inline],[static]`

4.5.4.9 **POL Mod** `void virNetworkDefFree ( virNetworkDefPtr def )`

virNetworkDefFree

Parameters

<i>def</i>	pointer to virNetworkDef structure
------------	------------------------------------

Free a virNetworkDef structure.

POL modification:

- Free also `def->source_bridge`, `def->bridge_type`, `def->tunnels`

4.5.4.10 `virNetworkIplDefPtr virNetworkDefGetIplByIndex ( const virNetworkDefPtr def, int family, size_t n )`

4.5.4.11 `virNetworkDefPtr virNetworkDefParseFile ( const char * filename )`

4.5.4.12 `virNetworkDefPtr virNetworkDefParseNode ( xmlDocPtr xml, xmlNodePtr root )`

4.5.4.13 `virNetworkDefPtr virNetworkDefParseString ( const char * xmlStr )`

4.5.4.14 `int virNetworkDeleteConfig ( const char * configDir, const char * autostartDir, virNetworkObjPtr net )`

4.5.4.15 `virNetworkObjPtr virNetworkFindByName ( const virNetworkObjListPtr nets, const char * name )`

4.5.4.16 `virNetworkObjPtr virNetworkFindByUUID ( const virNetworkObjListPtr nets, const unsigned char * uuid )`

4.5.4.17 `int virNetworkIplDefNetmask ( const virNetworkIplDefPtr def, virSocketAddrPtr netmask )`

4.5.4.18 `int virNetworkIplDefPrefix ( const virNetworkIplDefPtr def )`

4.5.4.19 `int virNetworkList ( virConnectPtr conn, virNetworkObjList netobjs, virNetworkPtr ** nets, unsigned int flags )`

4.5.4.20 `int virNetworkLoadAllConfigs ( virNetworkObjListPtr nets, const char * configDir, const char * autostartDir )`

4.5.4.21 `virNetworkObjPtr virNetworkLoadConfig ( virNetworkObjListPtr nets, const char * configDir, const char * autostartDir, const char * file )`

4.5.4.22 `int virNetworkObjAssignDef ( virNetworkObjPtr network, const virNetworkDefPtr def, bool live )`

4.5.4.23 `void virNetworkObjFree ( virNetworkObjPtr net )`

4.5.4.24 `virNetworkDefPtr virNetworkObjGetPersistentDef ( virNetworkObjPtr network )`

4.5.4.25 `static int virNetworkObjsActive ( const virNetworkObjPtr net )` `[inline],[static]`

4.5.4.26 `int virNetworkObjsDuplicate ( virNetworkObjListPtr doms, virNetworkDefPtr def, unsigned int check_active )`

- 4.5.4.27 void virNetworkObjListFree ( virNetworkObjListPtr *vms* )
- 4.5.4.28 void virNetworkObjLock ( virNetworkObjPtr *obj* )
- 4.5.4.29 int virNetworkObjReplacePersistentDef ( virNetworkObjPtr *network*, virNetworkDefPtr *def* )
- 4.5.4.30 int virNetworkObjSetDefTransient ( virNetworkObjPtr *network*, bool *live* )
- 4.5.4.31 void virNetworkObjUnlock ( virNetworkObjPtr *obj* )
- 4.5.4.32 void virNetworkObjUnsetDefTransient ( virNetworkObjPtr *network* )
- 4.5.4.33 int virNetworkObjUpdate ( virNetworkObjPtr *obj*, unsigned int *command*, unsigned int *section*, int *parentIndex*, const char \* *xml*, unsigned int *flags* )
- 4.5.4.34 void virNetworkRemovelInactive ( virNetworkObjListPtr *nets*, const virNetworkObjPtr *net* )
- 4.5.4.35 int virNetworkSaveConfig ( const char \* *configDir*, virNetworkDefPtr *def* )
- 4.5.4.36 int virNetworkSaveStatus ( const char \* *statusDir*, virNetworkObjPtr *net* )
- 4.5.4.37 int virNetworkSaveXML ( const char \* *configDir*, virNetworkDefPtr *def*, const char \* *xml* )
- 4.5.4.38 void virNetworkSetBridgeMacAddr ( virNetworkDefPtr *def* )
- 4.5.4.39 int virNetworkSetBridgeName ( const virNetworkObjListPtr *nets*, virNetworkDefPtr *def*, int *check\_collision* )
- 4.5.4.40 virPortGroupDefPtr virPortGroupFindByName ( virNetworkDefPtr *net*, const char \* *portgroup* )

## 4.6 src/driver.h File Reference

```
#include "config.h"
#include <unistd.h>
#include "internal.h"
#include "viruri.h"
```

### Data Structures

- struct [\\_virDriver](#)
- struct [\\_virNetworkDriver](#)
- struct [\\_virInterfaceDriver](#)
- struct [\\_virStorageDriver](#)
- struct [\\_virDeviceMonitor](#)
- struct [\\_virSecretDriver](#)
- struct [\\_virStreamDriver](#)
- struct [\\_virNWFilterDriver](#)

### Macros

- #define [VIR\\_DRV\\_SUPPORTS\\_FEATURE](#)(drv, conn, feature)

## Typedefs

- typedef [virDrvOpenStatus](#)(\* [virDrvOpen](#) )(virConnectPtr conn, [virConnectAuthPtr](#) auth, unsigned int flags)
- typedef int(\* [virDrvClose](#) )(virConnectPtr conn)
- typedef int(\* [virDrvDrvSupportsFeature](#) )(virConnectPtr conn, int feature)
- typedef const char \*(\* [virDrvGetType](#) )(virConnectPtr conn)
- typedef int(\* [virDrvGetVersion](#) )(virConnectPtr conn, unsigned long \*hvVer)
- typedef int(\* [virDrvGetLibVersion](#) )(virConnectPtr conn, unsigned long \*libVer)
- typedef char \*(\* [virDrvGetHostname](#) )(virConnectPtr conn)
- typedef char \*(\* [virDrvGetURI](#) )(virConnectPtr conn)
- typedef char \*(\* [virDrvGetSysinfo](#) )(virConnectPtr conn, unsigned int flags)
- typedef int(\* [virDrvGetMaxVcpus](#) )(virConnectPtr conn, const char \*type)
- typedef int(\* [virDrvNodeGetInfo](#) )(virConnectPtr conn, [virNodeInfoPtr](#) info)
- typedef char \*(\* [virDrvGetCapabilities](#) )(virConnectPtr conn)
- typedef int(\* [virDrvListDomains](#) )(virConnectPtr conn, int \*ids, int maxids)
- typedef int(\* [virDrvNumOfDomains](#) )(virConnectPtr conn)
- typedef [virDomainPtr](#)(\* [virDrvDomainCreateXML](#) )(virConnectPtr conn, const char \*xmlDesc, unsigned int flags)
- typedef [virDomainPtr](#)(\* [virDrvDomainLookupByID](#) )(virConnectPtr conn, int id)
- typedef [virDomainPtr](#)(\* [virDrvDomainLookupByUUID](#) )(virConnectPtr conn, const unsigned char \*uuid)
- typedef [virDomainPtr](#)(\* [virDrvDomainLookupByName](#) )(virConnectPtr conn, const char \*name)
- typedef int(\* [virDrvDomainSuspend](#) )(virDomainPtr domain)
- typedef int(\* [virDrvDomainResume](#) )(virDomainPtr domain)
- typedef int(\* [virDrvDomainPMSuspendForDuration](#) )(virDomainPtr, unsigned int target, unsigned long long duration, unsigned int flags)
- typedef int(\* [virDrvDomainPMWakeup](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainShutdown](#) )(virDomainPtr domain)
- typedef int(\* [virDrvDomainReboot](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainReset](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainDestroy](#) )(virDomainPtr domain)
- typedef int(\* [virDrvDomainDestroyFlags](#) )(virDomainPtr domain, unsigned int flags)
- typedef char \*(\* [virDrvDomainGetOSType](#) )(virDomainPtr domain)
- typedef char \*(\* [virDrvDomainGetHostname](#) )(virDomainPtr domain, unsigned int flags)
- typedef unsigned long long(\* [virDrvDomainGetMaxMemory](#) )(virDomainPtr domain)
- typedef int(\* [virDrvDomainSetMaxMemory](#) )(virDomainPtr domain, unsigned long memory)
- typedef int(\* [virDrvDomainSetMemory](#) )(virDomainPtr domain, unsigned long memory)
- typedef int(\* [virDrvDomainSetMemoryFlags](#) )(virDomainPtr domain, unsigned long memory, unsigned int flags)
- typedef int(\* [virDrvDomainSetMemoryParameters](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvDomainGetMemoryParameters](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- typedef int(\* [virDrvDomainSetNumaParameters](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvDomainGetNumaParameters](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- typedef int(\* [virDrvDomainSetBlkioParameters](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvDomainGetBlkioParameters](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- typedef int(\* [virDrvDomainGetInfo](#) )(virDomainPtr domain, [virDomainInfoPtr](#) info)
- typedef int(\* [virDrvDomainGetState](#) )(virDomainPtr domain, int \*state, int \*reason, unsigned int flags)
- typedef int(\* [virDrvDomainGetControllInfo](#) )(virDomainPtr domain, [virDomainControllInfoPtr](#) info, unsigned int flags)
- typedef int(\* [virDrvDomainSave](#) )(virDomainPtr domain, const char \*to)

- typedef int(\* [virDrvDomainSaveFlags](#) )(virDomainPtr domain, const char \*to, const char \*dxml, unsigned int flags)
- typedef int(\* [virDrvDomainRestore](#) )(virConnectPtr conn, const char \*from)
- typedef int(\* [virDrvDomainRestoreFlags](#) )(virConnectPtr conn, const char \*from, const char \*dxml, unsigned int flags)
- typedef char \*(\* [virDrvDomainSaveImageGetXMLDesc](#) )(virConnectPtr conn, const char \*file, unsigned int flags)
- typedef int(\* [virDrvDomainSaveImageDefineXML](#) )(virConnectPtr conn, const char \*file, const char \*dxml, unsigned int flags)
- typedef int(\* [virDrvDomainCoreDump](#) )(virDomainPtr domain, const char \*to, unsigned int flags)
- typedef char \*(\* [virDrvDomainScreenshot](#) )(virDomainPtr domain, virStreamPtr stream, unsigned int screen, unsigned int flags)
- typedef char \*(\* [virDrvDomainGetXMLDesc](#) )(virDomainPtr dom, unsigned int flags)
- typedef char \*(\* [virDrvConnectDomainXMLFromNative](#) )(virConnectPtr conn, const char \*nativeFormat, const char \*nativeConfig, unsigned int flags)
- typedef char \*(\* [virDrvConnectDomainXMLToNative](#) )(virConnectPtr conn, const char \*nativeFormat, const char \*domainXml, unsigned int flags)
- typedef int(\* [virDrvListDefinedDomains](#) )(virConnectPtr conn, char \*\*const names, int maxnames)
- typedef int(\* [virDrvListAllDomains](#) )(virConnectPtr conn, virDomainPtr \*\*domains, unsigned int flags)
- typedef int(\* [virDrvNumOfDefinedDomains](#) )(virConnectPtr conn)
- typedef int(\* [virDrvDomainCreate](#) )(virDomainPtr dom)
- typedef int(\* [virDrvDomainCreateWithFlags](#) )(virDomainPtr dom, unsigned int flags)
- typedef virDomainPtr(\* [virDrvDomainDefineXML](#) )(virConnectPtr conn, const char \*xml)
- typedef int(\* [virDrvDomainUndefine](#) )(virDomainPtr dom)
- typedef int(\* [virDrvDomainUndefineFlags](#) )(virDomainPtr dom, unsigned int flags)
- typedef int(\* [virDrvDomainSetVcpus](#) )(virDomainPtr domain, unsigned int nvcpus)
- typedef int(\* [virDrvDomainSetVcpusFlags](#) )(virDomainPtr domain, unsigned int nvcpus, unsigned int flags)
- typedef int(\* [virDrvDomainGetVcpusFlags](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainPinVcpu](#) )(virDomainPtr domain, unsigned int vcpu, unsigned char \*cpumap, int maplen)
- typedef int(\* [virDrvDomainPinVcpuFlags](#) )(virDomainPtr domain, unsigned int vcpu, unsigned char \*cpumap, int maplen, unsigned int flags)
- typedef int(\* [virDrvDomainGetVcpuPinInfo](#) )(virDomainPtr domain, int ncumaps, unsigned char \*cpumaps, int maplen, unsigned int flags)
- typedef int(\* [virDrvDomainPinEmulator](#) )(virDomainPtr domain, unsigned char \*cpumap, int maplen, unsigned int flags)
- typedef int(\* [virDrvDomainGetEmulatorPinInfo](#) )(virDomainPtr domain, unsigned char \*cpumaps, int maplen, unsigned int flags)
- typedef int(\* [virDrvDomainGetVcpus](#) )(virDomainPtr domain, virVcpuInfoPtr info, int maxinfo, unsigned char \*cpumaps, int maplen)
- typedef int(\* [virDrvDomainGetMaxVcpus](#) )(virDomainPtr domain)
- typedef int(\* [virDrvDomainGetSecurityLabel](#) )(virDomainPtr domain, virSecurityLabelPtr seclabel)
- typedef int(\* [virDrvDomainGetSecurityLabelList](#) )(virDomainPtr domain, virSecurityLabelPtr \*seclabels)
- typedef int(\* [virDrvNodeGetSecurityModel](#) )(virConnectPtr conn, virSecurityModelPtr secmodel)
- typedef int(\* [virDrvDomainAttachDevice](#) )(virDomainPtr domain, const char \*xml)
- typedef int(\* [virDrvDomainAttachDeviceFlags](#) )(virDomainPtr domain, const char \*xml, unsigned int flags)
- typedef int(\* [virDrvDomainDetachDevice](#) )(virDomainPtr domain, const char \*xml)
- typedef int(\* [virDrvDomainDetachDeviceFlags](#) )(virDomainPtr domain, const char \*xml, unsigned int flags)
- typedef int(\* [virDrvDomainUpdateDeviceFlags](#) )(virDomainPtr domain, const char \*xml, unsigned int flags)
- typedef int(\* [virDrvDomainGetAutostart](#) )(virDomainPtr domain, int \*autostart)
- typedef int(\* [virDrvDomainSetAutostart](#) )(virDomainPtr domain, int autostart)
- typedef char \*(\* [virDrvDomainGetSchedulerType](#) )(virDomainPtr domain, int \*nparams)
- typedef int(\* [virDrvDomainGetSchedulerParameters](#) )(virDomainPtr domain, virTypedParameterPtr params, int \*nparams)
- typedef int(\* [virDrvDomainGetSchedulerParametersFlags](#) )(virDomainPtr domain, virTypedParameterPtr params, int \*nparams, unsigned int flags)

- typedef int(\* [virDrvDomainSetSchedulerParameters](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, int nparams)
- typedef int(\* [virDrvDomainSetSchedulerParametersFlags](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvDomainBlockStats](#) )(virDomainPtr domain, const char \*path, struct [\\_virDomainBlockStats](#) \*stats)
- typedef int(\* [virDrvDomainBlockStatsFlags](#) )(virDomainPtr domain, const char \*path, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- typedef int(\* [virDrvDomainInterfaceStats](#) )(virDomainPtr domain, const char \*path, struct [\\_virDomainInterfaceStats](#) \*stats)
- typedef int(\* [virDrvDomainSetInterfaceParameters](#) )(virDomainPtr dom, const char \*device, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvDomainGetInterfaceParameters](#) )(virDomainPtr dom, const char \*device, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- typedef int(\* [virDrvDomainMemoryStats](#) )(virDomainPtr domain, struct [\\_virDomainMemoryStat](#) \*stats, unsigned int nr\_stats, unsigned int flags)
- typedef int(\* [virDrvDomainBlockPeek](#) )(virDomainPtr domain, const char \*path, unsigned long long offset, size\_t size, void \*buffer, unsigned int flags)
- typedef int(\* [virDrvDomainBlockResize](#) )(virDomainPtr domain, const char \*path, unsigned long long size, unsigned int flags)
- typedef int(\* [virDrvDomainMemoryPeek](#) )(virDomainPtr domain, unsigned long long start, size\_t size, void \*buffer, unsigned int flags)
- typedef int(\* [virDrvDomainGetBlockInfo](#) )(virDomainPtr domain, const char \*path, [virDomainBlockInfoPtr](#) info, unsigned int flags)
- typedef int(\* [virDrvDomainMigratePrepare](#) )(virConnectPtr dconn, char \*\*cookie, int \*cookieLen, const char \*uri\_in, char \*\*uri\_out, unsigned long flags, const char \*dname, unsigned long resource)
- typedef int(\* [virDrvDomainMigratePerform](#) )(virDomainPtr domain, const char \*cookie, int cookieLen, const char \*uri, unsigned long flags, const char \*dname, unsigned long resource)
- typedef [virDomainPtr](#)(\* [virDrvDomainMigrateFinish](#) )(virConnectPtr dconn, const char \*dname, const char \*cookie, int cookieLen, const char \*uri, unsigned long flags)
- typedef struct [\\_virDriver](#) virDriver
- typedef [virDriver](#) \* [virDriverPtr](#)
- typedef int(\* [virDrvNodeGetCPUStats](#) )(virConnectPtr conn, int cpuNum, [virNodeCPUStatsPtr](#) params, int \*nparams, unsigned int flags)
- typedef int(\* [virDrvNodeGetMemoryStats](#) )(virConnectPtr conn, int cellNum, [virNodeMemoryStatsPtr](#) params, int \*nparams, unsigned int flags)
- typedef int(\* [virDrvNodeGetCellsFreeMemory](#) )(virConnectPtr conn, unsigned long long \*freeMems, int startCell, int maxCells)
- typedef unsigned long long(\* [virDrvNodeGetFreeMemory](#) )(virConnectPtr conn)
- typedef int(\* [virDrvDomainEventRegister](#) )(virConnectPtr conn, [virConnectDomainEventCallback](#) cb, void \*opaque, [virFreeCallback](#) freeCb)
- typedef int(\* [virDrvDomainEventDeregister](#) )(virConnectPtr conn, [virConnectDomainEventCallback](#) cb)
- typedef int(\* [virDrvDomainMigratePrepare2](#) )(virConnectPtr dconn, char \*\*cookie, int \*cookieLen, const char \*uri\_in, char \*\*uri\_out, unsigned long flags, const char \*dname, unsigned long resource, const char \*dom\_xml)
- typedef [virDomainPtr](#)(\* [virDrvDomainMigrateFinish2](#) )(virConnectPtr dconn, const char \*dname, const char \*cookie, int cookieLen, const char \*uri, unsigned long flags, int retcode)
- typedef int(\* [virDrvNodeDeviceDetach](#) )(virNodeDevicePtr dev)
- typedef int(\* [virDrvNodeDeviceReAttach](#) )(virNodeDevicePtr dev)
- typedef int(\* [virDrvNodeDeviceReset](#) )(virNodeDevicePtr dev)
- typedef int(\* [virDrvDomainMigratePrepareTunnel](#) )(virConnectPtr dconn, [virStreamPtr](#) st, unsigned long flags, const char \*dname, unsigned long resource, const char \*dom\_xml)
- typedef int(\* [virDrvConnectIsEncrypted](#) )(virConnectPtr conn)
- typedef int(\* [virDrvConnectIsSecure](#) )(virConnectPtr conn)
- typedef int(\* [virDrvConnectIsAlive](#) )(virConnectPtr conn)
- typedef int(\* [virDrvDomainsIsActive](#) )(virDomainPtr dom)



- typedef int(\* [virDrvDomainsIsPersistent](#) )(virDomainPtr dom)
- typedef int(\* [virDrvDomainsIsUpdated](#) )(virDomainPtr dom)
- typedef int(\* [virDrvCompareCPU](#) )(virConnectPtr conn, const char \*cpu, unsigned int flags)
- typedef char \*(\* [virDrvBaselineCPU](#) )(virConnectPtr conn, const char \*\*xmlCPUs, unsigned int ncpus, unsigned int flags)
- typedef int(\* [virDrvDomainGetJobInfo](#) )(virDomainPtr domain, [virDomainJobInfoPtr](#) info)
- typedef int(\* [virDrvDomainAbortJob](#) )(virDomainPtr domain)
- typedef int(\* [virDrvDomainMigrateSetMaxDowntime](#) )(virDomainPtr domain, unsigned long long downtime, unsigned int flags)
- typedef int(\* [virDrvDomainMigrateSetMaxSpeed](#) )(virDomainPtr domain, unsigned long bandwidth, unsigned int flags)
- typedef int(\* [virDrvDomainMigrateGetMaxSpeed](#) )(virDomainPtr domain, unsigned long \*bandwidth, unsigned int flags)
- typedef int(\* [virDrvDomainEventRegisterAny](#) )(virConnectPtr conn, [virDomainPtr](#) dom, int eventID, [virConnectDomainEventGenericCallback](#) cb, void \*opaque, [virFreeCallback](#) freecb)
- typedef int(\* [virDrvDomainEventDeregisterAny](#) )(virConnectPtr conn, int callbackID)
- typedef int(\* [virDrvDomainManagedSave](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainHasManagedSaveImage](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainManagedSaveRemove](#) )(virDomainPtr domain, unsigned int flags)
- typedef [virDomainSnapshotPtr](#)(\* [virDrvDomainSnapshotCreateXML](#) )(virDomainPtr domain, const char \*xmlDesc, unsigned int flags)
- typedef char \*(\* [virDrvDomainSnapshotGetXMLDesc](#) )(virDomainSnapshotPtr snapshot, unsigned int flags)
- typedef int(\* [virDrvDomainSnapshotNum](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainSnapshotListNames](#) )(virDomainPtr domain, char \*\*names, int nameslen, unsigned int flags)
- typedef int(\* [virDrvDomainListAllSnapshots](#) )(virDomainPtr domain, [virDomainSnapshotPtr](#) \*\*snaps, unsigned int flags)
- typedef int(\* [virDrvDomainSnapshotNumChildren](#) )(virDomainSnapshotPtr snapshot, unsigned int flags)
- typedef int(\* [virDrvDomainSnapshotListChildrenNames](#) )(virDomainSnapshotPtr snapshot, char \*\*names, int nameslen, unsigned int flags)
- typedef int(\* [virDrvDomainSnapshotListAllChildren](#) )(virDomainSnapshotPtr snapshot, [virDomainSnapshotPtr](#) \*\*snaps, unsigned int flags)
- typedef [virDomainSnapshotPtr](#)(\* [virDrvDomainSnapshotLookupByName](#) )(virDomainPtr domain, const char \*name, unsigned int flags)
- typedef int(\* [virDrvDomainHasCurrentSnapshot](#) )(virDomainPtr domain, unsigned int flags)
- typedef [virDomainSnapshotPtr](#)(\* [virDrvDomainSnapshotGetParent](#) )(virDomainSnapshotPtr snapshot, unsigned int flags)
- typedef [virDomainSnapshotPtr](#)(\* [virDrvDomainSnapshotCurrent](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainSnapshotIsCurrent](#) )(virDomainSnapshotPtr snapshot, unsigned int flags)
- typedef int(\* [virDrvDomainSnapshotHasMetadata](#) )(virDomainSnapshotPtr snapshot, unsigned int flags)
- typedef int(\* [virDrvDomainRevertToSnapshot](#) )(virDomainSnapshotPtr snapshot, unsigned int flags)
- typedef int(\* [virDrvDomainSnapshotDelete](#) )(virDomainSnapshotPtr snapshot, unsigned int flags)
- typedef int(\* [virDrvDomainQemuMonitorCommand](#) )(virDomainPtr domain, const char \*cmd, char \*\*result, unsigned int flags)
- typedef char \*(\* [virDrvDomainQemuAgentCommand](#) )(virDomainPtr domain, const char \*cmd, int timeout, unsigned int flags)
- typedef [virDomainPtr](#)(\* [virDrvDomainQemuAttach](#) )(virConnectPtr conn, unsigned int pid\_value, unsigned int flags)
- typedef int(\* [virDrvDomainOpenConsole](#) )(virDomainPtr dom, const char \*dev\_name, [virStreamPtr](#) st, unsigned int flags)
- typedef int(\* [virDrvDomainOpenGraphics](#) )(virDomainPtr dom, unsigned int idx, int fd, unsigned int flags)
- typedef int(\* [virDrvDomainInjectNMI](#) )(virDomainPtr dom, unsigned int flags)
- typedef int(\* [virDrvDomainSendKey](#) )(virDomainPtr dom, unsigned int codeset, unsigned int holdtime, unsigned int \*keycodes, int nkeycodes, unsigned int flags)
- typedef char \*(\* [virDrvDomainMigrateBegin3](#) )(virDomainPtr domain, const char \*xmlin, char \*\*cookieout, int \*cookieoutlen, unsigned long flags, const char \*dname, unsigned long resource)

- typedef int(\* [virDrvDomainMigratePrepare3](#) )(virConnectPtr dconn, const char \*cookiein, int cookieinlen, char \*\*cookieout, int \*cookieoutlen, const char \*uri\_in, char \*\*uri\_out, unsigned long flags, const char \*dname, unsigned long resource, const char \*dom\_xml)
- typedef int(\* [virDrvDomainMigratePrepareTunnel3](#) )(virConnectPtr dconn, [virStreamPtr](#) st, const char \*cookiein, int cookieinlen, char \*\*cookieout, int \*cookieoutlen, unsigned long flags, const char \*dname, unsigned long resource, const char \*dom\_xml)
- typedef int(\* [virDrvDomainMigratePerform3](#) )(virDomainPtr dom, const char \*xmlin, const char \*cookiein, int cookieinlen, char \*\*cookieout, int \*cookieoutlen, const char \*dconnuri, const char \*uri, unsigned long flags, const char \*dname, unsigned long resource)
- typedef [virDomainPtr](#)(\* [virDrvDomainMigrateFinish3](#) )(virConnectPtr dconn, const char \*dname, const char \*cookiein, int cookieinlen, char \*\*cookieout, int \*cookieoutlen, const char \*dconnuri, const char \*uri, unsigned long flags, int cancelled)
- typedef int(\* [virDrvDomainMigrateConfirm3](#) )(virDomainPtr domain, const char \*cookiein, int cookieinlen, unsigned long flags, int cancelled)
- typedef int(\* [virDrvNodeSuspendForDuration](#) )(virConnectPtr conn, unsigned int target, unsigned long long duration, unsigned int flags)
- typedef int(\* [virDrvDomainBlockJobAbort](#) )(virDomainPtr dom, const char \*path, unsigned int flags)
- typedef int(\* [virDrvDomainGetBlockJobInfo](#) )(virDomainPtr dom, const char \*path, [virDomainBlockJobInfoPtr](#) info, unsigned int flags)
- typedef int(\* [virDrvDomainBlockJobSetSpeed](#) )(virDomainPtr dom, const char \*path, unsigned long bandwidth, unsigned int flags)
- typedef int(\* [virDrvDomainBlockPull](#) )(virDomainPtr dom, const char \*path, unsigned long bandwidth, unsigned int flags)
- typedef int(\* [virDrvDomainBlockRebase](#) )(virDomainPtr dom, const char \*path, const char \*base, unsigned long bandwidth, unsigned int flags)
- typedef int(\* [virDrvDomainBlockCommit](#) )(virDomainPtr dom, const char \*disk, const char \*base, const char \*top, unsigned long bandwidth, unsigned int flags)
- typedef int(\* [virDrvSetKeepAlive](#) )(virConnectPtr conn, int interval, unsigned int count)
- typedef int(\* [virDrvDomainSetBlockIoTune](#) )(virDomainPtr dom, const char \*disk, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvDomainGetBlockIoTune](#) )(virDomainPtr dom, const char \*disk, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvDomainShutdownFlags](#) )(virDomainPtr domain, unsigned int flags)
- typedef int(\* [virDrvDomainGetCPUStats](#) )(virDomainPtr domain, [virTypedParameterPtr](#) params, unsigned int nparams, int start\_cpu, unsigned int ncpus, unsigned int flags)
- typedef int(\* [virDrvDomainGetDiskErrors](#) )(virDomainPtr dom, [virDomainDiskErrorPtr](#) errors, unsigned int maxerrors, unsigned int flags)
- typedef int(\* [virDrvDomainSetMetadata](#) )(virDomainPtr dom, int type, const char \*metadata, const char \*key, const char \*uri, unsigned int flags)
- typedef char \*(\* [virDrvDomainGetMetadata](#) )(virDomainPtr dom, int type, const char \*uri, unsigned int flags)
- typedef int(\* [virDrvNodeGetMemoryParameters](#) )(virConnectPtr conn, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvNodeSetMemoryParameters](#) )(virConnectPtr conn, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- typedef int(\* [virDrvNumOfNetworks](#) )(virConnectPtr conn)
- typedef int(\* [virDrvListNetworks](#) )(virConnectPtr conn, char \*\*const names, int maxnames)
- typedef int(\* [virDrvNumOfDefinedNetworks](#) )(virConnectPtr conn)
- typedef int(\* [virDrvListDefinedNetworks](#) )(virConnectPtr conn, char \*\*const names, int maxnames)
- typedef int(\* [virDrvListAllNetworks](#) )(virConnectPtr conn, [virNetworkPtr](#) \*\*nets, unsigned int flags)
- typedef [virNetworkPtr](#)(\* [virDrvNetworkLookupByUUID](#) )(virConnectPtr conn, const unsigned char \*uuid)
- typedef [virNetworkPtr](#)(\* [virDrvNetworkLookupByName](#) )(virConnectPtr conn, const char \*name)
- typedef [virNetworkPtr](#)(\* [virDrvNetworkCreateXML](#) )(virConnectPtr conn, const char \*xmlDesc)
- typedef [virNetworkPtr](#)(\* [virDrvNetworkDefineXML](#) )(virConnectPtr conn, const char \*xml)
- typedef int(\* [virDrvNetworkUndefine](#) )(virNetworkPtr network)
- typedef int(\* [virDrvNetworkUpdate](#) )(virNetworkPtr network, unsigned int command, unsigned int section, int parentIndex, const char \*xml, unsigned int flags)



- typedef int(\* [virDrvNetworkCreate](#) )(virNetworkPtr network)
- typedef int(\* [virDrvNetworkDestroy](#) )(virNetworkPtr network)
- typedef char \*([virDrvNetworkGetXMLDesc](#) )(virNetworkPtr network, unsigned int flags)
- typedef char \*([virDrvNetworkGetBridgeName](#) )(virNetworkPtr network)
- **POL New** typedef char \*([virDrvNetworkGetBridgeType](#) )(virNetworkPtr network)
- typedef int(\* [virDrvNetworkGetAutostart](#) )(virNetworkPtr network, int \*autostart)
- typedef int(\* [virDrvNetworkSetAutostart](#) )(virNetworkPtr network, int autostart)
- typedef int(\* [virDrvNetworkIsActive](#) )(virNetworkPtr net)
- typedef int(\* [virDrvNetworkIsPersistent](#) )(virNetworkPtr net)
- typedef struct \_virNetworkDriver virNetworkDriver
- typedef virNetworkDriver \* virNetworkDriverPtr
- typedef int(\* [virDrvNumOfInterfaces](#) )(virConnectPtr conn)
- typedef int(\* [virDrvListInterfaces](#) )(virConnectPtr conn, char \*\*const names, int maxnames)
- typedef int(\* [virDrvNumOfDefinedInterfaces](#) )(virConnectPtr conn)
- typedef int(\* [virDrvListDefinedInterfaces](#) )(virConnectPtr conn, char \*\*const names, int maxnames)
- typedef int(\* [virDrvListAllInterfaces](#) )(virConnectPtr conn, virInterfacePtr \*\*ifaces, unsigned int flags)
- typedef virInterfacePtr(\* [virDrvInterfaceLookupByName](#) )(virConnectPtr conn, const char \*name)
- typedef virInterfacePtr(\* [virDrvInterfaceLookupByMACString](#) )(virConnectPtr conn, const char \*mac)
- typedef char \*([virDrvInterfaceGetXMLDesc](#) )(virInterfacePtr iface, unsigned int flags)
- typedef virInterfacePtr(\* [virDrvInterfaceDefineXML](#) )(virConnectPtr conn, const char \*xmlDesc, unsigned int flags)
- typedef int(\* [virDrvInterfaceUndefine](#) )(virInterfacePtr iface)
- typedef int(\* [virDrvInterfaceCreate](#) )(virInterfacePtr iface, unsigned int flags)
- typedef int(\* [virDrvInterfaceDestroy](#) )(virInterfacePtr iface, unsigned int flags)
- typedef int(\* [virDrvInterfaceIsActive](#) )(virInterfacePtr iface)
- typedef int(\* [virDrvInterfaceChangeBegin](#) )(virConnectPtr conn, unsigned int flags)
- typedef int(\* [virDrvInterfaceChangeCommit](#) )(virConnectPtr conn, unsigned int flags)
- typedef int(\* [virDrvInterfaceChangeRollback](#) )(virConnectPtr conn, unsigned int flags)
- typedef struct \_virInterfaceDriver virInterfaceDriver
- typedef virInterfaceDriver \* virInterfaceDriverPtr
- typedef int(\* [virDrvConnectNumOfStoragePools](#) )(virConnectPtr conn)
- typedef int(\* [virDrvConnectListStoragePools](#) )(virConnectPtr conn, char \*\*const names, int maxnames)
- typedef int(\* [virDrvConnectNumOfDefinedStoragePools](#) )(virConnectPtr conn)
- typedef int(\* [virDrvConnectListDefinedStoragePools](#) )(virConnectPtr conn, char \*\*const names, int maxnames)
- typedef int(\* [virDrvConnectListAllStoragePools](#) )(virConnectPtr conn, virStoragePoolPtr \*\*pools, unsigned int flags)
- typedef char \*([virDrvConnectFindStoragePoolSources](#) )(virConnectPtr conn, const char \*type, const char \*srcSpec, unsigned int flags)
- typedef virStoragePoolPtr(\* [virDrvStoragePoolLookupByName](#) )(virConnectPtr conn, const char \*name)
- typedef virStoragePoolPtr(\* [virDrvStoragePoolLookupByUUID](#) )(virConnectPtr conn, const unsigned char \*uuid)
- typedef virStoragePoolPtr(\* [virDrvStoragePoolLookupByVolume](#) )(virStorageVolPtr vol)
- typedef virStoragePoolPtr(\* [virDrvStoragePoolCreateXML](#) )(virConnectPtr conn, const char \*xmlDesc, unsigned int flags)
- typedef virStoragePoolPtr(\* [virDrvStoragePoolDefineXML](#) )(virConnectPtr conn, const char \*xmlDesc, unsigned int flags)
- typedef int(\* [virDrvStoragePoolUndefine](#) )(virStoragePoolPtr pool)
- typedef int(\* [virDrvStoragePoolBuild](#) )(virStoragePoolPtr pool, unsigned int flags)
- typedef int(\* [virDrvStoragePoolCreate](#) )(virStoragePoolPtr pool, unsigned int flags)
- typedef int(\* [virDrvStoragePoolDestroy](#) )(virStoragePoolPtr pool)
- typedef int(\* [virDrvStoragePoolDelete](#) )(virStoragePoolPtr pool, unsigned int flags)
- typedef int(\* [virDrvStoragePoolRefresh](#) )(virStoragePoolPtr pool, unsigned int flags)
- typedef int(\* [virDrvStoragePoolGetInfo](#) )(virStoragePoolPtr vol, virStoragePoolInfoPtr info)
- typedef char \*([virDrvStoragePoolGetXMLDesc](#) )(virStoragePoolPtr pool, unsigned int flags)

- typedef int(\* [virDrvStoragePoolGetAutostart](#) )(virStoragePoolPtr pool, int \*autostart)
- typedef int(\* [virDrvStoragePoolSetAutostart](#) )(virStoragePoolPtr pool, int autostart)
- typedef int(\* [virDrvStoragePoolNumOfVolumes](#) )(virStoragePoolPtr pool)
- typedef int(\* [virDrvStoragePoolListVolumes](#) )(virStoragePoolPtr pool, char \*\*const names, int maxnames)
- typedef int(\* [virDrvStoragePoolListAllVolumes](#) )(virStoragePoolPtr pool, [virStorageVolPtr](#) \*\*vols, unsigned int flags)
- typedef [virStorageVolPtr](#)(\* [virDrvStorageVolLookupByName](#) )(virStoragePoolPtr pool, const char \*name)
- typedef [virStorageVolPtr](#)(\* [virDrvStorageVolLookupByKey](#) )(virConnectPtr pool, const char \*key)
- typedef [virStorageVolPtr](#)(\* [virDrvStorageVolLookupByPath](#) )(virConnectPtr pool, const char \*path)
- typedef [virStorageVolPtr](#)(\* [virDrvStorageVolCreateXML](#) )(virStoragePoolPtr pool, const char \*xmldesc, unsigned int flags)
- typedef int(\* [virDrvStorageVolDelete](#) )(virStorageVolPtr vol, unsigned int flags)
- typedef int(\* [virDrvStorageVolWipe](#) )(virStorageVolPtr vol, unsigned int flags)
- typedef int(\* [virDrvStorageVolWipePattern](#) )(virStorageVolPtr vol, unsigned int algorithm, unsigned int flags)
- typedef int(\* [virDrvStorageVolGetInfo](#) )(virStorageVolPtr vol, [virStorageVolInfoPtr](#) info)
- typedef char \*(\* [virDrvStorageVolGetXMLDesc](#) )(virStorageVolPtr pool, unsigned int flags)
- typedef char \*(\* [virDrvStorageVolGetPath](#) )(virStorageVolPtr vol)
- typedef [virStorageVolPtr](#)(\* [virDrvStorageVolCreateXMLFrom](#) )(virStoragePoolPtr pool, const char \*xmldesc, [virStorageVolPtr](#) clone, unsigned int flags)
- typedef int(\* [virDrvStorageVolDownload](#) )(virStorageVolPtr vol, [virStreamPtr](#) stream, unsigned long long offset, unsigned long long length, unsigned int flags)
- typedef int(\* [virDrvStorageVolUpload](#) )(virStorageVolPtr vol, [virStreamPtr](#) stream, unsigned long long offset, unsigned long long length, unsigned int flags)
- typedef int(\* [virDrvStorageVolResize](#) )(virStorageVolPtr vol, unsigned long long capacity, unsigned int flags)
- typedef int(\* [virDrvStoragePoolsActive](#) )(virStoragePoolPtr pool)
- typedef int(\* [virDrvStoragePoolsPersistent](#) )(virStoragePoolPtr pool)
- typedef struct \_virStorageDriver virStorageDriver
- typedef [virStorageDriver](#) \* [virStorageDriverPtr](#)
- typedef struct \_virDeviceMonitor virDeviceMonitor
- typedef [virDeviceMonitor](#) \* [virDeviceMonitorPtr](#)
- typedef int(\* [virDevMonNumOfDevices](#) )(virConnectPtr conn, const char \*cap, unsigned int flags)
- typedef int(\* [virDevMonListDevices](#) )(virConnectPtr conn, const char \*cap, char \*\*const names, int maxnames, unsigned int flags)
- typedef int(\* [virDevMonListAllNodeDevices](#) )(virConnectPtr conn, [virNodeDevicePtr](#) \*\*devices, unsigned int flags)
- typedef [virNodeDevicePtr](#)(\* [virDevMonDeviceLookupByName](#) )(virConnectPtr conn, const char \*name)
- typedef char \*(\* [virDevMonDeviceGetXMLDesc](#) )(virNodeDevicePtr dev, unsigned int flags)
- typedef char \*(\* [virDevMonDeviceGetParent](#) )(virNodeDevicePtr dev)
- typedef int(\* [virDevMonDeviceNumOfCaps](#) )(virNodeDevicePtr dev)
- typedef int(\* [virDevMonDeviceListCaps](#) )(virNodeDevicePtr dev, char \*\*const names, int maxnames)
- typedef [virNodeDevicePtr](#)(\* [virDrvNodeDeviceCreateXML](#) )(virConnectPtr conn, const char \*xmlDesc, unsigned int flags)
- typedef int(\* [virDrvNodeDeviceDestroy](#) )(virNodeDevicePtr dev)
- typedef [virSecretPtr](#)(\* [virDrvSecretLookupByUUID](#) )(virConnectPtr conn, const unsigned char \*uuid)
- typedef [virSecretPtr](#)(\* [virDrvSecretLookupByUsage](#) )(virConnectPtr conn, int usageType, const char \*usage-ID)
- typedef [virSecretPtr](#)(\* [virDrvSecretDefineXML](#) )(virConnectPtr conn, const char \*xml, unsigned int flags)
- typedef char \*(\* [virDrvSecretGetXMLDesc](#) )(virSecretPtr secret, unsigned int flags)
- typedef int(\* [virDrvSecretSetValue](#) )(virSecretPtr secret, const unsigned char \*value, size\_t value\_size, unsigned int flags)
- typedef unsigned char \*(\* [virDrvSecretGetValue](#) )(virSecretPtr secret, size\_t \*value\_size, unsigned int flags, unsigned int internalFlags)
- typedef int(\* [virDrvSecretUndefine](#) )(virSecretPtr secret)
- typedef int(\* [virDrvNumOfSecrets](#) )(virConnectPtr conn)
- typedef int(\* [virDrvListSecrets](#) )(virConnectPtr conn, char \*\*uuids, int maxuuids)

- typedef int(\* [virDrvListAllSecrets](#) )(virConnectPtr conn, virSecretPtr \*\*secrets, unsigned int flags)
- typedef struct \_virSecretDriver virSecretDriver
- typedef virSecretDriver \* virSecretDriverPtr
- typedef struct \_virStreamDriver virStreamDriver
- typedef virStreamDriver \* virStreamDriverPtr
- typedef int(\* [virDrvStreamSend](#) )(virStreamPtr st, const char \*data, size\_t nbytes)
- typedef int(\* [virDrvStreamRecv](#) )(virStreamPtr st, char \*data, size\_t nbytes)
- typedef int(\* [virDrvStreamEventAddCallback](#) )(virStreamPtr stream, int events, virStreamEventCallback cb, void \*opaque, virFreeCallback ff)
- typedef int(\* [virDrvStreamEventUpdateCallback](#) )(virStreamPtr stream, int events)
- typedef int(\* [virDrvStreamEventRemoveCallback](#) )(virStreamPtr stream)
- typedef int(\* [virDrvStreamFinish](#) )(virStreamPtr st)
- typedef int(\* [virDrvStreamAbort](#) )(virStreamPtr st)
- typedef int(\* [virDrvConnectNumOfNWFilters](#) )(virConnectPtr conn)
- typedef int(\* [virDrvConnectListNWFilters](#) )(virConnectPtr conn, char \*\*const names, int maxnames)
- typedef int(\* [virDrvConnectListAllNWFilters](#) )(virConnectPtr conn, virNWFilterPtr \*\*filters, unsigned int flags)
- typedef virNWFilterPtr(\* [virDrvNWFilterLookupByName](#) )(virConnectPtr conn, const char \*name)
- typedef virNWFilterPtr(\* [virDrvNWFilterLookupByUUID](#) )(virConnectPtr conn, const unsigned char \*uuid)
- typedef virNWFilterPtr(\* [virDrvNWFilterDefineXML](#) )(virConnectPtr conn, const char \*xmlDesc)
- typedef int(\* [virDrvNWFilterUndefine](#) )(virNWFilterPtr nwfilter)
- typedef char \*(\* [virDrvNWFilterGetXMLDesc](#) )(virNWFilterPtr nwfilter, unsigned int flags)
- typedef struct \_virNWFilterDriver virNWFilterDriver
- typedef virNWFilterDriver \* virNWFilterDriverPtr

## Enumerations

- enum [virDrvNo](#) {  
[VIR\\_DRV\\_XEN\\_UNIFIED](#) = 1, [VIR\\_DRV\\_TEST](#) = 2, [VIR\\_DRV\\_QEMU](#) = 3, [VIR\\_DRV\\_REMOTE](#) = 4,  
[VIR\\_DRV\\_OPENVZ](#) = 5, [VIR\\_DRV\\_LXC](#) = 6, [VIR\\_DRV\\_UML](#) = 7, [VIR\\_DRV\\_VBOX](#) = 8,  
[VIR\\_DRV\\_ONE](#) = 9, [VIR\\_DRV\\_ESX](#) = 10, [VIR\\_DRV\\_PHYP](#) = 11, [VIR\\_DRV\\_XENAPI](#) = 12,  
[VIR\\_DRV\\_VMWARE](#) = 13, [VIR\\_DRV\\_LIBXL](#) = 14, [VIR\\_DRV\\_HYPERV](#) = 15, [VIR\\_DRV\\_PARALLELS](#) = 16 }
- enum [virDrvOpenStatus](#) { [VIR\\_DRV\\_OPEN\\_SUCCESS](#) = 0, [VIR\\_DRV\\_OPEN\\_DECLINED](#) = -1, [VIR\\_DRV\\_OPEN\\_ERROR](#) = -2 }
- enum { [VIR\\_SECRET\\_GET\\_VALUE\\_INTERNAL\\_CALL](#) = 1 << 0 }

## Functions

- int [virRegisterDriver](#) (virDriverPtr)
- int [virRegisterNetworkDriver](#) (virNetworkDriverPtr)
- int [virRegisterInterfaceDriver](#) (virInterfaceDriverPtr)
- int [virRegisterStorageDriver](#) (virStorageDriverPtr)
- int [virRegisterDeviceMonitor](#) (virDeviceMonitorPtr)
- int [virRegisterSecretDriver](#) (virSecretDriverPtr)
- int [virRegisterNWFilterDriver](#) (virNWFilterDriverPtr)
- void [virDriverModuleInitialize](#) (const char \*defmoddir)
- void \* [virDriverLoadModule](#) (const char \*name)

### 4.6.1 Macro Definition Documentation

#### 4.6.1.1 #define VIR\_DRV\_SUPPORTS\_FEATURE( drv, conn, feature )

##### Value:

```
((drv)->supports_feature ?  

    (drv)->supports_feature((conn), (feature)) > 0 : 0)
```

## 4.6.2 Typedef Documentation

4.6.2.1 typedef struct `_virDeviceMonitor` `virDeviceMonitor`

4.6.2.2 typedef `virDeviceMonitor*` `virDeviceMonitorPtr`

4.6.2.3 typedef `char*(* virDevMonDeviceGetParent)(virNodeDevicePtr dev)`

4.6.2.4 typedef `char*(* virDevMonDeviceGetXMLDesc)(virNodeDevicePtr dev, unsigned int flags)`

4.6.2.5 typedef `int(* virDevMonDeviceListCaps)(virNodeDevicePtr dev, char **const names, int maxnames)`

4.6.2.6 typedef `virNodeDevicePtr(* virDevMonDeviceLookupByName)(virConnectPtr conn, const char *name)`

4.6.2.7 typedef `int(* virDevMonDeviceNumOfCaps)(virNodeDevicePtr dev)`

4.6.2.8 typedef `int(* virDevMonListAllNodeDevices)(virConnectPtr conn, virNodeDevicePtr **devices, unsigned int flags)`

4.6.2.9 typedef `int(* virDevMonListDevices)(virConnectPtr conn, const char *cap, char **const names, int maxnames, unsigned int flags)`

4.6.2.10 typedef `int(* virDevMonNumOfDevices)(virConnectPtr conn, const char *cap, unsigned int flags)`

4.6.2.11 typedef struct `_virDriver` `virDriver`

4.6.2.12 typedef `virDriver*` `virDriverPtr`

4.6.2.13 typedef `char*(* virDrvBaselineCPU)(virConnectPtr conn, const char **xmlCPUs, unsigned int ncpus, unsigned int flags)`

4.6.2.14 typedef `int(* virDrvClose)(virConnectPtr conn)`

4.6.2.15 typedef `int(* virDrvCompareCPU)(virConnectPtr conn, const char *cpu, unsigned int flags)`

4.6.2.16 typedef `char*(* virDrvConnectDomainXMLFromNative)(virConnectPtr conn, const char *nativeFormat, const char *nativeConfig, unsigned int flags)`

4.6.2.17 typedef `char*(* virDrvConnectDomainXMLToNative)(virConnectPtr conn, const char *nativeFormat, const char *domainXml, unsigned int flags)`

4.6.2.18 typedef `char*(* virDrvConnectFindStoragePoolSources)(virConnectPtr conn, const char *type, const char *srcSpec, unsigned int flags)`

4.6.2.19 typedef `int(* virDrvConnectIsAlive)(virConnectPtr conn)`

4.6.2.20 typedef `int(* virDrvConnectIsEncrypted)(virConnectPtr conn)`

4.6.2.21 typedef `int(* virDrvConnectIsSecure)(virConnectPtr conn)`

4.6.2.22 typedef `int(* virDrvConnectListAllNWFilters)(virConnectPtr conn, virNWFilterPtr **filters, unsigned int flags)`

4.6.2.23 typedef `int(* virDrvConnectListAllStoragePools)(virConnectPtr conn, virStoragePoolPtr **pools, unsigned int flags)`

4.6.2.24 typedef `int(* virDrvConnectListDefinedStoragePools)(virConnectPtr conn, char **const names, int maxnames)`

- 4.6.2.25 `typedef int(* virDrvConnectListNWFilters)(virConnectPtr conn, char **const names, int maxnames)`
- 4.6.2.26 `typedef int(* virDrvConnectListStoragePools)(virConnectPtr conn, char **const names, int maxnames)`
- 4.6.2.27 `typedef int(* virDrvConnectNumOfDefinedStoragePools)(virConnectPtr conn)`
- 4.6.2.28 `typedef int(* virDrvConnectNumOfNWFilters)(virConnectPtr conn)`
- 4.6.2.29 `typedef int(* virDrvConnectNumOfStoragePools)(virConnectPtr conn)`
- 4.6.2.30 `typedef int(* virDrvDomainAbortJob)(virDomainPtr domain)`
- 4.6.2.31 `typedef int(* virDrvDomainAttachDevice)(virDomainPtr domain, const char *xml)`
- 4.6.2.32 `typedef int(* virDrvDomainAttachDeviceFlags)(virDomainPtr domain, const char *xml, unsigned int flags)`
- 4.6.2.33 `typedef int(* virDrvDomainBlockCommit)(virDomainPtr dom, const char *disk, const char *base, const char *top, unsigned long bandwidth, unsigned int flags)`
- 4.6.2.34 `typedef int(* virDrvDomainBlockJobAbort)(virDomainPtr dom, const char *path, unsigned int flags)`
- 4.6.2.35 `typedef int(* virDrvDomainBlockJobSetSpeed)(virDomainPtr dom, const char *path, unsigned long bandwidth, unsigned int flags)`
- 4.6.2.36 `typedef int(* virDrvDomainBlockPeek)(virDomainPtr domain, const char *path, unsigned long long offset, size_t size, void *buffer, unsigned int flags)`
- 4.6.2.37 `typedef int(* virDrvDomainBlockPull)(virDomainPtr dom, const char *path, unsigned long bandwidth, unsigned int flags)`
- 4.6.2.38 `typedef int(* virDrvDomainBlockRebase)(virDomainPtr dom, const char *path, const char *base, unsigned long bandwidth, unsigned int flags)`
- 4.6.2.39 `typedef int(* virDrvDomainBlockResize)(virDomainPtr domain, const char *path, unsigned long long size, unsigned int flags)`
- 4.6.2.40 `typedef int(* virDrvDomainBlockStats)(virDomainPtr domain, const char *path, struct _virDomainBlockStats *stats)`
- 4.6.2.41 `typedef int(* virDrvDomainBlockStatsFlags)(virDomainPtr domain, const char *path, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- 4.6.2.42 `typedef int(* virDrvDomainCoreDump)(virDomainPtr domain, const char *to, unsigned int flags)`
- 4.6.2.43 `typedef int(* virDrvDomainCreate)(virDomainPtr dom)`
- 4.6.2.44 `typedef int(* virDrvDomainCreateWithFlags)(virDomainPtr dom, unsigned int flags)`
- 4.6.2.45 `typedef virDomainPtr(* virDrvDomainCreateXML)(virConnectPtr conn, const char *xmlDesc, unsigned int flags)`
- 4.6.2.46 `typedef virDomainPtr(* virDrvDomainDefineXML)(virConnectPtr conn, const char *xml)`
- 4.6.2.47 `typedef int(* virDrvDomainDestroy)(virDomainPtr domain)`
- 4.6.2.48 `typedef int(* virDrvDomainDestroyFlags)(virDomainPtr domain, unsigned int flags)`

- 4.6.2.49 `typedef int(* virDrvDomainDetachDevice)(virDomainPtr domain, const char *xml)`
- 4.6.2.50 `typedef int(* virDrvDomainDetachDeviceFlags)(virDomainPtr domain, const char *xml, unsigned int flags)`
- 4.6.2.51 `typedef int(* virDrvDomainEventDeregister)(virConnectPtr conn, virConnectDomainEventCallback cb)`
- 4.6.2.52 `typedef int(* virDrvDomainEventDeregisterAny)(virConnectPtr conn, int callbackID)`
- 4.6.2.53 `typedef int(* virDrvDomainEventRegister)(virConnectPtr conn, virConnectDomainEventCallback cb, void *opaque, virFreeCallback freecb)`
- 4.6.2.54 `typedef int(* virDrvDomainEventRegisterAny)(virConnectPtr conn, virDomainPtr dom, int eventID, virConnectDomainEventGenericCallback cb, void *opaque, virFreeCallback freecb)`
- 4.6.2.55 `typedef int(* virDrvDomainGetAutostart)(virDomainPtr domain, int *autostart)`
- 4.6.2.56 `typedef int(* virDrvDomainGetBlkioParameters)(virDomainPtr domain, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- 4.6.2.57 `typedef int(* virDrvDomainGetBlockInfo)(virDomainPtr domain, const char *path, virDomainBlockInfoPtr info, unsigned int flags)`
- 4.6.2.58 `typedef int(* virDrvDomainGetBlockioTune)(virDomainPtr dom, const char *disk, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- 4.6.2.59 `typedef int(* virDrvDomainGetBlockJobInfo)(virDomainPtr dom, const char *path, virDomainBlockJobInfoPtr info, unsigned int flags)`
- 4.6.2.60 `typedef int(* virDrvDomainGetControllInfo)(virDomainPtr domain, virDomainControllInfoPtr info, unsigned int flags)`
- 4.6.2.61 `typedef int(* virDrvDomainGetCPUStats)(virDomainPtr domain, virTypedParameterPtr params, unsigned int nparams, int start_cpu, unsigned int ncpus, unsigned int flags)`
- 4.6.2.62 `typedef int(* virDrvDomainGetDiskErrors)(virDomainPtr dom, virDomainDiskErrorPtr errors, unsigned int maxerrors, unsigned int flags)`
- 4.6.2.63 `typedef int(* virDrvDomainGetEmulatorPinInfo)(virDomainPtr domain, unsigned char *cpumaps, int maplen, unsigned int flags)`
- 4.6.2.64 `typedef char*(* virDrvDomainGetHostname)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.65 `typedef int(* virDrvDomainGetInfo)(virDomainPtr domain, virDomainInfoPtr info)`
- 4.6.2.66 `typedef int(* virDrvDomainGetInterfaceParameters)(virDomainPtr dom, const char *device, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- 4.6.2.67 `typedef int(* virDrvDomainGetJobInfo)(virDomainPtr domain, virDomainJobInfoPtr info)`
- 4.6.2.68 `typedef unsigned long long(* virDrvDomainGetMaxMemory)(virDomainPtr domain)`
- 4.6.2.69 `typedef int(* virDrvDomainGetMaxVcpus)(virDomainPtr domain)`
- 4.6.2.70 `typedef int(* virDrvDomainGetMemoryParameters)(virDomainPtr domain, virTypedParameterPtr params, int *nparams, unsigned int flags)`

- 4.6.2.71 `typedef char*(* virDrvDomainGetMetadata)(virDomainPtr dom, int type, const char *uri, unsigned int flags)`
- 4.6.2.72 `typedef int(* virDrvDomainGetNumaParameters)(virDomainPtr domain, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- 4.6.2.73 `typedef char*(* virDrvDomainGetOSType)(virDomainPtr domain)`
- 4.6.2.74 `typedef int(* virDrvDomainGetSchedulerParameters)(virDomainPtr domain, virTypedParameterPtr params, int *nparams)`
- 4.6.2.75 `typedef int(* virDrvDomainGetSchedulerParametersFlags)(virDomainPtr domain, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- 4.6.2.76 `typedef char*(* virDrvDomainGetSchedulerType)(virDomainPtr domain, int *nparams)`
- 4.6.2.77 `typedef int(* virDrvDomainGetSecurityLabel)(virDomainPtr domain, virSecurityLabelPtr seclabel)`
- 4.6.2.78 `typedef int(* virDrvDomainGetSecurityLabelList)(virDomainPtr domain, virSecurityLabelPtr *seclabels)`
- 4.6.2.79 `typedef int(* virDrvDomainGetState)(virDomainPtr domain, int *state, int *reason, unsigned int flags)`
- 4.6.2.80 `typedef int(* virDrvDomainGetVcpuPinInfo)(virDomainPtr domain, int ncpumaps, unsigned char *cpumaps, int maplen, unsigned int flags)`
- 4.6.2.81 `typedef int(* virDrvDomainGetVcpus)(virDomainPtr domain, virVcpuInfoPtr info, int maxinfo, unsigned char *cpumaps, int maplen)`
- 4.6.2.82 `typedef int(* virDrvDomainGetVcpusFlags)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.83 `typedef char*(* virDrvDomainGetXMLDesc)(virDomainPtr dom, unsigned int flags)`
- 4.6.2.84 `typedef int(* virDrvDomainHasCurrentSnapshot)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.85 `typedef int(* virDrvDomainHasManagedSaveImage)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.86 `typedef int(* virDrvDomainInjectNMI)(virDomainPtr dom, unsigned int flags)`
- 4.6.2.87 `typedef int(* virDrvDomainInterfaceStats)(virDomainPtr domain, const char *path, struct _virDomainInterfaceStats *stats)`
- 4.6.2.88 `typedef int(* virDrvDomainsIsActive)(virDomainPtr dom)`
- 4.6.2.89 `typedef int(* virDrvDomainsIsPersistent)(virDomainPtr dom)`
- 4.6.2.90 `typedef int(* virDrvDomainsIsUpdated)(virDomainPtr dom)`
- 4.6.2.91 `typedef int(* virDrvDomainListAllSnapshots)(virDomainPtr domain, virDomainSnapshotPtr **snaps, unsigned int flags)`
- 4.6.2.92 `typedef virDomainPtr(* virDrvDomainLookupByID)(virConnectPtr conn, int id)`
- 4.6.2.93 `typedef virDomainPtr(* virDrvDomainLookupByName)(virConnectPtr conn, const char *name)`
- 4.6.2.94 `typedef virDomainPtr(* virDrvDomainLookupByUUID)(virConnectPtr conn, const unsigned char *uuid)`
- 4.6.2.95 `typedef int(* virDrvDomainManagedSave)(virDomainPtr domain, unsigned int flags)`

- 4.6.2.96 `typedef int(* virDrvDomainManagedSaveRemove)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.97 `typedef int(* virDrvDomainMemoryPeek)(virDomainPtr domain, unsigned long long start, size_t size, void *buffer, unsigned int flags)`
- 4.6.2.98 `typedef int(* virDrvDomainMemoryStats)(virDomainPtr domain, struct _virDomainMemoryStat *stats, unsigned int nr_stats, unsigned int flags)`
- 4.6.2.99 `typedef char*( * virDrvDomainMigrateBegin3)(virDomainPtr domain, const char *xmlin, char **cookieout, int *cookieoutlen, unsigned long flags, const char *dname, unsigned long resource)`
- 4.6.2.100 `typedef int(* virDrvDomainMigrateConfirm3)(virDomainPtr domain, const char *cookiein, int cookieinlen, unsigned long flags, int cancelled)`
- 4.6.2.101 `typedef virDomainPtr(* virDrvDomainMigrateFinish)(virConnectPtr dconn, const char *dname, const char *cookie, int cookieinlen, const char *uri, unsigned long flags)`
- 4.6.2.102 `typedef virDomainPtr(* virDrvDomainMigrateFinish2)(virConnectPtr dconn, const char *dname, const char *cookie, int cookieinlen, const char *uri, unsigned long flags, int retcode)`
- 4.6.2.103 `typedef virDomainPtr(* virDrvDomainMigrateFinish3)(virConnectPtr dconn, const char *dname, const char *cookiein, int cookieinlen, char **cookieout, int *cookieoutlen, const char *dconnuri, const char *uri, unsigned long flags, int cancelled)`
- 4.6.2.104 `typedef int(* virDrvDomainMigrateGetMaxSpeed)(virDomainPtr domain, unsigned long *bandwidth, unsigned int flags)`
- 4.6.2.105 `typedef int(* virDrvDomainMigratePerform)(virDomainPtr domain, const char *cookie, int cookieinlen, const char *uri, unsigned long flags, const char *dname, unsigned long resource)`
- 4.6.2.106 `typedef int(* virDrvDomainMigratePerform3)(virDomainPtr dom, const char *xmlin, const char *cookiein, int cookieinlen, char **cookieout, int *cookieoutlen, const char *dconnuri, const char *uri, unsigned long flags, const char *dname, unsigned long resource)`
- 4.6.2.107 `typedef int(* virDrvDomainMigratePrepare)(virConnectPtr dconn, char **cookie, int *cookieinlen, const char *uri_in, char **uri_out, unsigned long flags, const char *dname, unsigned long resource)`
- 4.6.2.108 `typedef int(* virDrvDomainMigratePrepare2)(virConnectPtr dconn, char **cookie, int *cookieinlen, const char *uri_in, char **uri_out, unsigned long flags, const char *dname, unsigned long resource, const char *dom_xml)`
- 4.6.2.109 `typedef int(* virDrvDomainMigratePrepare3)(virConnectPtr dconn, const char *cookiein, int cookieinlen, char **cookieout, int *cookieoutlen, const char *uri_in, char **uri_out, unsigned long flags, const char *dname, unsigned long resource, const char *dom_xml)`
- 4.6.2.110 `typedef int(* virDrvDomainMigratePrepareTunnel)(virConnectPtr dconn, virStreamPtr st, unsigned long flags, const char *dname, unsigned long resource, const char *dom_xml)`
- 4.6.2.111 `typedef int(* virDrvDomainMigratePrepareTunnel3)(virConnectPtr dconn, virStreamPtr st, const char *cookiein, int cookieinlen, char **cookieout, int *cookieoutlen, unsigned long flags, const char *dname, unsigned long resource, const char *dom_xml)`
- 4.6.2.112 `typedef int(* virDrvDomainMigrateSetMaxDowntime)(virDomainPtr domain, unsigned long long downtime, unsigned int flags)`
- 4.6.2.113 `typedef int(* virDrvDomainMigrateSetMaxSpeed)(virDomainPtr domain, unsigned long bandwidth, unsigned int flags)`



- 4.6.2.114 `typedef int(* virDrvDomainOpenConsole)(virDomainPtr dom, const char *dev_name, virStreamPtr st, unsigned int flags)`
- 4.6.2.115 `typedef int(* virDrvDomainOpenGraphics)(virDomainPtr dom, unsigned int idx, int fd, unsigned int flags)`
- 4.6.2.116 `typedef int(* virDrvDomainPinEmulator)(virDomainPtr domain, unsigned char *cpumap, int maplen, unsigned int flags)`
- 4.6.2.117 `typedef int(* virDrvDomainPinVcpu)(virDomainPtr domain, unsigned int vcpu, unsigned char *cpumap, int maplen)`
- 4.6.2.118 `typedef int(* virDrvDomainPinVcpuFlags)(virDomainPtr domain, unsigned int vcpu, unsigned char *cpumap, int maplen, unsigned int flags)`
- 4.6.2.119 `typedef int(* virDrvDomainPMSuspendForDuration)(virDomainPtr, unsigned int target, unsigned long long duration, unsigned int flags)`
- 4.6.2.120 `typedef int(* virDrvDomainPMWakeup)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.121 `typedef char*(* virDrvDomainQemuAgentCommand)(virDomainPtr domain, const char *cmd, int timeout, unsigned int flags)`
- 4.6.2.122 `typedef virDomainPtr(* virDrvDomainQemuAttach)(virConnectPtr conn, unsigned int pid_value, unsigned int flags)`
- 4.6.2.123 `typedef int(* virDrvDomainQemuMonitorCommand)(virDomainPtr domain, const char *cmd, char **result, unsigned int flags)`
- 4.6.2.124 `typedef int(* virDrvDomainReboot)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.125 `typedef int(* virDrvDomainReset)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.126 `typedef int(* virDrvDomainRestore)(virConnectPtr conn, const char *from)`
- 4.6.2.127 `typedef int(* virDrvDomainRestoreFlags)(virConnectPtr conn, const char *from, const char *dxml, unsigned int flags)`
- 4.6.2.128 `typedef int(* virDrvDomainResume)(virDomainPtr domain)`
- 4.6.2.129 `typedef int(* virDrvDomainRevertToSnapshot)(virDomainSnapshotPtr snapshot, unsigned int flags)`
- 4.6.2.130 `typedef int(* virDrvDomainSave)(virDomainPtr domain, const char *to)`
- 4.6.2.131 `typedef int(* virDrvDomainSaveFlags)(virDomainPtr domain, const char *to, const char *dxml, unsigned int flags)`
- 4.6.2.132 `typedef int(* virDrvDomainSavelmageDefineXML)(virConnectPtr conn, const char *file, const char *dxml, unsigned int flags)`
- 4.6.2.133 `typedef char*(* virDrvDomainSavelmageGetXMLDesc)(virConnectPtr conn, const char *file, unsigned int flags)`
- 4.6.2.134 `typedef char*(* virDrvDomainScreenshot)(virDomainPtr domain, virStreamPtr stream, unsigned int screen, unsigned int flags)`
- 4.6.2.135 `typedef int(* virDrvDomainSendKey)(virDomainPtr dom, unsigned int codeset, unsigned int holdtime, unsigned int *keycodes, int nkeycodes, unsigned int flags)`
- 4.6.2.136 `typedef int(* virDrvDomainSetAutostart)(virDomainPtr domain, int autostart)`

- 4.6.2.137 `typedef int(* virDrvDomainSetBlkioParameters)(virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)`
- 4.6.2.138 `typedef int(* virDrvDomainSetBlockioTune)(virDomainPtr dom, const char *disk, virTypedParameterPtr params, int nparams, unsigned int flags)`
- 4.6.2.139 `typedef int(* virDrvDomainSetInterfaceParameters)(virDomainPtr dom, const char *device, virTypedParameterPtr params, int nparams, unsigned int flags)`
- 4.6.2.140 `typedef int(* virDrvDomainSetMaxMemory)(virDomainPtr domain, unsigned long memory)`
- 4.6.2.141 `typedef int(* virDrvDomainSetMemory)(virDomainPtr domain, unsigned long memory)`
- 4.6.2.142 `typedef int(* virDrvDomainSetMemoryFlags)(virDomainPtr domain, unsigned long memory, unsigned int flags)`
- 4.6.2.143 `typedef int(* virDrvDomainSetMemoryParameters)(virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)`
- 4.6.2.144 `typedef int(* virDrvDomainSetMetadata)(virDomainPtr dom, int type, const char *metadata, const char *key, const char *uri, unsigned int flags)`
- 4.6.2.145 `typedef int(* virDrvDomainSetNumaParameters)(virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)`
- 4.6.2.146 `typedef int(* virDrvDomainSetSchedulerParameters)(virDomainPtr domain, virTypedParameterPtr params, int nparams)`
- 4.6.2.147 `typedef int(* virDrvDomainSetSchedulerParametersFlags)(virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)`
- 4.6.2.148 `typedef int(* virDrvDomainSetVcpus)(virDomainPtr domain, unsigned int nvcpus)`
- 4.6.2.149 `typedef int(* virDrvDomainSetVcpusFlags)(virDomainPtr domain, unsigned int nvcpus, unsigned int flags)`
- 4.6.2.150 `typedef int(* virDrvDomainShutdown)(virDomainPtr domain)`
- 4.6.2.151 `typedef int(* virDrvDomainShutdownFlags)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.152 `typedef virDomainSnapshotPtr(* virDrvDomainSnapshotCreateXML)(virDomainPtr domain, const char *xmlDesc, unsigned int flags)`
- 4.6.2.153 `typedef virDomainSnapshotPtr(* virDrvDomainSnapshotCurrent)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.154 `typedef int(* virDrvDomainSnapshotDelete)(virDomainSnapshotPtr snapshot, unsigned int flags)`
- 4.6.2.155 `typedef virDomainSnapshotPtr(* virDrvDomainSnapshotGetParent)(virDomainSnapshotPtr snapshot, unsigned int flags)`
- 4.6.2.156 `typedef char*( * virDrvDomainSnapshotGetXMLDesc)(virDomainSnapshotPtr snapshot, unsigned int flags)`
- 4.6.2.157 `typedef int(* virDrvDomainSnapshotHasMetadata)(virDomainSnapshotPtr snapshot, unsigned int flags)`
- 4.6.2.158 `typedef int(* virDrvDomainSnapshotIsCurrent)(virDomainSnapshotPtr snapshot, unsigned int flags)`
- 4.6.2.159 `typedef int(* virDrvDomainSnapshotListAllChildren)(virDomainSnapshotPtr snapshot, virDomainSnapshotPtr **snaps, unsigned int flags)`

- 4.6.2.160 `typedef int(* virDrvDomainSnapshotListChildrenNames)(virDomainSnapshotPtr snapshot, char **names, int nameslen, unsigned int flags)`
- 4.6.2.161 `typedef int(* virDrvDomainSnapshotListNames)(virDomainPtr domain, char **names, int nameslen, unsigned int flags)`
- 4.6.2.162 `typedef virDomainSnapshotPtr(* virDrvDomainSnapshotLookupByName)(virDomainPtr domain, const char *name, unsigned int flags)`
- 4.6.2.163 `typedef int(* virDrvDomainSnapshotNum)(virDomainPtr domain, unsigned int flags)`
- 4.6.2.164 `typedef int(* virDrvDomainSnapshotNumChildren)(virDomainSnapshotPtr snapshot, unsigned int flags)`
- 4.6.2.165 `typedef int(* virDrvDomainSuspend)(virDomainPtr domain)`
- 4.6.2.166 `typedef int(* virDrvDomainUndefine)(virDomainPtr dom)`
- 4.6.2.167 `typedef int(* virDrvDomainUndefineFlags)(virDomainPtr dom, unsigned int flags)`
- 4.6.2.168 `typedef int(* virDrvDomainUpdateDeviceFlags)(virDomainPtr domain, const char *xml, unsigned int flags)`
- 4.6.2.169 `typedef int(* virDrvDrvSupportsFeature)(virConnectPtr conn, int feature)`
- 4.6.2.170 `typedef char*(* virDrvGetCapabilities)(virConnectPtr conn)`
- 4.6.2.171 `typedef char*(* virDrvGetHostname)(virConnectPtr conn)`
- 4.6.2.172 `typedef int(* virDrvGetLibVersion)(virConnectPtr conn, unsigned long *libVer)`
- 4.6.2.173 `typedef int(* virDrvGetMaxVcpus)(virConnectPtr conn, const char *type)`
- 4.6.2.174 `typedef char*(* virDrvGetSysinfo)(virConnectPtr conn, unsigned int flags)`
- 4.6.2.175 `typedef const char*(* virDrvGetType)(virConnectPtr conn)`
- 4.6.2.176 `typedef char*(* virDrvGetURI)(virConnectPtr conn)`
- 4.6.2.177 `typedef int(* virDrvGetVersion)(virConnectPtr conn, unsigned long *hvVer)`
- 4.6.2.178 `typedef int(* virDrvInterfaceChangeBegin)(virConnectPtr conn, unsigned int flags)`
- 4.6.2.179 `typedef int(* virDrvInterfaceChangeCommit)(virConnectPtr conn, unsigned int flags)`
- 4.6.2.180 `typedef int(* virDrvInterfaceChangeRollback)(virConnectPtr conn, unsigned int flags)`
- 4.6.2.181 `typedef int(* virDrvInterfaceCreate)(virInterfacePtr iface, unsigned int flags)`
- 4.6.2.182 `typedef virInterfacePtr(* virDrvInterfaceDefineXML)(virConnectPtr conn, const char *xmlDesc, unsigned int flags)`
- 4.6.2.183 `typedef int(* virDrvInterfaceDestroy)(virInterfacePtr iface, unsigned int flags)`
- 4.6.2.184 `typedef char*(* virDrvInterfaceGetXMLDesc)(virInterfacePtr iface, unsigned int flags)`
- 4.6.2.185 `typedef int(* virDrvInterfaceIsActive)(virInterfacePtr iface)`

- 4.6.2.186 `typedef virInterfacePtr(* virDrvInterfaceLookupByMACString)(virConnectPtr conn, const char *mac)`
- 4.6.2.187 `typedef virInterfacePtr(* virDrvInterfaceLookupByName)(virConnectPtr conn, const char *name)`
- 4.6.2.188 `typedef int(* virDrvInterfaceUndefine)(virInterfacePtr iface)`
- 4.6.2.189 `typedef int(* virDrvListAllDomains)(virConnectPtr conn, virDomainPtr **domains, unsigned int flags)`
- 4.6.2.190 `typedef int(* virDrvListAllInterfaces)(virConnectPtr conn, virInterfacePtr **ifaces, unsigned int flags)`
- 4.6.2.191 `typedef int(* virDrvListAllNetworks)(virConnectPtr conn, virNetworkPtr **nets, unsigned int flags)`
- 4.6.2.192 `typedef int(* virDrvListAllSecrets)(virConnectPtr conn, virSecretPtr **secrets, unsigned int flags)`
- 4.6.2.193 `typedef int(* virDrvListDefinedDomains)(virConnectPtr conn, char **const names, int maxnames)`
- 4.6.2.194 `typedef int(* virDrvListDefinedInterfaces)(virConnectPtr conn, char **const names, int maxnames)`
- 4.6.2.195 `typedef int(* virDrvListDefinedNetworks)(virConnectPtr conn, char **const names, int maxnames)`
- 4.6.2.196 `typedef int(* virDrvListDomains)(virConnectPtr conn, int *ids, int maxids)`
- 4.6.2.197 `typedef int(* virDrvListInterfaces)(virConnectPtr conn, char **const names, int maxnames)`
- 4.6.2.198 `typedef int(* virDrvListNetworks)(virConnectPtr conn, char **const names, int maxnames)`
- 4.6.2.199 `typedef int(* virDrvListSecrets)(virConnectPtr conn, char **uuids, int maxuuids)`
- 4.6.2.200 `typedef int(* virDrvNetworkCreate)(virNetworkPtr network)`
- 4.6.2.201 `typedef virNetworkPtr(* virDrvNetworkCreateXML)(virConnectPtr conn, const char *xmlDesc)`
- 4.6.2.202 `typedef virNetworkPtr(* virDrvNetworkDefineXML)(virConnectPtr conn, const char *xml)`
- 4.6.2.203 `typedef int(* virDrvNetworkDestroy)(virNetworkPtr network)`
- 4.6.2.204 `typedef int(* virDrvNetworkGetAutostart)(virNetworkPtr network, int *autostart)`
- 4.6.2.205 `typedef char*(* virDrvNetworkGetBridgeName)(virNetworkPtr network)`
- 4.6.2.206 **POL New** `typedef char*(* virDrvNetworkGetBridgeType)(virNetworkPtr network)`
- 4.6.2.207 `typedef char*(* virDrvNetworkGetXMLDesc)(virNetworkPtr network, unsigned int flags)`
- 4.6.2.208 `typedef int(* virDrvNetworkIsActive)(virNetworkPtr net)`
- 4.6.2.209 `typedef int(* virDrvNetworkIsPersistent)(virNetworkPtr net)`
- 4.6.2.210 `typedef virNetworkPtr(* virDrvNetworkLookupByName)(virConnectPtr conn, const char *name)`
- 4.6.2.211 `typedef virNetworkPtr(* virDrvNetworkLookupByUUID)(virConnectPtr conn, const unsigned char *uuid)`
- 4.6.2.212 `typedef int(* virDrvNetworkSetAutostart)(virNetworkPtr network, int autostart)`
- 4.6.2.213 `typedef int(* virDrvNetworkUndefine)(virNetworkPtr network)`

- 4.6.2.214 `typedef int(* virDrvNetworkUpdate)(virNetworkPtr network, unsigned int command,unsigned int section,int parentIndex, const char *xml, unsigned int flags)`
- 4.6.2.215 `typedef virNodeDevicePtr(* virDrvNodeDeviceCreateXML)(virConnectPtr conn, const char *xmlDesc, unsigned int flags)`
- 4.6.2.216 `typedef int(* virDrvNodeDeviceDestroy)(virNodeDevicePtr dev)`
- 4.6.2.217 `typedef int(* virDrvNodeDeviceDetach)(virNodeDevicePtr dev)`
- 4.6.2.218 `typedef int(* virDrvNodeDeviceReAttach)(virNodeDevicePtr dev)`
- 4.6.2.219 `typedef int(* virDrvNodeDeviceReset)(virNodeDevicePtr dev)`
- 4.6.2.220 `typedef int(* virDrvNodeGetCellsFreeMemory)(virConnectPtr conn, unsigned long long *freeMems, int startCell, int maxCells)`
- 4.6.2.221 `typedef int(* virDrvNodeGetCPUStats)(virConnectPtr conn, int cpuNum, virNodeCPUStatsPtr params, int *nparams, unsigned int flags)`
- 4.6.2.222 `typedef unsigned long long(* virDrvNodeGetFreeMemory)(virConnectPtr conn)`
- 4.6.2.223 `typedef int(* virDrvNodeGetInfo)(virConnectPtr conn, virNodeInfoPtr info)`
- 4.6.2.224 `typedef int(* virDrvNodeGetMemoryParameters)(virConnectPtr conn, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- 4.6.2.225 `typedef int(* virDrvNodeGetMemoryStats)(virConnectPtr conn, int cellNum, virNodeMemoryStatsPtr params, int *nparams, unsigned int flags)`
- 4.6.2.226 `typedef int(* virDrvNodeGetSecurityModel)(virConnectPtr conn, virSecurityModelPtr secmodel)`
- 4.6.2.227 `typedef int(* virDrvNodeSetMemoryParameters)(virConnectPtr conn, virTypedParameterPtr params, int nparams, unsigned int flags)`
- 4.6.2.228 `typedef int(* virDrvNodeSuspendForDuration)(virConnectPtr conn, unsigned int target, unsigned long long duration, unsigned int flags)`
- 4.6.2.229 `typedef int(* virDrvNumOfDefinedDomains)(virConnectPtr conn)`
- 4.6.2.230 `typedef int(* virDrvNumOfDefinedInterfaces)(virConnectPtr conn)`
- 4.6.2.231 `typedef int(* virDrvNumOfDefinedNetworks)(virConnectPtr conn)`
- 4.6.2.232 `typedef int(* virDrvNumOfDomains)(virConnectPtr conn)`
- 4.6.2.233 `typedef int(* virDrvNumOfInterfaces)(virConnectPtr conn)`
- 4.6.2.234 `typedef int(* virDrvNumOfNetworks)(virConnectPtr conn)`
- 4.6.2.235 `typedef int(* virDrvNumOfSecrets)(virConnectPtr conn)`
- 4.6.2.236 `typedef virNWFilterPtr(* virDrvNWFilterDefineXML)(virConnectPtr conn, const char *xmlDesc)`
- 4.6.2.237 `typedef char*( * virDrvNWFilterGetXMLDesc)(virNWFilterPtr nwfilter, unsigned int flags)`

- 4.6.2.238 `typedef virNWFilterPtr(* virDrvNWFilterLookupByName)(virConnectPtr conn, const char *name)`
- 4.6.2.239 `typedef virNWFilterPtr(* virDrvNWFilterLookupByUUID)(virConnectPtr conn, const unsigned char *uuid)`
- 4.6.2.240 `typedef int(* virDrvNWFilterUndefine)(virNWFilterPtr nwfilter)`
- 4.6.2.241 `typedef virDrvOpenStatus(* virDrvOpen)(virConnectPtr conn, virConnectAuthPtr auth, unsigned int flags)`
- 4.6.2.242 `typedef virSecretPtr(* virDrvSecretDefineXML)(virConnectPtr conn, const char *xml, unsigned int flags)`
- 4.6.2.243 `typedef unsigned char*( * virDrvSecretGetValue)(virSecretPtr secret, size_t *value_size, unsigned int flags, unsigned int internalFlags)`
- 4.6.2.244 `typedef char*( * virDrvSecretGetXMLDesc)(virSecretPtr secret, unsigned int flags)`
- 4.6.2.245 `typedef virSecretPtr(* virDrvSecretLookupByUsage)(virConnectPtr conn, int usageType, const char *usageID)`
- 4.6.2.246 `typedef virSecretPtr(* virDrvSecretLookupByUUID)(virConnectPtr conn, const unsigned char *uuid)`
- 4.6.2.247 `typedef int(* virDrvSecretSetValue)(virSecretPtr secret, const unsigned char *value, size_t value_size, unsigned int flags)`
- 4.6.2.248 `typedef int(* virDrvSecretUndefine)(virSecretPtr secret)`
- 4.6.2.249 `typedef int(* virDrvSetKeepAlive)(virConnectPtr conn, int interval, unsigned int count)`
- 4.6.2.250 `typedef int(* virDrvStoragePoolBuild)(virStoragePoolPtr pool, unsigned int flags)`
- 4.6.2.251 `typedef int(* virDrvStoragePoolCreate)(virStoragePoolPtr pool, unsigned int flags)`
- 4.6.2.252 `typedef virStoragePoolPtr(* virDrvStoragePoolCreateXML)(virConnectPtr conn, const char *xmlDesc, unsigned int flags)`
- 4.6.2.253 `typedef virStoragePoolPtr(* virDrvStoragePoolDefineXML)(virConnectPtr conn, const char *xmlDesc, unsigned int flags)`
- 4.6.2.254 `typedef int(* virDrvStoragePoolDelete)(virStoragePoolPtr pool, unsigned int flags)`
- 4.6.2.255 `typedef int(* virDrvStoragePoolDestroy)(virStoragePoolPtr pool)`
- 4.6.2.256 `typedef int(* virDrvStoragePoolGetAutostart)(virStoragePoolPtr pool, int *autostart)`
- 4.6.2.257 `typedef int(* virDrvStoragePoolGetInfo)(virStoragePoolPtr vol, virStoragePoolInfoPtr info)`
- 4.6.2.258 `typedef char*( * virDrvStoragePoolGetXMLDesc)(virStoragePoolPtr pool, unsigned int flags)`
- 4.6.2.259 `typedef int(* virDrvStoragePoolsActive)(virStoragePoolPtr pool)`
- 4.6.2.260 `typedef int(* virDrvStoragePoolsPersistent)(virStoragePoolPtr pool)`
- 4.6.2.261 `typedef int(* virDrvStoragePoolListAllVolumes)(virStoragePoolPtr pool, virStorageVolPtr **vols, unsigned int flags)`
- 4.6.2.262 `typedef int(* virDrvStoragePoolListVolumes)(virStoragePoolPtr pool, char **const names, int maxnames)`
- 4.6.2.263 `typedef virStoragePoolPtr(* virDrvStoragePoolLookupByName)(virConnectPtr conn, const char *name)`

- 4.6.2.264 `typedef virStoragePoolPtr(* virDrvStoragePoolLookupByUUID)(virConnectPtr conn, const unsigned char *uuid)`
- 4.6.2.265 `typedef virStoragePoolPtr(* virDrvStoragePoolLookupByVolume)(virStorageVolPtr vol)`
- 4.6.2.266 `typedef int(* virDrvStoragePoolNumOfVolumes)(virStoragePoolPtr pool)`
- 4.6.2.267 `typedef int(* virDrvStoragePoolRefresh)(virStoragePoolPtr pool, unsigned int flags)`
- 4.6.2.268 `typedef int(* virDrvStoragePoolSetAutostart)(virStoragePoolPtr pool, int autostart)`
- 4.6.2.269 `typedef int(* virDrvStoragePoolUndefine)(virStoragePoolPtr pool)`
- 4.6.2.270 `typedef virStorageVolPtr(* virDrvStorageVolCreateXML)(virStoragePoolPtr pool, const char *xmldesc, unsigned int flags)`
- 4.6.2.271 `typedef virStorageVolPtr(* virDrvStorageVolCreateXMLFrom)(virStoragePoolPtr pool, const char *xmldesc, virStorageVolPtr clone, unsigned int flags)`
- 4.6.2.272 `typedef int(* virDrvStorageVolDelete)(virStorageVolPtr vol, unsigned int flags)`
- 4.6.2.273 `typedef int(* virDrvStorageVolDownload)(virStorageVolPtr vol, virStreamPtr stream, unsigned long long offset, unsigned long long length, unsigned int flags)`
- 4.6.2.274 `typedef int(* virDrvStorageVolGetInfo)(virStorageVolPtr vol, virStorageVolInfoPtr info)`
- 4.6.2.275 `typedef char*(* virDrvStorageVolGetPath)(virStorageVolPtr vol)`
- 4.6.2.276 `typedef char*(* virDrvStorageVolGetXMLDesc)(virStorageVolPtr pool, unsigned int flags)`
- 4.6.2.277 `typedef virStorageVolPtr(* virDrvStorageVolLookupByKey)(virConnectPtr pool, const char *key)`
- 4.6.2.278 `typedef virStorageVolPtr(* virDrvStorageVolLookupByName)(virStoragePoolPtr pool, const char *name)`
- 4.6.2.279 `typedef virStorageVolPtr(* virDrvStorageVolLookupByPath)(virConnectPtr pool, const char *path)`
- 4.6.2.280 `typedef int(* virDrvStorageVolResize)(virStorageVolPtr vol, unsigned long long capacity, unsigned int flags)`
- 4.6.2.281 `typedef int(* virDrvStorageVolUpload)(virStorageVolPtr vol, virStreamPtr stream, unsigned long long offset, unsigned long long length, unsigned int flags)`
- 4.6.2.282 `typedef int(* virDrvStorageVolWipe)(virStorageVolPtr vol, unsigned int flags)`
- 4.6.2.283 `typedef int(* virDrvStorageVolWipePattern)(virStorageVolPtr vol, unsigned int algorithm, unsigned int flags)`
- 4.6.2.284 `typedef int(* virDrvStreamAbort)(virStreamPtr st)`
- 4.6.2.285 `typedef int(* virDrvStreamEventAddCallback)(virStreamPtr stream, int events, virStreamEventCallback cb, void *opaque, virFreeCallback ff)`
- 4.6.2.286 `typedef int(* virDrvStreamEventRemoveCallback)(virStreamPtr stream)`
- 4.6.2.287 `typedef int(* virDrvStreamEventUpdateCallback)(virStreamPtr stream, int events)`
- 4.6.2.288 `typedef int(* virDrvStreamFinish)(virStreamPtr st)`

4.6.2.289 `typedef int(* virDrvStreamRecv)(virStreamPtr st, char *data, size_t nbytes)`

4.6.2.290 `typedef int(* virDrvStreamSend)(virStreamPtr st, const char *data, size_t nbytes)`

4.6.2.291 `typedef struct _virInterfaceDriver virInterfaceDriver`

4.6.2.292 `typedef virInterfaceDriver* virInterfaceDriverPtr`

4.6.2.293 `typedef struct _virNetworkDriver virNetworkDriver`

4.6.2.294 `typedef virNetworkDriver* virNetworkDriverPtr`

4.6.2.295 `typedef struct _virNWFilterDriver virNWFilterDriver`

4.6.2.296 `typedef virNWFilterDriver* virNWFilterDriverPtr`

4.6.2.297 `typedef struct _virSecretDriver virSecretDriver`

4.6.2.298 `typedef virSecretDriver* virSecretDriverPtr`

4.6.2.299 `typedef struct _virStorageDriver virStorageDriver`

4.6.2.300 `typedef virStorageDriver* virStorageDriverPtr`

4.6.2.301 `typedef struct _virStreamDriver virStreamDriver`

4.6.2.302 `typedef virStreamDriver* virStreamDriverPtr`

### 4.6.3 Enumeration Type Documentation

#### 4.6.3.1 anonymous enum

Enumerator

***VIR\_SECRET\_GET\_VALUE\_INTERNAL\_CALL***

#### 4.6.3.2 enum virDrvNo

Enumerator

***VIR\_DRV\_XEN\_UNIFIED***

***VIR\_DRV\_TEST***

***VIR\_DRV\_QEMU***

***VIR\_DRV\_REMOTE***

***VIR\_DRV\_OPENVZ***

***VIR\_DRV\_LXC***

***VIR\_DRV\_UML***

***VIR\_DRV\_VBOX***

***VIR\_DRV\_ONE***

***VIR\_DRV\_ESX***

***VIR\_DRV\_PHYP***

***VIR\_DRV\_XENAPI***

***VIR\_DRV\_VMWARE***



***VIR\_DRV\_LIBXL***

***VIR\_DRV\_HYPERV***

***VIR\_DRV\_PARALLELS***

#### 4.6.3.3 enum virDrvOpenStatus

Enumerator

***VIR\_DRV\_OPEN\_SUCCESS***

***VIR\_DRV\_OPEN\_DECLINED***

***VIR\_DRV\_OPEN\_ERROR***

#### 4.6.4 Function Documentation

4.6.4.1 void\* virDriverLoadModule ( const char \* *name* )

4.6.4.2 void virDriverModuleInitialize ( const char \* *defmoddir* )

4.6.4.3 int virRegisterDeviceMonitor ( virDeviceMonitorPtr *driver* )

virRegisterDeviceMonitor: : pointer to a device monitor block

Register a device monitor

Returns the driver priority or -1 in case of error.

4.6.4.4 int virRegisterDriver ( virDriverPtr *driver* )

virRegisterDriver: : pointer to a driver block

Register a virtualization driver

Returns the driver priority or -1 in case of error.

4.6.4.5 int virRegisterInterfaceDriver ( virInterfaceDriverPtr *driver* )

virRegisterInterfaceDriver: : pointer to an interface driver block

Register an interface virtualization driver

Returns the driver priority or -1 in case of error.

4.6.4.6 int virRegisterNetworkDriver ( virNetworkDriverPtr *driver* )

virRegisterNetworkDriver: : pointer to a network driver block

Register a network virtualization driver

Returns the driver priority or -1 in case of error.

4.6.4.7 int virRegisterNWFilterDriver ( virNWFilterDriverPtr *driver* )

virRegisterNWFilterDriver: : pointer to a network filter driver block

Register a network filter virtualization driver

Returns the driver priority or -1 in case of error.

#### 4.6.4.8 int virRegisterSecretDriver ( virSecretDriverPtr driver )

virRegisterSecretDriver: : pointer to a secret driver block

Register a secret driver

Returns the driver priority or -1 in case of error.

#### 4.6.4.9 int virRegisterStorageDriver ( virStorageDriverPtr driver )

virRegisterStorageDriver: : pointer to a storage driver block

Register a storage virtualization driver

Returns the driver priority or -1 in case of error.

## 4.7 src/libvirt.c File Reference

```
#include <config.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/wait.h>
#include <time.h>
#include <gcrypt.h>
#include <libxml/parser.h>
#include <libxml/xpath.h>
#include "getpass.h"
#include "virterror_internal.h"
#include "logging.h"
#include "datatypes.h"
#include "uuid.h"
#include "memory.h"
#include "configmake.h"
#include "intprops.h"
#include "conf.h"
#include "rpc/virnettlscontext.h"
#include "command.h"
#include "virrandom.h"
#include "viruri.h"
```

## Macros

- #define VIR\_FROM\_THIS VIR\_FROM\_NONE
- #define MAX\_DRIVERS 20
- #define VIR\_ARG15(\_1, \_2, \_3, \_4, \_5, \_6, \_7, \_8, \_9, \_10, \_11, \_12, \_13, \_14, \_15,...) \_15
- #define VIR\_HAS\_COMMA(...) VIR\_ARG15(\_\_VA\_ARGS\_\_, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
- #define VIR\_DOMAIN\_DEBUG\_EXPAND(a, b,...) VIR\_DOMAIN\_DEBUG\_PASTE(a, b, \_\_VA\_ARGS\_\_)
- #define VIR\_DOMAIN\_DEBUG\_PASTE(a, b,...) a##b(\_\_VA\_ARGS\_\_)
- #define VIR\_DOMAIN\_DEBUG\_0(dom) VIR\_DOMAIN\_DEBUG\_2(dom, "%s", "")
- #define VIR\_DOMAIN\_DEBUG\_1(dom, fmt,...) VIR\_DOMAIN\_DEBUG\_2(dom, ", " fmt, \_\_VA\_ARGS\_\_)
- #define VIR\_DOMAIN\_DEBUG\_2(dom, fmt,...)

- #define `VIR_DOMAIN_DEBUG(...)`
- #define `VIR_UUID_DEBUG(conn, uuid)`
- #define `virLibConnError(code,...)`
- #define `virLibDomainError(code,...)`
- #define `virLibNetworkError(code,...)`
- #define `virLibStoragePoolError(code,...)`
- #define `virLibStorageVolError(code,...)`
- #define `virLibInterfaceError(code,...)`
- #define `virLibNodeDeviceError(code,...)`
- #define `virLibSecretError(code,...)`
- #define `virLibStreamError(code,...)`
- #define `virLibNWFilterError(code,...)`
- #define `virLibDomainSnapshotError(code,...)`
- #define `URI_ALIAS_CHARS "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_"`

## Functions

- static int `virConnectAuthCallbackDefault` (`virConnectCredentialPtr` cred, unsigned int ncred, void \*cbdata ATTRIBUTE\_UNUSED)
- static int `virTLSMutexInit` (void \*\*priv)
- static int `virTLSMutexDestroy` (void \*\*priv)
- static int `virTLSMutexLock` (void \*\*priv)
- static int `virTLSMutexUnlock` (void \*\*priv)
- int `virInitialize` (void)
- int `virRegisterNetworkDriver` (`virNetworkDriverPtr` driver)
- int `virRegisterInterfaceDriver` (`virInterfaceDriverPtr` driver)
- int `virRegisterStorageDriver` (`virStorageDriverPtr` driver)
- int `virRegisterDeviceMonitor` (`virDeviceMonitorPtr` driver)
- int `virRegisterSecretDriver` (`virSecretDriverPtr` driver)
- int `virRegisterNWFilterDriver` (`virNWFilterDriverPtr` driver)
- int `virRegisterDriver` (`virDriverPtr` driver)
- int `virGetVersion` (unsigned long \*libVer, const char \*type ATTRIBUTE\_UNUSED, unsigned long \*typeVer)
- static char \* `virConnectGetConfigFilePath` (void)
- static int `virConnectGetConfigFile` (`virConfPtr` \*conf)
- static int `virConnectOpenFindURIAliasMatch` (`virConfValuePtr` value, const char \*alias, char \*\*uri)
- static int `virConnectOpenResolveURIAlias` (`virConfPtr` conf, const char \*alias, char \*\*uri)
- static int `virConnectGetDefaultURI` (`virConfPtr` conf, const char \*\*name)
- static `virConnectPtr` `do_open` (const char \*name, `virConnectAuthPtr` auth, unsigned int flags)

is NULL, if the `LIBVIRT_DEFAULT_URI` environment variable is set,

*then it will be used. Otherwise if the client configuration file has the "uri\_default" parameter set, then it will be used. Finally probing will be done to determine a suitable default driver to activate. This involves trying each hypervisor in turn until one successfully opens.*

*If connecting to an unprivileged hypervisor driver which requires the libvirtd daemon to be active, it will automatically be launched if not already running. This can be prevented by setting the environment variable `LIBVIRT_AUTOSTART=0`*

URIs are documented at <http://libvirt.org/uri.html>

- `virConnectPtr` `virConnectOpen` (const char \*name)

**: URI of the hypervisor**

*virConnectOpenAuth:*

*: Authenticate callback parameters : bitwise-OR of virConnectFlags*

*This function should be called first to get a connection to the Hypervisor. If necessary, authentication will be performed fetching credentials via the callback*

*See virConnectOpen for notes about environment variables which can have an effect on opening drivers*

*Returns a pointer to the hypervisor connection or NULL in case of error*

*URLs are documented at <http://libvirt.org/uri.html>*

- [virConnectPtr virConnectOpenReadOnly](#) (const char \*name)
- [virConnectPtr virConnectOpenAuth](#) (const char \*name, [virConnectAuthPtr](#) auth, unsigned int flags)
- int [virConnectClose](#) ([virConnectPtr](#) conn)
- int [virConnectRef](#) ([virConnectPtr](#) conn)
- int [virDrvSupportsFeature](#) ([virConnectPtr](#) conn, int feature)
- const char \* [virConnectGetType](#) ([virConnectPtr](#) conn)
- int [virConnectGetVersion](#) ([virConnectPtr](#) conn, unsigned long \*hvVer)
- int [virConnectGetLibVersion](#) ([virConnectPtr](#) conn, unsigned long \*libVer)
- char \* [virConnectGetHostname](#) ([virConnectPtr](#) conn)
- char \* [virConnectGetURI](#) ([virConnectPtr](#) conn)
- char \* [virConnectGetSysinfo](#) ([virConnectPtr](#) conn, unsigned int flags)
- int [virConnectGetMaxVcpus](#) ([virConnectPtr](#) conn, const char \*type)
- int [virConnectListDomains](#) ([virConnectPtr](#) conn, int \*ids, int maxids)
- int [virConnectNumOfDomains](#) ([virConnectPtr](#) conn)
- [virConnectPtr virDomainGetConnect](#) ([virDomainPtr](#) dom)
- [virDomainPtr virDomainCreateXML](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- [virDomainPtr virDomainCreateLinux](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- [virDomainPtr virDomainLookupByID](#) ([virConnectPtr](#) conn, int id)
- [virDomainPtr virDomainLookupByUUID](#) ([virConnectPtr](#) conn, const unsigned char \*uuid)
- [virDomainPtr virDomainLookupByUUIDString](#) ([virConnectPtr](#) conn, const char \*uuidstr)

**: name for the domain**

*virDomainLookupByName: : pointer to the hypervisor connection*

*Try to lookup a domain on the given hypervisor based on its name.*

*Returns a new domain object or NULL in case of failure. If the domain cannot be found, then VIR\_ERR\_NO\_DOMAIN error is raised.*

- [virDomainPtr virDomainLookupByName](#) ([virConnectPtr](#) conn, const char \*name)
- int [virDomainDestroy](#) ([virDomainPtr](#) domain)
- int [virDomainDestroyFlags](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainFree](#) ([virDomainPtr](#) domain)
- int [virDomainRef](#) ([virDomainPtr](#) domain)
- int [virDomainSuspend](#) ([virDomainPtr](#) domain)
- int [virDomainResume](#) ([virDomainPtr](#) domain)
- int [virDomainPMSuspendForDuration](#) ([virDomainPtr](#) dom, unsigned int target, unsigned long long duration, unsigned int flags)
- int [virDomainPMWakeup](#) ([virDomainPtr](#) dom, unsigned int flags)
- int [virDomainSave](#) ([virDomainPtr](#) domain, const char \*to)
- int [virDomainSaveFlags](#) ([virDomainPtr](#) domain, const char \*to, const char \*dxml, unsigned int flags)
- int [virDomainRestore](#) ([virConnectPtr](#) conn, const char \*from)
- int [virDomainRestoreFlags](#) ([virConnectPtr](#) conn, const char \*from, const char \*dxml, unsigned int flags)
- char \* [virDomainSaveImageGetXMLDesc](#) ([virConnectPtr](#) conn, const char \*file, unsigned int flags)
- int [virDomainSaveImageDefineXML](#) ([virConnectPtr](#) conn, const char \*file, const char \*dxml, unsigned int flags)
- int [virDomainCoreDump](#) ([virDomainPtr](#) domain, const char \*to, unsigned int flags)
- char \* [virDomainScreenshot](#) ([virDomainPtr](#) domain, [virStreamPtr](#) stream, unsigned int screen, unsigned int flags)
- int [virDomainShutdown](#) ([virDomainPtr](#) domain)
- int [virDomainShutdownFlags](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainReboot](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainReset](#) ([virDomainPtr](#) domain, unsigned int flags)

- `const char * virDomainGetName (virDomainPtr domain)`
- `int virDomainGetUUID (virDomainPtr domain, unsigned char *uuid)`
- `int virDomainGetUUIDString (virDomainPtr domain, char *buf)`
- `unsigned int virDomainGetID (virDomainPtr domain)`
- `char * virDomainGetOSType (virDomainPtr domain)`
- `unsigned long virDomainGetMaxMemory (virDomainPtr domain)`
- `int virDomainSetMaxMemory (virDomainPtr domain, unsigned long memory)`
- `int virDomainSetMemory (virDomainPtr domain, unsigned long memory)`
- `int virDomainSetMemoryFlags (virDomainPtr domain, unsigned long memory, unsigned int flags)`
- `static int virTypedParameterValidateSet (virConnectPtr conn, virTypedParameterPtr params, int nparams)`
- `int virDomainSetMemoryParameters (virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)`
- `int virDomainGetMemoryParameters (virDomainPtr domain, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- `int virDomainSetNumaParameters (virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)`
- `int virDomainGetNumaParameters (virDomainPtr domain, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- `int virDomainSetBlkioParameters (virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags)`
- `int virDomainGetBlkioParameters (virDomainPtr domain, virTypedParameterPtr params, int *nparams, unsigned int flags)`
- `int virDomainGetInfo (virDomainPtr domain, virDomainInfoPtr info)`
- `int virDomainGetState (virDomainPtr domain, int *state, int *reason, unsigned int flags)`
- `int virDomainGetControllInfo (virDomainPtr domain, virDomainControllInfoPtr info, unsigned int flags)`
- `char * virDomainGetXMLDesc (virDomainPtr domain, unsigned int flags)`
- `char * virConnectDomainXMLFromNative (virConnectPtr conn, const char *nativeFormat, const char *nativeConfig, unsigned int flags)`
- `char * virConnectDomainXMLToNative (virConnectPtr conn, const char *nativeFormat, const char *domainXml, unsigned int flags)`
- `static virDomainPtr virDomainMigrateVersion1 (virDomainPtr domain, virConnectPtr dconn, unsigned long flags, const char *dname, const char *uri, unsigned long bandwidth)`
- `static virDomainPtr virDomainMigrateVersion2 (virDomainPtr domain, virConnectPtr dconn, unsigned long flags, const char *dname, const char *uri, unsigned long bandwidth)`
- `static virDomainPtr virDomainMigrateVersion3 (virDomainPtr domain, virConnectPtr dconn, const char *xmlin, unsigned long flags, const char *dname, const char *uri, unsigned long bandwidth)`
- `static int virDomainMigratePeer2Peer (virDomainPtr domain, const char *xmlin, unsigned long flags, const char *dname, const char *dconnuri, const char *uri, unsigned long bandwidth)`
- `static int virDomainMigrateDirect (virDomainPtr domain, const char *xmlin, unsigned long flags, const char *dname, const char *uri, unsigned long bandwidth)`
- `virDomainPtr virDomainMigrate (virDomainPtr domain, virConnectPtr dconn, unsigned long flags, const char *dname, const char *uri, unsigned long bandwidth)`
- `virDomainPtr virDomainMigrate2 (virDomainPtr domain, virConnectPtr dconn, const char *dxml, unsigned long flags, const char *dname, const char *uri, unsigned long bandwidth)`
- `int virDomainMigrateToURI (virDomainPtr domain, const char *duri, unsigned long flags, const char *dname, unsigned long bandwidth)`
- `int virDomainMigrateToURI2 (virDomainPtr domain, const char *dconnuri, const char *miguri, const char *dxml, unsigned long flags, const char *dname, unsigned long bandwidth)`
- `int virDomainMigratePrepare (virConnectPtr dconn, char **cookie, int *cookielen, const char *uri_in, char **uri_out, unsigned long flags, const char *dname, unsigned long bandwidth)`
- `int virDomainMigratePerform (virDomainPtr domain, const char *cookie, int cookielen, const char *uri, unsigned long flags, const char *dname, unsigned long bandwidth)`
- `virDomainPtr virDomainMigrateFinish (virConnectPtr dconn, const char *dname, const char *cookie, int cookielen, const char *uri, unsigned long flags)`
- `int virDomainMigratePrepare2 (virConnectPtr dconn, char **cookie, int *cookielen, const char *uri_in, char **uri_out, unsigned long flags, const char *dname, unsigned long bandwidth, const char *dom_xml)`
- `virDomainPtr virDomainMigrateFinish2 (virConnectPtr dconn, const char *dname, const char *cookie, int cookielen, const char *uri, unsigned long flags, int retcode)`
- `int virDomainMigratePrepareTunnel (virConnectPtr conn, virStreamPtr st, unsigned long flags, const char *dname, unsigned long bandwidth, const char *dom_xml)`
- `char * virDomainMigrateBegin3 (virDomainPtr domain, const char *xmlin, char **cookieout, int *cookieoutlen, unsigned long flags, const char *dname, unsigned long bandwidth)`

- int [virDomainMigratePrepare3](#) ([virConnectPtr](#) dconn, const char \*cookiein, int cookieinlen, char \*\*cookieout, int \*cookieoutlen, const char \*uri\_in, char \*\*uri\_out, unsigned long flags, const char \*dname, unsigned long bandwidth, const char \*dom\_xml)
- int [virDomainMigratePrepareTunnel3](#) ([virConnectPtr](#) conn, [virStreamPtr](#) st, const char \*cookiein, int cookieinlen, char \*\*cookieout, int \*cookieoutlen, unsigned long flags, const char \*dname, unsigned long bandwidth, const char \*dom\_xml)
- int [virDomainMigratePerform3](#) ([virDomainPtr](#) domain, const char \*xmlin, const char \*cookiein, int cookieinlen, char \*\*cookieout, int \*cookieoutlen, const char \*dconnuri, const char \*uri, unsigned long flags, const char \*dname, unsigned long bandwidth)
- [virDomainPtr](#) [virDomainMigrateFinish3](#) ([virConnectPtr](#) dconn, const char \*dname, const char \*cookiein, int cookieinlen, char \*\*cookieout, int \*cookieoutlen, const char \*dconnuri, const char \*uri, unsigned long flags, int cancelled)
- int [virDomainMigrateConfirm3](#) ([virDomainPtr](#) domain, const char \*cookiein, int cookieinlen, unsigned long flags, int cancelled)
- int [virNodeGetInfo](#) ([virConnectPtr](#) conn, [virNodeInfoPtr](#) info)
- char \* [virConnectGetCapabilities](#) ([virConnectPtr](#) conn)
- int [virNodeGetCPUStats](#) ([virConnectPtr](#) conn, int cpuNum, [virNodeCPUStatsPtr](#) params, int \*nparams, unsigned int flags)
- int [virNodeGetMemoryStats](#) ([virConnectPtr](#) conn, int cellNum, [virNodeMemoryStatsPtr](#) params, int \*nparams, unsigned int flags)
- unsigned long long [virNodeGetFreeMemory](#) ([virConnectPtr](#) conn)
- int [virNodeSuspendForDuration](#) ([virConnectPtr](#) conn, unsigned int target, unsigned long long duration, unsigned int flags)
- int [virNodeGetMemoryParameters](#) ([virConnectPtr](#) conn, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- int [virNodeSetMemoryParameters](#) ([virConnectPtr](#) conn, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- char \* [virDomainGetSchedulerType](#) ([virDomainPtr](#) domain, int \*nparams)
- int [virDomainGetSchedulerParameters](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, int \*nparams)
- int [virDomainGetSchedulerParametersFlags](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- int [virDomainSetSchedulerParameters](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, int nparams)
- int [virDomainSetSchedulerParametersFlags](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- int [virDomainBlockStats](#) ([virDomainPtr](#) dom, const char \*disk, [virDomainBlockStatsPtr](#) stats, size\_t size)
- int [virDomainBlockStatsFlags](#) ([virDomainPtr](#) dom, const char \*disk, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- int [virDomainInterfaceStats](#) ([virDomainPtr](#) dom, const char \*path, [virDomainInterfaceStatsPtr](#) stats, size\_t size)
- int [virDomainSetInterfaceParameters](#) ([virDomainPtr](#) domain, const char \*device, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- int [virDomainGetInterfaceParameters](#) ([virDomainPtr](#) domain, const char \*device, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- int [virDomainMemoryStats](#) ([virDomainPtr](#) dom, [virDomainMemoryStatPtr](#) stats, unsigned int nr\_stats, unsigned int flags)
- int [virDomainBlockPeek](#) ([virDomainPtr](#) dom, const char \*disk, unsigned long long offset, size\_t size, void \*buffer, unsigned int flags)
- int [virDomainBlockResize](#) ([virDomainPtr](#) dom, const char \*disk, unsigned long long size, unsigned int flags)
- int [virDomainMemoryPeek](#) ([virDomainPtr](#) dom, unsigned long long start, size\_t size, void \*buffer, unsigned int flags)
- int [virDomainGetBlockInfo](#) ([virDomainPtr](#) domain, const char \*disk, [virDomainBlockInfoPtr](#) info, unsigned int flags)
- [virDomainPtr](#) [virDomainDefineXML](#) ([virConnectPtr](#) conn, const char \*xml)
- int [virDomainUndefine](#) ([virDomainPtr](#) domain)
- int [virDomainUndefineFlags](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virConnectNumOfDefinedDomains](#) ([virConnectPtr](#) conn)
- int [virConnectListDefinedDomains](#) ([virConnectPtr](#) conn, char \*\*const names, int maxnames)
- int [virConnectListAllDomains](#) ([virConnectPtr](#) conn, [virDomainPtr](#) \*\*domains, unsigned int flags)
- int [virDomainCreate](#) ([virDomainPtr](#) domain)
- int [virDomainCreateWithFlags](#) ([virDomainPtr](#) domain, unsigned int flags)

- int [virDomainGetAutostart](#) (virDomainPtr domain, int \*autostart)
- int [virDomainSetAutostart](#) (virDomainPtr domain, int autostart)
- int [virDomainInjectNMI](#) (virDomainPtr domain, unsigned int flags)
- int [virDomainSendKey](#) (virDomainPtr domain, unsigned int codeset, unsigned int holdtime, unsigned int \*keycodes, int nkeycodes, unsigned int flags)
- int [virDomainSetVcpus](#) (virDomainPtr domain, unsigned int nvcpus)
- int [virDomainSetVcpusFlags](#) (virDomainPtr domain, unsigned int nvcpus, unsigned int flags)
- int [virDomainGetVcpusFlags](#) (virDomainPtr domain, unsigned int flags)
- int [virDomainPinVcpu](#) (virDomainPtr domain, unsigned int vcpu, unsigned char \*cpumap, int maplen)
- int [virDomainPinVcpuFlags](#) (virDomainPtr domain, unsigned int vcpu, unsigned char \*cpumap, int maplen, unsigned int flags)
- int [virDomainGetVcpuPinInfo](#) (virDomainPtr domain, int ncpumaps, unsigned char \*cpumaps, int maplen, unsigned int flags)
- int [virDomainPinEmulator](#) (virDomainPtr domain, unsigned char \*cpumap, int maplen, unsigned int flags)
- int [virDomainGetEmulatorPinInfo](#) (virDomainPtr domain, unsigned char \*cpumap, int maplen, unsigned int flags)
- int [virDomainGetVcpus](#) (virDomainPtr domain, virVcpuInfoPtr info, int maxinfo, unsigned char \*cpumaps, int maplen)
- int [virDomainGetMaxVcpus](#) (virDomainPtr domain)
- int [virDomainGetSecurityLabel](#) (virDomainPtr domain, virSecurityLabelPtr seclabel)
- int [virDomainGetSecurityLabelList](#) (virDomainPtr domain, virSecurityLabelPtr \*seclabels)
- int [virDomainSetMetadata](#) (virDomainPtr domain, int type, const char \*metadata, const char \*key, const char \*uri, unsigned int flags)
- char \* [virDomainGetMetadata](#) (virDomainPtr domain, int type, const char \*uri, unsigned int flags)
- int [virNodeGetSecurityModel](#) (virConnectPtr conn, virSecurityModelPtr secmodel)
- int [virDomainAttachDevice](#) (virDomainPtr domain, const char \*xml)
- int [virDomainAttachDeviceFlags](#) (virDomainPtr domain, const char \*xml, unsigned int flags)
- int [virDomainDetachDevice](#) (virDomainPtr domain, const char \*xml)
- int [virDomainDetachDeviceFlags](#) (virDomainPtr domain, const char \*xml, unsigned int flags)
- int [virDomainUpdateDeviceFlags](#) (virDomainPtr domain, const char \*xml, unsigned int flags)
- int [virNodeGetCellsFreeMemory](#) (virConnectPtr conn, unsigned long long \*freeMems, int startCell, int maxCells)
- virConnectPtr [virNetworkGetConnect](#) (virNetworkPtr net)
- int [virConnectListAllNetworks](#) (virConnectPtr conn, virNetworkPtr \*\*nets, unsigned int flags)
- int [virConnectNumOfNetworks](#) (virConnectPtr conn)
- int [virConnectListNetworks](#) (virConnectPtr conn, char \*\*const names, int maxnames)
- int [virConnectNumOfDefinedNetworks](#) (virConnectPtr conn)
- int [virConnectListDefinedNetworks](#) (virConnectPtr conn, char \*\*const names, int maxnames)

#### : name for the network

*virNetworkLookupByName*: : pointer to the hypervisor connection

Try to lookup a network on the given hypervisor based on its name.

Returns a new network object or NULL in case of failure. If the network cannot be found, then VIR\_ERR\_NO\_NETWORK error is raised.

- virNetworkPtr [virNetworkLookupByName](#) (virConnectPtr conn, const char \*name)
- virNetworkPtr [virNetworkLookupByUUID](#) (virConnectPtr conn, const unsigned char \*uuid)
- virNetworkPtr [virNetworkLookupByUUIDString](#) (virConnectPtr conn, const char \*uuidstr)
- virNetworkPtr [virNetworkCreateXML](#) (virConnectPtr conn, const char \*xmlDesc)
- virNetworkPtr [virNetworkDefineXML](#) (virConnectPtr conn, const char \*xml)
- int [virNetworkUndefine](#) (virNetworkPtr network)
- int [virNetworkUpdate](#) (virNetworkPtr network, unsigned int command, unsigned int section, int parentIndex, const char \*xml, unsigned int flags)
- int [virNetworkCreate](#) (virNetworkPtr network)
- int [virNetworkDestroy](#) (virNetworkPtr network)
- int [virNetworkFree](#) (virNetworkPtr network)
- int [virNetworkRef](#) (virNetworkPtr network)
- const char \* [virNetworkGetName](#) (virNetworkPtr network)
- int [virNetworkGetUUID](#) (virNetworkPtr network, unsigned char \*uuid)
- int [virNetworkGetUUIDString](#) (virNetworkPtr network, char \*buf)
- char \* [virNetworkGetXMLDesc](#) (virNetworkPtr network, unsigned int flags)
- char \* [virNetworkGetBridgeName](#) (virNetworkPtr network)



- **POL New** char \* virNetworkGetBridgeType (virNetworkPtr network)
- int virNetworkGetAutostart (virNetworkPtr network, int \*autostart)
- int virNetworkSetAutostart (virNetworkPtr network, int autostart)
- virConnectPtr virInterfaceGetConnect (virInterfacePtr iface)
- int virConnectListAllInterfaces (virConnectPtr conn, virInterfacePtr \*\*ifaces, unsigned int flags)
- int virConnectNumOfInterfaces (virConnectPtr conn)
- int virConnectListInterfaces (virConnectPtr conn, char \*\*const names, int maxnames)
- int virConnectNumOfDefinedInterfaces (virConnectPtr conn)
- int virConnectListDefinedInterfaces (virConnectPtr conn, char \*\*const names, int maxnames)

#### : name for the interface

virInterfaceLookupByName: : pointer to the hypervisor connection

Try to lookup an interface on the given hypervisor based on its name.

Returns a new interface object or NULL in case of failure. If the interface cannot be found, then VIR\_ERR\_NO\_INTERFACE error is raised.

- virInterfacePtr virInterfaceLookupByName (virConnectPtr conn, const char \*name)
- virInterfacePtr virInterfaceLookupByMACString (virConnectPtr conn, const char \*macstr)
- const char \* virInterfaceGetName (virInterfacePtr iface)
- const char \* virInterfaceGetMACString (virInterfacePtr iface)
- char \* virInterfaceGetXMLDesc (virInterfacePtr iface, unsigned int flags)
- virInterfacePtr virInterfaceDefineXML (virConnectPtr conn, const char \*xml, unsigned int flags)
- int virInterfaceUndefine (virInterfacePtr iface)
- int virInterfaceCreate (virInterfacePtr iface, unsigned int flags)
- int virInterfaceDestroy (virInterfacePtr iface, unsigned int flags)
- int virInterfaceRef (virInterfacePtr iface)
- int virInterfaceFree (virInterfacePtr iface)
- int virInterfaceChangeBegin (virConnectPtr conn, unsigned int flags)
- int virInterfaceChangeCommit (virConnectPtr conn, unsigned int flags)
- int virInterfaceChangeRollback (virConnectPtr conn, unsigned int flags)
- virConnectPtr virStoragePoolGetConnect (virStoragePoolPtr pool)
- int virConnectListAllStoragePools (virConnectPtr conn, virStoragePoolPtr \*\*pools, unsigned int flags)
- int virConnectNumOfStoragePools (virConnectPtr conn)
- int virConnectListStoragePools (virConnectPtr conn, char \*\*const names, int maxnames)
- int virConnectNumOfDefinedStoragePools (virConnectPtr conn)
- int virConnectListDefinedStoragePools (virConnectPtr conn, char \*\*const names, int maxnames)
- char \* virConnectFindStoragePoolSources (virConnectPtr conn, const char \*type, const char \*srcSpec, unsigned int flags)

#### : name of pool to fetch

virStoragePoolLookupByName: : pointer to hypervisor connection

Fetch a storage pool based on its unique name

Returns a virStoragePoolPtr object, or NULL if no matching pool is found

- virStoragePoolPtr virStoragePoolLookupByName (virConnectPtr conn, const char \*name)
- virStoragePoolPtr virStoragePoolLookupByUUID (virConnectPtr conn, const unsigned char \*uuid)
- virStoragePoolPtr virStoragePoolLookupByUUIDString (virConnectPtr conn, const char \*uuidstr)
- virStoragePoolPtr virStoragePoolLookupByVolume (virStorageVolPtr vol)
- virStoragePoolPtr virStoragePoolCreateXML (virConnectPtr conn, const char \*xmlDesc, unsigned int flags)
- virStoragePoolPtr virStoragePoolDefineXML (virConnectPtr conn, const char \*xml, unsigned int flags)
- int virStoragePoolBuild (virStoragePoolPtr pool, unsigned int flags)
- int virStoragePoolUndefine (virStoragePoolPtr pool)
- int virStoragePoolCreate (virStoragePoolPtr pool, unsigned int flags)
- int virStoragePoolDestroy (virStoragePoolPtr pool)
- int virStoragePoolDelete (virStoragePoolPtr pool, unsigned int flags)
- int virStoragePoolFree (virStoragePoolPtr pool)
- int virStoragePoolRef (virStoragePoolPtr pool)
- int virStoragePoolRefresh (virStoragePoolPtr pool, unsigned int flags)
- const char \* virStoragePoolGetName (virStoragePoolPtr pool)
- int virStoragePoolGetUUID (virStoragePoolPtr pool, unsigned char \*uuid)



- int [virStoragePoolGetUUIDString](#) (virStoragePoolPtr pool, char \*buf)
- int [virStoragePoolGetInfo](#) (virStoragePoolPtr pool, [virStoragePoolInfoPtr](#) info)
- char \* [virStoragePoolGetXMLDesc](#) (virStoragePoolPtr pool, unsigned int flags)
- int [virStoragePoolGetAutostart](#) (virStoragePoolPtr pool, int \*autostart)
- int [virStoragePoolSetAutostart](#) (virStoragePoolPtr pool, int autostart)
- int [virStoragePoolListAllVolumes](#) (virStoragePoolPtr pool, [virStorageVolPtr](#) \*\*vols, unsigned int flags)
- int [virStoragePoolNumOfVolumes](#) (virStoragePoolPtr pool)
- int [virStoragePoolListVolumes](#) (virStoragePoolPtr pool, char \*\*const names, int maxnames)
- [virConnectPtr](#) [virStorageVolGetConnect](#) (virStorageVolPtr vol)

#### : name of storage volume

*virStorageVolLookupByName*: : pointer to storage pool

Fetch a pointer to a storage volume based on its name within a pool

Returns a storage volume, or NULL if not found / error

- [virStorageVolPtr](#) [virStorageVolLookupByName](#) (virStoragePoolPtr pool, const char \*name)
- [virStorageVolPtr](#) [virStorageVolLookupByKey](#) (virConnectPtr conn, const char \*key)
- [virStorageVolPtr](#) [virStorageVolLookupByPath](#) (virConnectPtr conn, const char \*path)
- const char \* [virStorageVolGetName](#) (virStorageVolPtr vol)
- const char \* [virStorageVolGetKey](#) (virStorageVolPtr vol)
- [virStorageVolPtr](#) [virStorageVolCreateXML](#) (virStoragePoolPtr pool, const char \*xmldesc, unsigned int flags)
- [virStorageVolPtr](#) [virStorageVolCreateXMLFrom](#) (virStoragePoolPtr pool, const char \*xmldesc, [virStorageVolPtr](#) clonevol, unsigned int flags)
- int [virStorageVolDownload](#) (virStorageVolPtr vol, [virStreamPtr](#) stream, unsigned long long offset, unsigned long long length, unsigned int flags)
- int [virStorageVolUpload](#) (virStorageVolPtr vol, [virStreamPtr](#) stream, unsigned long long offset, unsigned long long length, unsigned int flags)
- int [virStorageVolDelete](#) (virStorageVolPtr vol, unsigned int flags)
- int [virStorageVolWipe](#) (virStorageVolPtr vol, unsigned int flags)
- int [virStorageVolWipePattern](#) (virStorageVolPtr vol, unsigned int algorithm, unsigned int flags)
- int [virStorageVolFree](#) (virStorageVolPtr vol)
- int [virStorageVolRef](#) (virStorageVolPtr vol)
- int [virStorageVolGetInfo](#) (virStorageVolPtr vol, [virStorageVolInfoPtr](#) info)
- char \* [virStorageVolGetXMLDesc](#) (virStorageVolPtr vol, unsigned int flags)
- char \* [virStorageVolGetPath](#) (virStorageVolPtr vol)
- int [virStorageVolResize](#) (virStorageVolPtr vol, unsigned long long capacity, unsigned int flags)
- int [virNodeNumOfDevices](#) (virConnectPtr conn, const char \*cap, unsigned int flags)
- int [virConnectListAllNodeDevices](#) (virConnectPtr conn, [virNodeDevicePtr](#) \*\*devices, unsigned int flags)
- int [virNodeListDevices](#) (virConnectPtr conn, const char \*cap, char \*\*const names, int maxnames, unsigned int flags)

#### : unique device name

*virNodeDeviceLookupByName*: : pointer to the hypervisor connection

Lookup a node device by its name.

Returns a [virNodeDevicePtr](#) if found, NULL otherwise.

- [virNodeDevicePtr](#) [virNodeDeviceLookupByName](#) (virConnectPtr conn, const char \*name)
- char \* [virNodeDeviceGetXMLDesc](#) (virNodeDevicePtr dev, unsigned int flags)
- const char \* [virNodeDeviceGetName](#) (virNodeDevicePtr dev)
- const char \* [virNodeDeviceGetParent](#) (virNodeDevicePtr dev)
- int [virNodeDeviceNumOfCaps](#) (virNodeDevicePtr dev)
- int [virNodeDeviceListCaps](#) (virNodeDevicePtr dev, char \*\*const names, int maxnames)
- int [virNodeDeviceFree](#) (virNodeDevicePtr dev)
- int [virNodeDeviceRef](#) (virNodeDevicePtr dev)
- int [virNodeDeviceDetach](#) (virNodeDevicePtr dev)
- int [virNodeDeviceReAttach](#) (virNodeDevicePtr dev)
- int [virNodeDeviceReset](#) (virNodeDevicePtr dev)
- [virNodeDevicePtr](#) [virNodeDeviceCreateXML](#) (virConnectPtr conn, const char \*xmlDesc, unsigned int flags)

- int `virNodeDeviceDestroy` (`virNodeDevicePtr` dev)
- int `virConnectDomainEventRegister` (`virConnectPtr` conn, `virConnectDomainEventCallback` cb, void \*opaque, `virFreeCallback` freecb)
- int `virConnectDomainEventDeregister` (`virConnectPtr` conn, `virConnectDomainEventCallback` cb)
- `virConnectPtr` `virSecretGetConnect` (`virSecretPtr` secret)
- int `virConnectNumOfSecrets` (`virConnectPtr` conn)
- int `virConnectListAllSecrets` (`virConnectPtr` conn, `virSecretPtr` \*\*secrets, unsigned int flags)
- int `virConnectListSecrets` (`virConnectPtr` conn, char \*\*uuids, int maxuuids)
- `virSecretPtr` `virSecretLookupByUUID` (`virConnectPtr` conn, const unsigned char \*uuid)
- `virSecretPtr` `virSecretLookupByUUIDString` (`virConnectPtr` conn, const char \*uuidstr)
- `virSecretPtr` `virSecretLookupByUsage` (`virConnectPtr` conn, int usageType, const char \*usageID)
- `virSecretPtr` `virSecretDefineXML` (`virConnectPtr` conn, const char \*xml, unsigned int flags)
- int `virSecretGetUUID` (`virSecretPtr` secret, unsigned char \*uuid)
- int `virSecretGetUUIDString` (`virSecretPtr` secret, char \*buf)
- int `virSecretGetUsageType` (`virSecretPtr` secret)
- const char \* `virSecretGetUsageID` (`virSecretPtr` secret)
- char \* `virSecretGetXMLDesc` (`virSecretPtr` secret, unsigned int flags)
- int `virSecretSetValue` (`virSecretPtr` secret, const unsigned char \*value, size\_t value\_size, unsigned int flags)
- unsigned char \* `virSecretGetValue` (`virSecretPtr` secret, size\_t \*value\_size, unsigned int flags)
- int `virSecretUndefine` (`virSecretPtr` secret)
- int `virSecretRef` (`virSecretPtr` secret)
- int `virSecretFree` (`virSecretPtr` secret)
- `virStreamPtr` `virStreamNew` (`virConnectPtr` conn, unsigned int flags)
- int `virStreamRef` (`virStreamPtr` stream)
- int `virStreamSend` (`virStreamPtr` stream, const char \*data, size\_t nbytes)
- int `virStreamRecv` (`virStreamPtr` stream, char \*data, size\_t nbytes)
- int `virStreamSendAll` (`virStreamPtr` stream, `virStreamSourceFunc` handler, void \*opaque)
- int `virStreamRecvAll` (`virStreamPtr` stream, `virStreamSinkFunc` handler, void \*opaque)
- int `virStreamEventAddCallback` (`virStreamPtr` stream, int events, `virStreamEventCallback` cb, void \*opaque, `virFreeCallback` ff)
- int `virStreamEventUpdateCallback` (`virStreamPtr` stream, int events)
- int `virStreamEventRemoveCallback` (`virStreamPtr` stream)
- int `virStreamFinish` (`virStreamPtr` stream)
- int `virStreamAbort` (`virStreamPtr` stream)
- int `virStreamFree` (`virStreamPtr` stream)
- int `virDomainsActive` (`virDomainPtr` dom)
- int `virDomainsPersistent` (`virDomainPtr` dom)
- int `virDomainsUpdated` (`virDomainPtr` dom)
- int `virNetworksActive` (`virNetworkPtr` net)
- int `virNetworksPersistent` (`virNetworkPtr` net)
- int `virStoragePoolsActive` (`virStoragePoolPtr` pool)
- int `virStoragePoolsPersistent` (`virStoragePoolPtr` pool)
- int `virConnectNumOfNWFilters` (`virConnectPtr` conn)
- int `virConnectListAllNWFilters` (`virConnectPtr` conn, `virNWFilterPtr` \*\*filters, unsigned int flags)
- int `virConnectListNWFilters` (`virConnectPtr` conn, char \*\*const names, int maxnames)

#### : name for the network filter

`virNWFilterLookupByName`: : pointer to the hypervisor connection

Try to lookup a network filter on the given hypervisor based on its name.

Returns a new nwfilter object or NULL in case of failure. If the network filter cannot be found, then `VIR_ERR_NO_NWFILTER` error is raised.

- `virNWFilterPtr` `virNWFilterLookupByName` (`virConnectPtr` conn, const char \*name)
- `virNWFilterPtr` `virNWFilterLookupByUUID` (`virConnectPtr` conn, const unsigned char \*uuid)
- `virNWFilterPtr` `virNWFilterLookupByUUIDString` (`virConnectPtr` conn, const char \*uuidstr)
- int `virNWFilterFree` (`virNWFilterPtr` nwfilter)
- const char \* `virNWFilterGetName` (`virNWFilterPtr` nwfilter)
- int `virNWFilterGetUUID` (`virNWFilterPtr` nwfilter, unsigned char \*uuid)
- int `virNWFilterGetUUIDString` (`virNWFilterPtr` nwfilter, char \*buf)
- `virNWFilterPtr` `virNWFilterDefineXML` (`virConnectPtr` conn, const char \*xmlDesc)
- int `virNWFilterUndefine` (`virNWFilterPtr` nwfilter)

- char \* [virNWFilterGetXMLDesc](#) ([virNWFilterPtr](#) nwfilter, unsigned int flags)
- int [virNWFilterRef](#) ([virNWFilterPtr](#) nwfilter)
- int [virInterfaceIsActive](#) ([virInterfacePtr](#) iface)
- int [virConnectIsEncrypted](#) ([virConnectPtr](#) conn)
- int [virConnectIsSecure](#) ([virConnectPtr](#) conn)
- int [virConnectCompareCPU](#) ([virConnectPtr](#) conn, const char \*xmlDesc, unsigned int flags)
- char \* [virConnectBaselineCPU](#) ([virConnectPtr](#) conn, const char \*\*xmlCPUs, unsigned int ncpus, unsigned int flags)
- int [virDomainGetJobInfo](#) ([virDomainPtr](#) domain, [virDomainJobInfoPtr](#) info)
- int [virDomainAbortJob](#) ([virDomainPtr](#) domain)
- int [virDomainMigrateSetMaxDowntime](#) ([virDomainPtr](#) domain, unsigned long long downtime, unsigned int flags)
- int [virDomainMigrateSetMaxSpeed](#) ([virDomainPtr](#) domain, unsigned long bandwidth, unsigned int flags)
- int [virDomainMigrateGetMaxSpeed](#) ([virDomainPtr](#) domain, unsigned long \*bandwidth, unsigned int flags)
- int [virConnectDomainEventRegisterAny](#) ([virConnectPtr](#) conn, [virDomainPtr](#) dom, int eventID, [virConnectDomainEventGenericCallback](#) cb, void \*opaque, [virFreeCallback](#) freecb)
- int [virConnectDomainEventDeregisterAny](#) ([virConnectPtr](#) conn, int callbackID)
- int [virDomainManagedSave](#) ([virDomainPtr](#) dom, unsigned int flags)
- int [virDomainHasManagedSaveImage](#) ([virDomainPtr](#) dom, unsigned int flags)
- int [virDomainManagedSaveRemove](#) ([virDomainPtr](#) dom, unsigned int flags)
- const char \* [virDomainSnapshotGetName](#) ([virDomainSnapshotPtr](#) snapshot)
- [virDomainPtr](#) [virDomainSnapshotGetDomain](#) ([virDomainSnapshotPtr](#) snapshot)
- [virConnectPtr](#) [virDomainSnapshotGetConnect](#) ([virDomainSnapshotPtr](#) snapshot)
- [virDomainSnapshotPtr](#) [virDomainSnapshotCreateXML](#) ([virDomainPtr](#) domain, const char \*xmlDesc, unsigned int flags)
- char \* [virDomainSnapshotGetXMLDesc](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotNum](#) ([virDomainPtr](#) domain, unsigned int flags)
- int [virDomainSnapshotListNames](#) ([virDomainPtr](#) domain, char \*\*names, int nameslen, unsigned int flags)
- int [virDomainListAllSnapshots](#) ([virDomainPtr](#) domain, [virDomainSnapshotPtr](#) \*\*snaps, unsigned int flags)
- int [virDomainSnapshotNumChildren](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotListChildrenNames](#) ([virDomainSnapshotPtr](#) snapshot, char \*\*names, int nameslen, unsigned int flags)
- int [virDomainSnapshotListAllChildren](#) ([virDomainSnapshotPtr](#) snapshot, [virDomainSnapshotPtr](#) \*\*snaps, unsigned int flags)

#### : name for the domain snapshot

*virDomainSnapshotLookupByName*: : a domain object

: extra flags; not used yet, so callers should always pass 0

Try to lookup a domain snapshot based on its name.

Returns a domain snapshot object or NULL in case of failure. If the domain snapshot cannot be found, then the `VIR_ERR_NO_DOMAIN_SNAPSHOT` error is raised.

- [virDomainSnapshotPtr](#) [virDomainSnapshotLookupByName](#) ([virDomainPtr](#) domain, const char \*name, unsigned int flags)
- int [virDomainHasCurrentSnapshot](#) ([virDomainPtr](#) domain, unsigned int flags)
- [virDomainSnapshotPtr](#) [virDomainSnapshotCurrent](#) ([virDomainPtr](#) domain, unsigned int flags)
- [virDomainSnapshotPtr](#) [virDomainSnapshotGetParent](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotsCurrent](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotHasMetadata](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainRevertToSnapshot](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotDelete](#) ([virDomainSnapshotPtr](#) snapshot, unsigned int flags)
- int [virDomainSnapshotRef](#) ([virDomainSnapshotPtr](#) snapshot)
- int [virDomainSnapshotFree](#) ([virDomainSnapshotPtr](#) snapshot)
- int [virDomainOpenConsole](#) ([virDomainPtr](#) dom, const char \*dev\_name, [virStreamPtr](#) st, unsigned int flags)
- int [virDomainBlockJobAbort](#) ([virDomainPtr](#) dom, const char \*disk, unsigned int flags)
- int [virDomainGetBlockJobInfo](#) ([virDomainPtr](#) dom, const char \*disk, [virDomainBlockJobInfoPtr](#) info, unsigned int flags)
- int [virDomainBlockJobSetSpeed](#) ([virDomainPtr](#) dom, const char \*disk, unsigned long bandwidth, unsigned int flags)
- int [virDomainBlockPull](#) ([virDomainPtr](#) dom, const char \*disk, unsigned long bandwidth, unsigned int flags)

- int [virDomainBlockRebase](#) ([virDomainPtr](#) dom, const char \*disk, const char \*base, unsigned long bandwidth, unsigned int flags)
- int [virDomainBlockCommit](#) ([virDomainPtr](#) dom, const char \*disk, const char \*base, const char \*top, unsigned long bandwidth, unsigned int flags)
- int [virDomainOpenGraphics](#) ([virDomainPtr](#) dom, unsigned int idx, int fd, unsigned int flags)
- int [virConnectSetKeepAlive](#) ([virConnectPtr](#) conn, int interval, unsigned int count)
- int [virConnectIsAlive](#) ([virConnectPtr](#) conn)
- int [virConnectRegisterCloseCallback](#) ([virConnectPtr](#) conn, [virConnectCloseFunc](#) cb, void \*opaque, [virFreeCallback](#) freecb)
- int [virConnectUnregisterCloseCallback](#) ([virConnectPtr](#) conn, [virConnectCloseFunc](#) cb)
- int [virDomainSetBlockioTune](#) ([virDomainPtr](#) dom, const char \*disk, [virTypedParameterPtr](#) params, int nparams, unsigned int flags)
- int [virDomainGetBlockioTune](#) ([virDomainPtr](#) dom, const char \*disk, [virTypedParameterPtr](#) params, int \*nparams, unsigned int flags)
- int [virDomainGetCPUStats](#) ([virDomainPtr](#) domain, [virTypedParameterPtr](#) params, unsigned int nparams, int start\_cpu, unsigned int ncpus, unsigned int flags)
- int [virDomainGetDiskErrors](#) ([virDomainPtr](#) dom, [virDomainDiskErrorPtr](#) errors, unsigned int maxerrors, unsigned int flags)
- char \* [virDomainGetHostname](#) ([virDomainPtr](#) domain, unsigned int flags)

## Variables

- static [virDriverPtr](#) [virDriverTab](#) [[MAX\\_DRIVERS](#)]
- static int [virDriverTabCount](#) = 0
- static [virNetworkDriverPtr](#) [virNetworkDriverTab](#) [[MAX\\_DRIVERS](#)]
- static int [virNetworkDriverTabCount](#) = 0
- static [virInterfaceDriverPtr](#) [virInterfaceDriverTab](#) [[MAX\\_DRIVERS](#)]
- static int [virInterfaceDriverTabCount](#) = 0
- static [virStorageDriverPtr](#) [virStorageDriverTab](#) [[MAX\\_DRIVERS](#)]
- static int [virStorageDriverTabCount](#) = 0
- static [virDeviceMonitorPtr](#) [virDeviceMonitorTab](#) [[MAX\\_DRIVERS](#)]
- static int [virDeviceMonitorTabCount](#) = 0
- static [virSecretDriverPtr](#) [virSecretDriverTab](#) [[MAX\\_DRIVERS](#)]
- static int [virSecretDriverTabCount](#) = 0
- static [virNWFilterDriverPtr](#) [virNWFilterDriverTab](#) [[MAX\\_DRIVERS](#)]
- static int [virNWFilterDriverTabCount](#) = 0
- static int [initialized](#) = 0
- static int [virConnectCredTypeDefault](#) []
- static [virConnectAuth](#) [virConnectAuthDefault](#)
- [virConnectAuthPtr](#) [virConnectAuthPtrDefault](#) = &[virConnectAuthDefault](#)
- static struct gcry\_thread\_cbs [virTLSThreadImpl](#)

## 4.7.1 Macro Definition Documentation

4.7.1.1 `#define MAX_DRIVERS 20`

4.7.1.2 `#define URI_ALIAS_CHARS "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_-"`

4.7.1.3 `#define VIR_ARG15( _1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, ... ) _15`

4.7.1.4 `#define VIR_DOMAIN_DEBUG( ... )`

### Value:

```
VIR_DOMAIN_DEBUG_EXPAND (VIR_DOMAIN_DEBUG_,
                          VIR_HAS_COMMA ( __VA_ARGS__ ), \
                          __VA_ARGS__)
```

VIR\_DOMAIN\_DEBUG: : domain : optional format for additional information ...: optional arguments corresponding to .

4.7.1.5 #define VIR\_DOMAIN\_DEBUG\_0( dom ) VIR\_DOMAIN\_DEBUG\_2(dom, "%s", "")

4.7.1.6 #define VIR\_DOMAIN\_DEBUG\_1( dom, fmt, ... ) VIR\_DOMAIN\_DEBUG\_2(dom, ", " fmt, \_\_VA\_ARGS\_\_)

4.7.1.7 #define VIR\_DOMAIN\_DEBUG\_2( dom, fmt, ... )

**Value:**

```
do {
    char _uuidstr[VIR_UUID_STRING_BUFLen];
    const char *_domname = NULL;

    if (!VIR_IS_DOMAIN(dom)) {
        memset(_uuidstr, 0, sizeof(_uuidstr));
    } else {
        virUUIDFormat((dom)->uuid, _uuidstr);
        _domname = (dom)->name;
    }

    VIR_DEBUG("dom=%p, (VM: name=%s, uuid=%s)" fmt,
              dom, NULLSTR(_domname), _uuidstr, __VA_ARGS__);
} while (0)
```

4.7.1.8 #define VIR\_DOMAIN\_DEBUG\_EXPAND( a, b, ... ) VIR\_DOMAIN\_DEBUG\_PASTE(a, b, \_\_VA\_ARGS\_\_)

4.7.1.9 #define VIR\_DOMAIN\_DEBUG\_PASTE( a, b, ... ) a##b(\_\_VA\_ARGS\_\_)

4.7.1.10 #define VIR\_FROM\_THIS VIR\_FROM\_NONE

4.7.1.11 #define VIR\_HAS\_COMMA( ... ) VIR\_ARG15(\_\_VA\_ARGS\_\_, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)

4.7.1.12 #define VIR\_UUID\_DEBUG( conn, uuid )

**Value:**

```
do {
    if (uuid) {
        char _uuidstr[VIR_UUID_STRING_BUFLen];
        virUUIDFormat(uuid, _uuidstr);
        VIR_DEBUG("conn=%p, uuid=%s", conn, _uuidstr);
    } else {
        VIR_DEBUG("conn=%p, uuid=(null)", conn);
    }
} while (0)
```

VIR\_UUID\_DEBUG: : connection : possibly null UUID array

4.7.1.13 #define virLibConnError( code, ... )

**Value:**

```
virReportErrorHelper(VIR_FROM_NONE, code, __FILE__,
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

4.7.1.14 #define virLibDomainError( code, ... )

**Value:**

```
virReportErrorHelper(VIR_FROM_DOM, code, __FILE__,
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

**4.7.1.15 #define virLibDomainSnapshotError( code, ... )****Value:**

```
virReportErrorHelper(VIR_FROM_DOMAIN_SNAPSHOT, code, __FILE__, \
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

**4.7.1.16 #define virLibInterfaceError( code, ... )****Value:**

```
virReportErrorHelper(VIR_FROM_INTERFACE, code, __FILE__, \
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

**4.7.1.17 #define virLibNetworkError( code, ... )****Value:**

```
virReportErrorHelper(VIR_FROM_NETWORK, code, __FILE__, \
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

**4.7.1.18 #define virLibNodeDeviceError( code, ... )****Value:**

```
virReportErrorHelper(VIR_FROM_NODEDEV, code, __FILE__, \
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

**4.7.1.19 #define virLibNWFilterError( code, ... )****Value:**

```
virReportErrorHelper(VIR_FROM_NWFILTER, code, __FILE__, \
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

**4.7.1.20 #define virLibSecretError( code, ... )****Value:**

```
virReportErrorHelper(VIR_FROM_SECRET, code, __FILE__, \
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

**4.7.1.21 #define virLibStoragePoolError( code, ... )****Value:**

```
virReportErrorHelper(VIR_FROM_STORAGE, code, __FILE__, \
                    __FUNCTION__, __LINE__, __VA_ARGS__)
```

## 4.7.1.22 #define virLibStorageVolError( code, ... )

## Value:

```
virReportErrorHelper(VIR_FROM_STORAGE, code, __FILE__,
                    __FUNCTION__, __LINE__, __VA_ARGS__) \
```

## 4.7.1.23 #define virLibStreamError( code, ... )

## Value:

```
virReportErrorHelper(VIR_FROM_STREAMS, code, __FILE__,
                    __FUNCTION__, __LINE__, __VA_ARGS__) \
```

## 4.7.2 Function Documentation

## 4.7.2.1 static virConnectPtr do\_open ( const char \* name, virConnectAuthPtr auth, unsigned int flags ) [static]

## 4.7.2.2 static int virConnectAuthCallbackDefault ( virConnectCredentialPtr cred, unsigned int ncred, void \*cbdata ATTRIBUTE\_UNUSED ) [static]

## 4.7.2.3 char\* virConnectBaselineCPU ( virConnectPtr conn, const char \*\* xmlCPUs, unsigned int ncpus, unsigned int flags )

virConnectBaselineCPU:

: virConnect connection : array of XML descriptions of host CPUs : number of CPUs in xmlCPUs : extra flags; not used yet, so callers should always pass 0

Computes the most feature-rich CPU which is compatible with all given host CPUs.

Returns XML description of the computed CPU or NULL on error.

## 4.7.2.4 int virConnectClose ( virConnectPtr conn )

virConnectClose: : pointer to the hypervisor connection

This function closes the connection to the Hypervisor. This should not be called if further interaction with the Hypervisor are needed especially if there is running domain which need further monitoring by the application.

Connections are reference counted; the count is explicitly increased by the initial open (virConnectOpen, virConnectOpenAuth, and the like) as well as virConnectRef; it is also temporarily increased by other API that depend on the connection remaining alive. The open and every virConnectRef call should have a matching virConnectClose, and all other references will be released after the corresponding operation completes.

Returns a positive number if at least 1 reference remains on success. The returned value should not be assumed to be the total reference count. A return of 0 implies no references remain and the connection is closed and memory has been freed. A return of -1 implies a failure.

It is possible for the last virConnectClose to return a positive value if some other object still has a temporary reference to the connection, but the application should not try to further use a connection after the virConnectClose that matches the initial open.

## 4.7.2.5 int virConnectCompareCPU ( virConnectPtr conn, const char \* xmlDesc, unsigned int flags )

virConnectCompareCPU: : virConnect connection : XML describing the CPU to compare with host CPU : extra flags; not used yet, so callers should always pass 0

Compares the given CPU description with the host CPU

Returns comparison result according to enum virCPUCompareResult

#### 4.7.2.6 `int virConnectDomainEventDeregister ( virConnectPtr conn, virConnectDomainEventCallback cb )`

`virConnectDomainEventDeregister`: : pointer to the connection : callback to the function handling domain events

Removes a callback previously registered with the `virConnectDomainEventRegister` function.

Use of this method is no longer recommended. Instead applications should try `virConnectDomainEventUnregisterAny` which has a more flexible API contract

Returns 0 on success, -1 on failure

#### 4.7.2.7 `int virConnectDomainEventDeregisterAny ( virConnectPtr conn, int callbackID )`

`virConnectDomainEventDeregisterAny`: : pointer to the connection : the callback identifier

Removes an event callback. The `callbackID` parameter should be the value obtained from a previous `virDomainEventRegisterAny` method.

Returns 0 on success, -1 on failure

#### 4.7.2.8 `int virConnectDomainEventRegister ( virConnectPtr conn, virConnectDomainEventCallback cb, void * opaque, virFreeCallback freecb )`

`virConnectDomainEventRegister`: : pointer to the connection : callback to the function handling domain events : opaque data to pass on to the callback : optional function to deallocate opaque when not used anymore

Adds a callback to receive notifications of domain lifecycle events occurring on a connection

Use of this method is no longer recommended. Instead applications should try `virConnectDomainEventRegisterAny` which has a more flexible API contract

The `virDomainPtr` object handle passed into the callback upon delivery of an event is only valid for the duration of execution of the callback. If the callback wishes to keep the domain object after the callback returns, it shall take a reference to it, by calling `virDomainRef`. The reference can be released once the object is no longer required by calling `virDomainFree`.

Returns 0 on success, -1 on failure

#### 4.7.2.9 `int virConnectDomainEventRegisterAny ( virConnectPtr conn, virDomainPtr dom, int eventID, virConnectDomainEventGenericCallback cb, void * opaque, virFreeCallback freecb )`

`virConnectDomainEventRegisterAny`: : pointer to the connection : pointer to the domain : the event type to receive : callback to the function handling domain events : opaque data to pass on to the callback : optional function to deallocate opaque when not used anymore

Adds a callback to receive notifications of arbitrary domain events occurring on a domain.

If `dom` is `NULL`, then events will be monitored for any domain. If `dom` is non-`NULL`, then only the specific domain will be monitored

Most types of event have a callback providing a custom set of parameters for the event. When registering an event, it is thus necessary to use the [VIR\\_DOMAIN\\_EVENT\\_CALLBACK\(\)](#) macro to cast the supplied function pointer to match the signature of this method.

The `virDomainPtr` object handle passed into the callback upon delivery of an event is only valid for the duration of execution of the callback. If the callback wishes to keep the domain object after the callback returns, it shall take a reference to it, by calling `virDomainRef`. The reference can be released once the object is no longer required by calling `virDomainFree`.

The return value from this method is a positive integer identifier for the callback. To unregister a callback, this callback ID should be passed to the `virDomainEventUnregisterAny` method

Returns a callback identifier on success, -1 on failure



**4.7.2.10** `char* virConnectDomainXMLFromNative ( virConnectPtr conn, const char * nativeFormat, const char * nativeConfig, unsigned int flags )`

virConnectDomainXMLFromNative: : a connection object : configuration format importing from : the configuration data to import : extra flags; not used yet, so callers should always pass 0

Reads native configuration data describing a domain, and generates libvirt domain XML. The format of the native data is hypervisor dependant.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

**4.7.2.11** `char* virConnectDomainXMLToNative ( virConnectPtr conn, const char * nativeFormat, const char * domainXml, unsigned int flags )`

virConnectDomainXMLToNative: : a connection object : configuration format exporting to : the domain configuration to export : extra flags; not used yet, so callers should always pass 0

Reads a domain XML configuration document, and generates a native configuration file describing the domain. The format of the native data is hypervisor dependant.

Returns a 0 terminated UTF-8 encoded native config datafile, or NULL in case of error. the caller must free() the returned value.

**4.7.2.12** `char* virConnectFindStoragePoolSources ( virConnectPtr conn, const char * type, const char * srcSpec, unsigned int flags )`

virConnectFindStoragePoolSources: : pointer to hypervisor connection : type of storage pool sources to discover : XML document specifying discovery source : extra flags; not used yet, so callers should always pass 0

Talks to a storage backend and attempts to auto-discover the set of available storage pool sources. e.g. For iSCSI this would be a set of iSCSI targets. For NFS this would be a list of exported paths. The srcSpec (optional for some storage pool types, e.g. local ones) is an instance of the storage pool's source element specifying where to look for the pools.

srcSpec is not required for some types (e.g., those querying local storage resources only)

Returns an xml document consisting of a SourceList element containing a source document appropriate to the given pool type for each discovered source.

**4.7.2.13** `char* virConnectGetCapabilities ( virConnectPtr conn )`

virConnectGetCapabilities: : pointer to the hypervisor connection

Provides capabilities of the hypervisor / driver.

Returns NULL in case of error, or an XML string defining the capabilities. The client must free the returned string after use.

**4.7.2.14** `static int virConnectGetConfigFile ( virConfPtr * conf ) [static]`

**4.7.2.15** `static char* virConnectGetConfigFilePath ( void ) [static]`

**4.7.2.16** `static int virConnectGetDefaultURI ( virConfPtr conf, const char ** name ) [static]`

**4.7.2.17** `char* virConnectGetHostname ( virConnectPtr conn )`

virConnectGetHostname: : pointer to a hypervisor connection

This returns the system hostname on which the hypervisor is running (the result of the `gethostname` system call). If we are connected to a remote system, then this returns the hostname of the remote system.

Returns the hostname which must be freed by the caller, or NULL if there was an error.

#### 4.7.2.18 `int virConnectGetLibVersion ( virConnectPtr conn, unsigned long * libVer )`

`virConnectGetLibVersion`: : pointer to the hypervisor connection : returns the libvirt library version used on the connection (OUT)

Provides , which is the version of libvirt used by the daemon running on the host

Returns -1 in case of failure, 0 otherwise, and values for have the format `major * 1,000,000 + minor * 1,000 + release`.

#### 4.7.2.19 `int virConnectGetMaxVcpus ( virConnectPtr conn, const char * type )`

`virConnectGetMaxVcpus`: : pointer to the hypervisor connection : value of the 'type' attribute in the <domain> element

Provides the maximum number of virtual CPUs supported for a guest VM of a specific type. The 'type' parameter here corresponds to the 'type' attribute in the <domain> element of the XML.

Returns the maximum of virtual CPU or -1 in case of error.

#### 4.7.2.20 `char* virConnectGetSysinfo ( virConnectPtr conn, unsigned int flags )`

`virConnectGetSysinfo`: : pointer to a hypervisor connection : extra flags; not used yet, so callers should always pass 0

This returns the XML description of the sysinfo details for the host on which the hypervisor is running, in the same format as the <sysinfo> element of a domain XML. This information is generally available only for hypervisors running with root privileges.

Returns the XML string which must be freed by the caller, or NULL if there was an error.

#### 4.7.2.21 `const char* virConnectGetType ( virConnectPtr conn )`

`virConnectGetType`: : pointer to the hypervisor connection

Get the name of the Hypervisor software used.

Returns NULL in case of error, a static zero terminated string otherwise.

See also: <http://www.redhat.com/archives/libvir-list/2007-February/msg00096.-html>

#### 4.7.2.22 `char* virConnectGetURI ( virConnectPtr conn )`

`virConnectGetURI`: : pointer to a hypervisor connection

This returns the URI (name) of the hypervisor connection. Normally this is the same as or similar to the string passed to the `virConnectOpen/virConnectOpenReadOnly` call, but the driver may make the URI canonical. If name == NULL was passed to `virConnectOpen`, then the driver will return a non-NULL URI which can be used to connect to the same hypervisor later.

Returns the URI string which must be freed by the caller, or NULL if there was an error.

**4.7.2.23 int virConnectGetVersion ( virConnectPtr conn, unsigned long \* hvVer )**

virConnectGetVersion: : pointer to the hypervisor connection : return value for the version of the running hypervisor (OUT)

Get the version level of the Hypervisor running. This may work only with hypervisor call, i.e. with privileged access to the hypervisor, not with a Read-Only connection.

Returns -1 in case of error, 0 otherwise. if the version can't be extracted by lack of capacities returns 0 and is 0, otherwise value is major \* 1,000,000 + minor \* 1,000 + release

**4.7.2.24 int virConnectIsAlive ( virConnectPtr conn )**

virConnectIsAlive: : pointer to the connection object

Determine if the connection to the hypervisor is still alive

A connection will be classed as alive if it is either local, or running over a channel (TCP or UNIX socket) which is not closed.

Returns 1 if alive, 0 if dead, -1 on error

**4.7.2.25 int virConnectIsEncrypted ( virConnectPtr conn )**

virConnectIsEncrypted: : pointer to the connection object

Determine if the connection to the hypervisor is encrypted

Returns 1 if encrypted, 0 if not encrypted, -1 on error

**4.7.2.26 int virConnectIsSecure ( virConnectPtr conn )**

virConnectIsSecure: : pointer to the connection object

Determine if the connection to the hypervisor is secure

A connection will be classed as secure if it is either encrypted, or running over a channel which is not exposed to eavesdropping (eg a UNIX domain socket, or pipe)

Returns 1 if secure, 0 if secure, -1 on error

**4.7.2.27 int virConnectListAllDomains ( virConnectPtr conn, virDomainPtr \*\* domains, unsigned int flags )**

virConnectListAllDomains: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing domain objects or NULL if the list is not required (just returns number of guests). : bitwise-OR of virConnectListAllDomainsFlags

Collect a possibly-filtered list of all domains, and return an allocated array of information for each. This API solves the race inherent in [virConnectListDomains\(\)](#) and [virConnectListDefinedDomains\(\)](#).

Normally, all domains are returned; however, can be used to filter the results for a smaller list of targeted domains. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a domain, and where all bits within a group describe all possible domains. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction (for example, not all hypervisors can tell whether domains have snapshots). For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination (such as an inactive transient domain), in that case a hypervisor may return either 0 or an error.

The first group of is VIR\_CONNECT\_LIST\_DOMAINS\_ACTIVE (online domains) and VIR\_CONNECT\_LIST\_DOMAINS\_INACTIVE (offline domains).

The next group of is `VIR_CONNECT_LIST_DOMAINS_PERSISTENT` (defined domains) and `VIR_CONNECT_LIST_DOMAINS_TRANSIENT` (running but not defined).

The next group of covers various domain states: `VIR_CONNECT_LIST_DOMAINS_RUNNING`, `VIR_CONNECT_LIST_DOMAINS_PAUSED`, `VIR_CONNECT_LIST_DOMAINS_SHUTOFF`, and a catch-all for all other states (such as crashed, this catch-all covers the possibility of adding new states).

The remaining groups cover boolean attributes commonly asked about domains; they include `VIR_CONNECT_LIST_DOMAINS_MANAGEDSAVE` and `VIR_CONNECT_LIST_DOMAINS_NO_MANAGEDSAVE`, for filtering based on whether a managed save image exists; `VIR_CONNECT_LIST_DOMAINS_AUTOSTART` and `VIR_CONNECT_LIST_DOMAINS_NO_AUTOSTART`, for filtering based on autostart; `VIR_CONNECT_LIST_DOMAINS_HAS_SNAPSHOT` and `VIR_CONNECT_LIST_DOMAINS_NO_SNAPSHOT`, for filtering based on whether a domain has snapshots.

Returns the number of domains found or -1 and sets domains to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling `virDomainFree()` on each array element, then calling `free()` on .

Example of usage: `virDomainPtr *domains; virDomainPtr dom; int i; int ret; unsigned int flags = VIR_CONNECT_LIST_RUNNING | VIR_CONNECT_LIST_PERSISTENT;`

```
ret = virConnectListAllDomains(conn, &domains, flags); if (ret < 0) error();
```

```
for (i = 0; i < ret; i++) { do_something_with_domain(domains[i]);
```

```
//here or in a separate loop if needed virDomainFree(domains[i]); }
```

```
free(domains);
```

#### 4.7.2.28 `int virConnectListAllInterfaces ( virConnectPtr conn, virInterfacePtr ** ifaces, unsigned int flags )`

`virConnectListAllInterfaces`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the interface objects or NULL if the list is not required (just returns number of interfaces). : bitwise-OR of `virConnectListAllInterfacesFlags`.

Collect the list of interfaces, and allocate an array to store those objects. This API solves the race inherent between `virConnectListInterfaces` and `virConnectListDefinedInterfaces`.

Normally, all interfaces are returned; however, can be used to filter the results for a smaller list of targeted interfaces. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a interface, and where all bits within a group describe all possible interfaces.

The only group of is `VIR_CONNECT_LIST_INTERFACES_ACTIVE` (up) and `VIR_CONNECT_LIST_INTERFACES_INACTIVE` (down) to filter the interfaces by state.

Returns the number of interfaces found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling `virStorageInterfaceFree()` on each array element, then calling `free()` on .

#### 4.7.2.29 `int virConnectListAllNetworks ( virConnectPtr conn, virNetworkPtr ** nets, unsigned int flags )`

`virConnectListAllNetworks`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the network objects or NULL if the list is not required (just returns number of networks). : bitwise-OR of `virConnectListAllNetworksFlags`.

Collect the list of networks, and allocate an array to store those objects. This API solves the race inherent between `virConnectListNetworks` and `virConnectListDefinedNetworks`.

Normally, all networks are returned; however, can be used to filter the results for a smaller list of targeted networks. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a network, and where all bits within a group describe all possible networks.

The first group of is `VIR_CONNECT_LIST_NETWORKS_ACTIVE` (up) and `VIR_CONNECT_LIST_NETWORKS_INACTIVE` (down) to filter the networks by state.

The second group of is `VIR_CONNECT_LIST_NETWORKS_PERSISTENT` (defined) and `VIR_CONNECT_LIST_NETWORKS_TRANSIENT` (running but not defined), to filter the networks by whether they have persistent config or not.

The third group of is `VIR_CONNECT_LIST_NETWORKS_AUTOSTART` and `VIR_CONNECT_LIST_NETWORKS_NO_AUTOSTART`, to filter the networks by whether they are marked as autostart or not.

Returns the number of networks found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virNetworkFree\(\)](#) on each array element, then calling `free()` on .

#### 4.7.2.30 `int virConnectListAllNodeDevices ( virConnectPtr conn, virNodeDevicePtr ** devices, unsigned int flags )`

`virConnectListAllNodeDevices`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the node device objects or NULL if the list is not required (just returns number of node devices). : bitwise-OR of `virConnectListAllNodeDevices`.

Collect the list of node devices, and allocate an array to store those objects.

Normally, all node devices are returned; however, can be used to filter the results for a smaller list of targeted node devices. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a node device, and where all bits within a group describe all possible node devices.

Only one group of the is provided to filter the node devices by capability type, flags include: `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SYSTEM` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_PCI_DEV` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_USB_DEV` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_USB_INTERFACE` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_NET` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI_HOST` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI_TARGET` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_SCSI` `VIR_CONNECT_LIST_NODE_DEVICES_CAP_STORAGE`

Returns the number of node devices found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virNodeDeviceFree\(\)](#) on each array element, then calling `free()` on .

#### 4.7.2.31 `int virConnectListAllNWFilters ( virConnectPtr conn, virNWFilterPtr ** filters, unsigned int flags )`

`virConnectListAllNWFilters`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the network filter objects or NULL if the list is not required (just returns number of network filters). : extra flags; not used yet, so callers should always pass 0

Collect the list of network filters, and allocate an array to store those objects.

Returns the number of network filters found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virNWFilterFree\(\)](#) on each array element, then calling `free()` on .

#### 4.7.2.32 `int virConnectListAllSecrets ( virConnectPtr conn, virSecretPtr ** secrets, unsigned int flags )`

`virConnectListAllSecrets`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing the secret objects or NULL if the list is not required (just returns the number of secrets). : extra flags; not used yet, so callers should always pass 0

Collect the list of secrets, and allocate an array to store those objects.

Normally, all secrets are returned; however, can be used to filter the results for a smaller list of targeted secrets. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a secret, and where all bits within a group describe all possible secrets.

The first group of is used to filter secrets by its storage location. Flag `VIR_CONNECT_LIST_SECRETS_EPHEMERAL` selects secrets that are kept only in memory. Flag `VIR_CONNECT_LIST_SECRETS_NO_EPHEMERAL` selects secrets that are kept in persistent storage.

The second group of is used to filter secrets by privacy. Flag `VIR_CONNECT_LIST_SECRETS_PRIVATE` selects secrets that are never revealed to any caller of libvirt nor to any other node. Flag `VIR_CONNECT_LIST_SECRETS_NO_PRIVATE` selects non-private secrets.

Returns the number of secrets found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virSecretFree\(\)](#) on each array element, then calling `free()` on .

#### 4.7.2.33 `int virConnectListAllStoragePools ( virConnectPtr conn, virStoragePoolPtr ** pools, unsigned int flags )`

`virConnectListAllStoragePools`: : Pointer to the hypervisor connection. : Pointer to a variable to store the array containing storage pool objects or NULL if the list is not required (just returns number of pools). : bitwise-OR of `virConnectListAllStoragePoolsFlags`.

Collect the list of storage pools, and allocate an array to store those objects. This API solves the race inherent between `virConnectListStoragePools` and `virConnectListDefinedStoragePools`.

Normally, all storage pools are returned; however, can be used to filter the results for a smaller list of targeted pools. The valid flags are divided into groups, where each group contains bits that describe mutually exclusive attributes of a pool, and where all bits within a group describe all possible pools.

The first group of is `VIR_CONNECT_LIST_STORAGE_POOLS_ACTIVE` (online) and `VIR_CONNECT_LIST_STORAGE_POOLS_INACTIVE` (offline) to filter the pools by state.

The second group of is `VIR_CONNECT_LIST_STORAGE_POOLS_PERSISTENT` (defined) and `VIR_CONNECT_LIST_STORAGE_POOLS_TRANSIENT` (running but not defined), to filter the pools by whether they have persistent config or not.

The third group of is `VIR_CONNECT_LIST_STORAGE_POOLS_AUTOSTART` and `VIR_CONNECT_LIST_STORAGE_POOLS_NO_AUTOSTART`, to filter the pools by whether they are marked as autostart or not.

The last group of is provided to filter the pools by the types, the flags include: `VIR_CONNECT_LIST_STORAGE_POOLS_DIR` `VIR_CONNECT_LIST_STORAGE_POOLS_FS` `VIR_CONNECT_LIST_STORAGE_POOLS_NETFS` `VIR_CONNECT_LIST_STORAGE_POOLS_LOGICAL` `VIR_CONNECT_LIST_STORAGE_POOLS_DISK` `VIR_CONNECT_LIST_STORAGE_POOLS_ISCSI` `VIR_CONNECT_LIST_STORAGE_POOLS_SCSI` `VIR_CONNECT_LIST_STORAGE_POOLS_MPATH` `VIR_CONNECT_LIST_STORAGE_POOLS_RBD` `VIR_CONNECT_LIST_STORAGE_POOLS_SHEEPDOG`

Returns the number of storage pools found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virStoragePoolFree\(\)](#) on each array element, then calling `free()` on .

#### 4.7.2.34 `int virConnectListDefinedDomains ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListDefinedDomains`: : pointer to the hypervisor connection : pointer to an array to store the names : size of the array

list the defined but inactive domains, stores the pointers to the names in

For active domains, see [virConnectListDomains\(\)](#). For more control over the results, see [virConnectListAllDomains\(\)](#).

Returns the number of names provided in the array or -1 in case of error. Note that this command is inherently racy; a domain can be defined between a call to [virConnectNumOfDefinedDomains\(\)](#) and this call; you are only guaranteed that all currently defined domains were listed if the return is less than . The client must call `free()` on each returned name.

#### 4.7.2.35 `int virConnectListDefinedInterfaces ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListDefinedInterfaces`: : pointer to the hypervisor connection : array to collect the list of names of interfaces : size of

Collect the list of defined (inactive) physical host interfaces, and store their names in .

For more control over the results, see [virConnectListAllInterfaces\(\)](#).

Returns the number of names provided in the array or -1 in case of error. Note that this command is inherently racy; a interface can be defined between a call to [virConnectNumOfDefinedInterfaces\(\)](#) and this call; you are only guaranteed that all currently defined interfaces were listed if the return is less than . The client must call `free()` on each returned name.

#### 4.7.2.36 `int virConnectListDefinedNetworks ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListDefinedNetworks`: : pointer to the hypervisor connection : pointer to an array to store the names : size of the array

list the inactive networks, stores the pointers to the names in

For more control over the results, see [virConnectListAllNetworks\(\)](#).

Returns the number of names provided in the array or -1 in case of error. Note that this command is inherently racy; a network can be defined between a call to [virConnectNumOfDefinedNetworks\(\)](#) and this call; you are only guaranteed that all currently defined networks were listed if the return is less than . The client must call `free()` on each returned name.

#### 4.7.2.37 `int virConnectListDefinedStoragePools ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListDefinedStoragePools`: : pointer to hypervisor connection : array of char \* to fill with pool names (allocated by caller) : size of the names array

Provides the list of names of inactive storage pools up to maxnames. If there are more than maxnames, the remaining names will be silently ignored.

For more control over the results, see [virConnectListAllStoragePools\(\)](#).

Returns the number of names provided in the array or -1 in case of error. Note that this command is inherently racy; a pool can be defined between a call to [virConnectNumOfDefinedStoragePools\(\)](#) and this call; you are only guaranteed that all currently defined pools were listed if the return is less than . The client must call `free()` on each returned name.

#### 4.7.2.38 `int virConnectListDomains ( virConnectPtr conn, int * ids, int maxids )`

`virConnectListDomains`: : pointer to the hypervisor connection : array to collect the list of IDs of active domains : size of

Collect the list of active domains, and store their IDs in array

For inactive domains, see [virConnectListDefinedDomains\(\)](#). For more control over the results, see [virConnectListAllDomains\(\)](#).

Returns the number of domains found or -1 in case of error. Note that this command is inherently racy; a domain can be started between a call to [virConnectNumOfDomains\(\)](#) and this call; you are only guaranteed that all currently active domains were listed if the return is less than .

#### 4.7.2.39 `int virConnectListInterfaces ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListInterfaces`: : pointer to the hypervisor connection : array to collect the list of names of interfaces : size of

Collect the list of active physical host interfaces, and store their names in

For more control over the results, see [virConnectListAllInterfaces\(\)](#).

Returns the number of interfaces found or -1 in case of error. Note that this command is inherently racy; a interface

can be started between a call to [virConnectNumOfInterfaces\(\)](#) and this call; you are only guaranteed that all currently active interfaces were listed if the return is less than .

#### 4.7.2.40 `int virConnectListNetworks ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListNetworks`: : pointer to the hypervisor connection : array to collect the list of names of active networks  
: size of

Collect the list of active networks, and store their names in

For more control over the results, see [virConnectListAllNetworks\(\)](#).

Returns the number of networks found or -1 in case of error. Note that this command is inherently racy; a network can be started between a call to [virConnectNumOfNetworks\(\)](#) and this call; you are only guaranteed that all currently active networks were listed if the return is less than .

#### 4.7.2.41 `int virConnectListNWFilters ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListNWFilters`: : pointer to the hypervisor connection : array to collect the list of names of network filters  
: size of

Collect the list of network filters, and store their names in

Returns the number of network filters found or -1 in case of error

#### 4.7.2.42 `int virConnectListSecrets ( virConnectPtr conn, char ** uuids, int maxuuids )`

`virConnectListSecrets`: : `virConnect` connection : Pointer to an array to store the UUIDs : size of the array.

List UUIDs of defined secrets, store pointers to names in `uuids`.

Returns the number of UUIDs provided in the array, or -1 on failure.

#### 4.7.2.43 `int virConnectListStoragePools ( virConnectPtr conn, char **const names, int maxnames )`

`virConnectListStoragePools`: : pointer to hypervisor connection : array of `char *` to fill with pool names (allocated by caller) : size of the names array

Provides the list of names of active storage pools up to `maxnames`. If there are more than `maxnames`, the remaining names will be silently ignored.

For more control over the results, see [virConnectListAllStoragePools\(\)](#).

Returns the number of pools found or -1 in case of error. Note that this command is inherently racy; a pool can be started between a call to [virConnectNumOfStoragePools\(\)](#) and this call; you are only guaranteed that all currently active pools were listed if the return is less than .

#### 4.7.2.44 `int virConnectNumOfDefinedDomains ( virConnectPtr conn )`

`virConnectNumOfDefinedDomains`: : pointer to the hypervisor connection

Provides the number of defined but inactive domains.

Returns the number of domain found or -1 in case of error

#### 4.7.2.45 `int virConnectNumOfDefinedInterfaces ( virConnectPtr conn )`

`virConnectNumOfDefinedInterfaces`: : pointer to the hypervisor connection

Provides the number of defined (inactive) interfaces on the physical host.



Returns the number of defined interface found or -1 in case of error

#### 4.7.2.46 int virConnectNumOfDefinedNetworks ( virConnectPtr conn )

virConnectNumOfDefinedNetworks: : pointer to the hypervisor connection

Provides the number of inactive networks.

Returns the number of networks found or -1 in case of error

#### 4.7.2.47 int virConnectNumOfDefinedStoragePools ( virConnectPtr conn )

virConnectNumOfDefinedStoragePools: : pointer to hypervisor connection

Provides the number of inactive storage pools

Returns the number of pools found, or -1 on error

#### 4.7.2.48 int virConnectNumOfDomains ( virConnectPtr conn )

virConnectNumOfDomains: : pointer to the hypervisor connection

Provides the number of active domains.

Returns the number of domain found or -1 in case of error

#### 4.7.2.49 int virConnectNumOfInterfaces ( virConnectPtr conn )

virConnectNumOfInterfaces: : pointer to the hypervisor connection

Provides the number of active interfaces on the physical host.

Returns the number of active interfaces found or -1 in case of error

#### 4.7.2.50 int virConnectNumOfNetworks ( virConnectPtr conn )

virConnectNumOfNetworks: : pointer to the hypervisor connection

Provides the number of active networks.

Returns the number of network found or -1 in case of error

#### 4.7.2.51 int virConnectNumOfNWFilters ( virConnectPtr conn )

virConnectNumOfNWFilters: : pointer to the hypervisor connection

Provides the number of nwfilters.

Returns the number of nwfilters found or -1 in case of error

#### 4.7.2.52 int virConnectNumOfSecrets ( virConnectPtr conn )

virConnectNumOfSecrets: : virConnect connection

Fetch number of currently defined secrets.

Returns the number currently defined secrets.

**4.7.2.53** `int virConnectNumOfStoragePools ( virConnectPtr conn )`

`virConnectNumOfStoragePools`: : pointer to hypervisor connection

Provides the number of active storage pools

Returns the number of pools found, or -1 on error

**4.7.2.54** `virConnectPtr virConnectOpen ( const char * name )`

**4.7.2.55** `virConnectPtr virConnectOpenAuth ( const char * name, virConnectAuthPtr auth, unsigned int flags )`

**4.7.2.56** `static int virConnectOpenFindURIAliasMatch ( virConfValuePtr value, const char * alias, char ** uri )` [static]

**4.7.2.57** `virConnectPtr virConnectOpenReadOnly ( const char * name )`

**4.7.2.58** `static int virConnectOpenResolveURIAlias ( virConfPtr conf, const char * alias, char ** uri )` [static]

**4.7.2.59** `int virConnectRef ( virConnectPtr conn )`

`virConnectRef`: : the connection to hold a reference on

Increment the reference count on the connection. For each additional call to this method, there shall be a corresponding call to `virConnectClose` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a connection would increment the reference count.

Returns 0 in case of success, -1 in case of failure

**4.7.2.60** `int virConnectRegisterCloseCallback ( virConnectPtr conn, virConnectCloseFunc cb, void * opaque, virFreeCallback freecb )`

`virConnectRegisterCloseCallback`: : pointer to connection object : callback to invoke upon close : user data to pass to : callback to free

Registers a callback to be invoked when the connection is closed. This callback is invoked when there is any condition that causes the socket connection to the hypervisor to be closed.

This function is only applicable to hypervisor drivers which maintain a persistent open connection. Drivers which open a new connection for every operation will not invoke this.

The must not invoke any other libvirt public APIs, since it is not called from a re-entrant safe context.

Returns 0 on success, -1 on error

**4.7.2.61** `int virConnectSetKeepAlive ( virConnectPtr conn, int interval, unsigned int count )`

`virConnectSetKeepAlive`: : pointer to a hypervisor connection : number of seconds of inactivity before a keepalive message is sent : number of messages that can be sent in a row

Start sending keepalive messages after interval second of inactivity and consider the connection to be broken when no response is received after count keepalive messages sent in a row. In other words, sending count + 1 keepalive message results in closing the connection. When interval is  $\leq 0$ , no keepalive messages will be sent. When count is 0, the connection will be automatically closed after interval seconds of inactivity without sending any keepalive messages.

Note: client has to implement and run event loop to be able to use keepalive messages. Failure to do so may result in connections being closed unexpectedly.

Note: This API function controls only keepalive messages sent by the client. If the server is configured to use keepalive you still need to run the event loop to respond to them, even if you disable keepalives by this function.

Returns -1 on error, 0 on success, 1 when remote party doesn't support keepalive messages.

#### 4.7.2.62 `int virConnectUnregisterCloseCallback ( virConnectPtr conn, virConnectCloseFunc cb )`

`virConnectUnregisterCloseCallback`: : pointer to connection object : pointer to the current registered callback

Unregisters the callback previously set with the `virConnectRegisterCloseCallback` method. The callback will no longer receive notifications when the connection closes. If a `virFreeCallback` was provided at time of registration, it will be invoked

Returns 0 on success, -1 on error

#### 4.7.2.63 `int virDomainAbortJob ( virDomainPtr domain )`

`virDomainAbortJob`: : a domain object

Requests that the current background job be aborted at the soonest opportunity. This will block until the job has either completed, or aborted.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.64 `int virDomainAttachDevice ( virDomainPtr domain, const char * xml )`

`virDomainAttachDevice`: : pointer to domain object : pointer to XML description of one device

Create a virtual device attachment to backend. This function, having hotplug semantics, is only allowed on an active domain.

For compatibility, this method can also be used to change the media in an existing CDROM/Floppy device, however, applications are recommended to use the `virDomainUpdateDeviceFlag` method instead.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.65 `int virDomainAttachDeviceFlags ( virDomainPtr domain, const char * xml, unsigned int flags )`

`virDomainAttachDeviceFlags`: : pointer to domain object : pointer to XML description of one device : bitwise-OR of `virDomainDeviceModifyFlags`

Attach a virtual device to a domain, using the flags parameter to control how the device is attached. `VIR_DOMAIN_AFFECT_CURRENT` specifies that the device allocation is made based on current domain state. `VIR_DOMAIN_AFFECT_LIVE` specifies that the device shall be allocated to the active domain instance only and is not added to the persisted domain configuration. `VIR_DOMAIN_AFFECT_CONFIG` specifies that the device shall be allocated to the persisted domain configuration only. Note that the target hypervisor must return an error if unable to satisfy flags. E.g. the hypervisor driver will return failure if LIVE is specified but it only supports modifying the persisted device allocation.

For compatibility, this method can also be used to change the media in an existing CDROM/Floppy device, however, applications are recommended to use the `virDomainUpdateDeviceFlag` method instead.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.66 `int virDomainBlockCommit ( virDomainPtr dom, const char * disk, const char * base, const char * top, unsigned long bandwidth, unsigned int flags )`

`virDomainBlockCommit`: : pointer to domain object : path to the block device, or device shorthand : path to backing file to merge into, or NULL for default : path to file within backing chain that contains data to be merged, or NULL to

merge all possible data : (optional) specify commit bandwidth limit in MiB/s : bitwise-OR of `virDomainBlockCommitFlags`

Commit changes that were made to temporary top-level files within a disk image backing file chain into a lower-level base file. In other words, take all the difference between and , and update to contain that difference; after the commit, any portion of the chain that previously depended on will now depend on , and all files after up to and including will now be invalidated. A typical use of this command is to reduce the length of a backing file chain after taking an external disk snapshot. To move data in the opposite direction, see [virDomainBlockPull\(\)](#).

This command starts a long-running commit block job, whose status may be tracked by [virDomainBlockJobInfo\(\)](#) with a job type of `VIR_DOMAIN_BLOCK_JOB_TYPE_COMMIT`, and the operation can be aborted with [virDomainBlockJobAbort\(\)](#). When finished, an asynchronous event is raised to indicate the final status, and the job no longer exists. If the job is aborted, it is up to the hypervisor whether starting a new job will resume from the same point, or start over.

Be aware that this command may invalidate files even if it is aborted; the user is cautioned against relying on the contents of invalidated intermediate files such as without manually rebasing those files to use a backing file of a read-only copy of prior to the point where the commit operation was started (although such a rebase cannot be safely done until the commit has successfully completed). However, the domain itself will not have any issues; the active layer remains valid throughout the entire commit operation. As a convenience, if contains `VIR_DOMAIN_BLOCK_COMMIT_DELETE`, this command will unlink all files that were invalidated, after the commit successfully completes.

By default, if is `NULL`, the commit target will be the bottom of the backing chain; if contains `VIR_DOMAIN_BLOCK_COMMIT_SHALLOW`, then the immediate backing file of will be used instead. If is `NULL`, the active image at the top of the chain will be used. Some hypervisors place restrictions on how much can be committed, and might fail if is not the immediate backing file of , or if is the active layer in use by a running domain, or if is not the top-most file; restrictions may differ for online vs. offline domains.

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

The maximum bandwidth (in MiB/s) that will be used to do the commit can be specified with the bandwidth parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0; in this case, it might still be possible for a later call to [virDomainBlockJobSetSpeed\(\)](#) to succeed. The actual speed can be determined with [virDomainGetBlockJobInfo\(\)](#).

Returns 0 if the operation has started, -1 on failure.

#### 4.7.2.67 `int virDomainBlockJobAbort ( virDomainPtr dom, const char * disk, unsigned int flags )`

`virDomainBlockJobAbort` : pointer to domain object : path to the block device, or device shorthand : bitwise-OR of `virDomainBlockJobAbortFlags`

Cancel the active block job on the given disk.

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `"/path/to/image"`), or (since 0.9.5) the device target shorthand (the `<target dev="...">` sub-element, such as `"xvda"`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

If the current block job for is `VIR_DOMAIN_BLOCK_JOB_TYPE_PULL`, then by default, this function performs a synchronous operation and the caller may assume that the operation has completed when 0 is returned. However, BlockJob operations may take a long time to cancel, and during this time further domain interactions may be unresponsive. To avoid this problem, pass `VIR_DOMAIN_BLOCK_JOB_ABORT_ASYNC` in the argument to enable asynchronous behavior, returning as soon as possible. When the job has been canceled, a BlockJob event will be emitted, with status `VIR_DOMAIN_BLOCK_JOB_CANCELED` (even if the `ABORT_ASYNC` flag was not used); it is also possible to poll [virDomainBlockJobInfo\(\)](#) to see if the job cancellation is still pending. This type of job can be restarted to pick up from where it left off.

If the current block job for is `VIR_DOMAIN_BLOCK_JOB_TYPE_COPY`, then the default is to abort the mirroring and revert to the source disk; adding of `VIR_DOMAIN_BLOCK_JOB_ABORT_PIVOT` causes this call to fail with

VIR\_ERR\_BLOCK\_COPY\_ACTIVE if the copy is not fully populated, otherwise it will swap the disk over to the copy to end the mirroring. An event will be issued when the job is ended, and it is possible to use VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_ASYNC to control whether this command waits for the completion of the job. Restarting this job requires starting over from the beginning of the first phase.

Returns -1 in case of failure, 0 when successful.

**4.7.2.68** `int virDomainBlockJobSetSpeed ( virDomainPtr dom, const char * disk, unsigned long bandwidth, unsigned int flags )`

virDomainBlockJobSetSpeed: : pointer to domain object : path to the block device, or device shorthand : specify bandwidth limit in MiB/s : extra flags; not used yet, so callers should always pass 0

Set the maximum allowable bandwidth that a block job may consume. If bandwidth is 0, the limit will revert to the hypervisor default.

The parameter is either an unambiguous source name of the block device (the <source file="..."> sub-element, such as "/path/to/image"), or (since 0.9.5) the device target shorthand (the <target dev="..."> sub-element, such as "xvda"). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within //domain/devices/disk.

Returns -1 in case of failure, 0 when successful.

**4.7.2.69** `int virDomainBlockPeek ( virDomainPtr dom, const char * disk, unsigned long long offset, size_t size, void * buffer, unsigned int flags )`

virDomainBlockPeek: : pointer to the domain object : path to the block device, or device shorthand : offset within block device : size to read : return buffer (must be at least size bytes) : extra flags; not used yet, so callers should always pass 0

This function allows you to read the contents of a domain's disk device.

Typical uses for this are to determine if the domain has written a Master Boot Record (indicating that the domain has completed installation), or to try to work out the state of the domain's filesystems.

(Note that in the local case you might try to open the block device or file directly, but that won't work in the remote case, nor if you don't have sufficient permission. Hence the need for this call).

The parameter is either an unambiguous source name of the block device (the <source file="..."> sub-element, such as "/path/to/image"), or (since 0.9.5) the device target shorthand (the <target dev="..."> sub-element, such as "xvda"). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within //domain/devices/disk.

'offset' and 'size' represent an area which must lie entirely within the device or file. 'size' may be 0 to test if the call would succeed.

'buffer' is the return buffer and must be at least 'size' bytes.

NB. The remote driver imposes a 64K byte limit on 'size'. For your program to be able to work reliably over a remote connection you should split large requests to <= 65536 bytes. However, with 0.9.13 this RPC limit has been raised to 1M byte.

Returns: 0 in case of success or -1 in case of failure.

**4.7.2.70** `int virDomainBlockPull ( virDomainPtr dom, const char * disk, unsigned long bandwidth, unsigned int flags )`

virDomainBlockPull: : pointer to domain object : path to the block device, or device shorthand : (optional) specify copy bandwidth limit in MiB/s : extra flags; not used yet, so callers should always pass 0

Populate a disk image with data from its backing image. Once all data from its backing image has been pulled, the disk no longer depends on a backing image. This function pulls data for the entire device in the background. Progress of the operation can be checked with [virDomainGetBlockJobInfo\(\)](#) and the operation can be aborted with

[virDomainBlockJobAbort\(\)](#). When finished, an asynchronous event is raised to indicate the final status. To move data in the opposite direction, see [virDomainBlockCommit\(\)](#).

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `/path/to/image`), or (since 0.9.5) the device target shorthand (the `<target dev="...">` sub-element, such as `xvda`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

The maximum bandwidth (in MiB/s) that will be used to do the copy can be specified with the `bandwidth` parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0; in this case, it might still be possible for a later call to [virDomainBlockJobSetSpeed\(\)](#) to succeed. The actual speed can be determined with [virDomainGetBlockJobInfo\(\)](#).

This is shorthand for [virDomainBlockRebase\(\)](#) with a NULL base.

Returns 0 if the operation has started, -1 on failure.

**4.7.2.71** `int virDomainBlockRebase ( virDomainPtr dom, const char * disk, const char * base, unsigned long bandwidth, unsigned int flags )`

`virDomainBlockRebase`: : pointer to domain object : path to the block device, or device shorthand : path to backing file to keep, or NULL for no backing file : (optional) specify copy bandwidth limit in MiB/s : bitwise-OR of `virDomainBlockRebaseFlags`

Populate a disk image with data from its backing image chain, and setting the backing image to , or alternatively copy an entire backing chain to a new file .

When `bandwidth` is 0, this starts a pull, where `base` must be the absolute path of one of the backing images further up the chain, or NULL to convert the disk image so that it has no backing image. Once all data from its backing image chain has been pulled, the disk no longer depends on those intermediate backing images. This function pulls data for the entire device in the background. Progress of the operation can be checked with [virDomainGetBlockJobInfo\(\)](#) with a job type of `VIR_DOMAIN_BLOCK_JOB_TYPE_PULL`, and the operation can be aborted with [virDomainBlockJobAbort\(\)](#). When finished, an asynchronous event is raised to indicate the final status, and the job no longer exists. If the job is aborted, a new one can be started later to resume from the same point.

When `base` includes `VIR_DOMAIN_BLOCK_REBASE_COPY`, this starts a copy, where `base` must be the name of a new file to copy the chain to. By default, the copy will pull the entire source chain into the destination file, but if also contains `VIR_DOMAIN_BLOCK_REBASE_SHALLOW`, then only the top of the source chain will be copied (the source and destination have a common backing file). By default, `base` will be created with the same file format as the source, but this can be altered by adding `VIR_DOMAIN_BLOCK_REBASE_COPY_RAW` to force the copy to be raw (does not make sense with the shallow flag unless the source is also raw), or by using `VIR_DOMAIN_BLOCK_REBASE_REUSE_EXT` to reuse an existing file with initial contents identical to the backing file of the source (this allows a management app to pre-create files with relative backing file names, rather than the default of absolute backing file names; as a security precaution, you should generally only use `reuse_ext` with the shallow flag and a non-raw destination file).

A copy job has two parts; in the first phase, the parameter affects how fast the source is pulled into the destination, and the job can only be canceled by reverting to the source file; progress in this phase can be tracked via the [virDomainBlockJobInfo\(\)](#) command, with a job type of `VIR_DOMAIN_BLOCK_JOB_TYPE_COPY`. The job transitions to the second phase when the job info states `cur == end`, and remains alive to mirror all further changes to both source and destination. The user must call [virDomainBlockJobAbort\(\)](#) to end the mirroring while choosing whether to revert to source or pivot to the destination. An event is issued when the job ends, and in the future, an event may be added when the job transitions from pulling to mirroring. If the job is aborted, a new job will have to start over from the beginning of the first phase.

Some hypervisors will restrict certain actions, such as [virDomainSave\(\)](#) or [virDomainDetachDevice\(\)](#), while a copy job is active; they may also restrict a copy job to transient domains.

The parameter is either an unambiguous source name of the block device (the `<source file="...">` sub-element, such as `/path/to/image`), or the device target shorthand (the `<target dev="...">` sub-element, such as `xvda`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

The maximum bandwidth (in MiB/s) that will be used to do the copy can be specified with the `bandwidth` parameter.

If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0; in this case, it might still be possible for a later call to [virDomainBlockJobSetSpeed\(\)](#) to succeed. The actual speed can be determined with [virDomainGetBlockJobInfo\(\)](#).

When is NULL and is 0, this is identical to [virDomainBlockPull\(\)](#).

Returns 0 if the operation has started, -1 on failure.

#### 4.7.2.72 `int virDomainBlockResize ( virDomainPtr dom, const char * disk, unsigned long long size, unsigned int flags )`

`virDomainBlockResize`: : pointer to the domain object : path to the block image, or shorthand : new size of the block image, see below for unit : bitwise-OR of `virDomainBlockResizeFlags`

Resize a block device of domain while the domain is running. If is 0, then is in kibibytes (blocks of 1024 bytes); since 0.9.11, if includes `VIR_DOMAIN_BLOCK_RESIZE_BYTES`, is in bytes instead. is taken directly as the new size. Depending on the file format, the hypervisor may round up to the next alignment boundary.

The parameter is either an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `/path/to/image`), or (since 0.9.5) the device target shorthand (the `<target dev=...">` sub-element, such as `xvda`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Note that this call may fail if the underlying virtualization hypervisor does not support it; this call requires privileged access to the hypervisor.

Returns: 0 in case of success or -1 in case of failure.

#### 4.7.2.73 `int virDomainBlockStats ( virDomainPtr dom, const char * disk, virDomainBlockStatsPtr stats, size_t size )`

`virDomainBlockStats`: : pointer to the domain object : path to the block device, or device shorthand : block device stats (returned) : size of stats structure

This function returns block device (disk) stats for block devices attached to the domain.

The parameter is either the device target shorthand (the `<target dev=...">` sub-element, such as `xvda`), or (since 0.9.8) an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `/path/to/image`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Domains may have more than one block device. To get stats for each you should make multiple calls to this function.

Individual fields within the stats structure may be returned as -1, which indicates that the hypervisor does not support that particular statistic.

Returns: 0 in case of success or -1 in case of failure.

#### 4.7.2.74 `int virDomainBlockStatsFlags ( virDomainPtr dom, const char * disk, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainBlockStatsFlags`: : pointer to domain object : path to the block device, or device shorthand : pointer to block stats parameter object (return value) : pointer to number of block stats; input and output : bitwise-OR of `virTypedParameterFlags`

This function is to get block stats parameters for block devices attached to the domain.

The parameter is either the device target shorthand (the `<target dev=...">` sub-element, such as `xvda`), or (since 0.9.8) an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `/path/to/image`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Domains may have more than one block device. To get stats for each you should make multiple calls to this function.

On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. (Note that block devices of different types might support different parameters, so it might be necessary to compute for each block device). The caller should then allocate array, i.e. (sizeof() \* ) bytes and call the API again. See [virDomainGetMemoryParameters\(\)](#) for more details.

Returns -1 in case of error, 0 in case of success.

#### 4.7.2.75 int virDomainCoreDump ( virDomainPtr domain, const char \* to, unsigned int flags )

virDomainCoreDump: : a domain object : path for the core file : bitwise-OR of virDomainCoreDumpFlags

This method will dump the core of a domain on a given file for analysis. Note that for remote Xen Daemon the file path will be interpreted in the remote host. Hypervisors may require the user to manually ensure proper permissions on the file named by .

If includes VIR\_DUMP\_CRASH, then leave the guest shut off with a crashed state after the dump completes. If includes VIR\_DUMP\_LIVE, then make the core dump while continuing to allow the guest to run; otherwise, the guest is suspended during the dump. VIR\_DUMP\_RESET flag forces reset of the quest after dump. The above three flags are mutually exclusive.

Additionally, if includes VIR\_DUMP\_BYPASS\_CACHE, then libvirt will attempt to bypass the file system cache while creating the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing saves to NFS.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.76 int virDomainCreate ( virDomainPtr domain )

virDomainCreate: : pointer to a defined domain

Launch a defined domain. If the call succeeds the domain moves from the defined to the running domains pools. The domain will be paused only if restoring from managed state created from a paused domain. For more control, see [virDomainCreateWithFlags\(\)](#).

Returns 0 in case of success, -1 in case of error

#### 4.7.2.77 virDomainPtr virDomainCreateLinux ( virConnectPtr conn, const char \* xmlDesc, unsigned int flags )

virDomainCreateLinux: : pointer to the hypervisor connection : string containing an XML description of the domain : extra flags; not used yet, so callers should always pass 0

Deprecated after 0.4.6. Renamed to [virDomainCreateXML\(\)](#) providing identical functionality. This existing name will left indefinitely for API compatibility.

Returns a new domain object or NULL in case of failure

#### 4.7.2.78 int virDomainCreateWithFlags ( virDomainPtr domain, unsigned int flags )

virDomainCreateWithFlags: : pointer to a defined domain : bitwise-OR of supported virDomainCreateFlags

Launch a defined domain. If the call succeeds the domain moves from the defined to the running domains pools.

If the VIR\_DOMAIN\_START\_PAUSED flag is set, or if the guest domain has a managed save image that requested paused state (see [virDomainManagedSave\(\)](#)) the guest domain will be started, but its CPUs will remain paused. The CPUs can later be manually started using [virDomainResume\(\)](#). In all other cases, the guest domain will be running.

If the VIR\_DOMAIN\_START\_AUTODESTROY flag is set, the guest domain will be automatically destroyed when the virConnectPtr object is finally released. This will also happen if the client application crashes / loses its connection to the libvirtd daemon. Any domains marked for auto destroy will block attempts at migration, save-to-file, or snapshots.



If the `VIR_DOMAIN_START_BYPASS_CACHE` flag is set, and there is a managed save file for this domain (created by [virDomainManagedSave\(\)](#)), then libvirt will attempt to bypass the file system cache while restoring the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing loads from NFS.

If the `VIR_DOMAIN_START_FORCE_BOOT` flag is set, then any managed save file for this domain is discarded, and the domain boots from scratch.

Returns 0 in case of success, -1 in case of error

#### 4.7.2.79 `virDomainPtr virDomainCreateXML ( virConnectPtr conn, const char * xmlDesc, unsigned int flags )`

`virDomainCreateXML`: : pointer to the hypervisor connection : string containing an XML description of the domain : bitwise-OR of supported `virDomainCreateFlags`

Launch a new guest domain, based on an XML description similar to the one returned by [virDomainGetXMLDesc\(\)](#). This function may require privileged access to the hypervisor. The domain is not persistent, so its definition will disappear when it is destroyed, or if the host is restarted (see [virDomainDefineXML\(\)](#) to define persistent domains).

If the `VIR_DOMAIN_START_PAUSED` flag is set, the guest domain will be started, but its CPUs will remain paused. The CPUs can later be manually started using `virDomainResume`.

If the `VIR_DOMAIN_START_AUTODESTROY` flag is set, the guest domain will be automatically destroyed when the `virConnectPtr` object is finally released. This will also happen if the client application crashes / loses its connection to the libvirtd daemon. Any domains marked for auto destroy will block attempts at migration, save-to-file, or snapshots.

Returns a new domain object or NULL in case of failure

#### 4.7.2.80 `virDomainPtr virDomainDefineXML ( virConnectPtr conn, const char * xml )`

`virDomainDefineXML`: : pointer to the hypervisor connection : the XML description for the domain, preferably in UTF-8

Define a domain, but does not start it. This definition is persistent, until explicitly undefined with [virDomainUndefine\(\)](#). A previous definition for this domain would be overridden if it already exists.

Some hypervisors may prevent this operation if there is a current block copy operation on a transient domain with the same id as the domain being defined; in that case, use [virDomainBlockJobAbort\(\)](#) to stop the block copy first.

Returns NULL in case of error, a pointer to the domain otherwise

#### 4.7.2.81 `int virDomainDestroy ( virDomainPtr domain )`

`virDomainDestroy`: : a domain object

Destroy the domain object. The running instance is shutdown if not down already and all resources used by it are given back to the hypervisor. This does not free the associated `virDomainPtr` object. This function may require privileged access.

`virDomainDestroy` first requests that a guest terminate (e.g. `SIGTERM`), then waits for it to comply. After a reasonable timeout, if the guest still exists, `virDomainDestroy` will forcefully terminate the guest (e.g. `SIGKILL`) if necessary (which may produce undesirable results, for example unflushed disk cache in the guest). To avoid this possibility, it's recommended to instead call `virDomainDestroyFlags`, sending the `VIR_DOMAIN_DESTROY_GRACEFUL` flag.

If the domain is transient and has any snapshot metadata (see [virDomainSnapshotNum\(\)](#)), then that metadata will automatically be deleted when the domain quits.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.82 `int virDomainDestroyFlags ( virDomainPtr domain, unsigned int flags )`

`virDomainDestroyFlags`: : a domain object : bitwise-OR of `virDomainDestroyFlagsValues`

Destroy the domain object. The running instance is shutdown if not down already and all resources used by it are given back to the hypervisor. This does not free the associated `virDomainPtr` object. This function may require privileged access.

Calling this function with no set (equal to zero) is equivalent to calling `virDomainDestroy`, and after a reasonable timeout will forcefully terminate the guest (e.g. SIGKILL) if necessary (which may produce undesirable results, for example unflushed disk cache in the guest). Including `VIR_DOMAIN_DESTROY_GRACEFUL` in the flags will prevent the forceful termination of the guest, and `virDomainDestroyFlags` will instead return an error if the guest doesn't terminate by the end of the timeout; at that time, the management application can decide if calling again without `VIR_DOMAIN_DESTROY_GRACEFUL` is appropriate.

Another alternative which may produce cleaner results for the guest's disks is to use `virDomainShutdown()` instead, but that depends on guest support (some hypervisor/guest combinations may ignore the shutdown request).

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.83 `int virDomainDetachDevice ( virDomainPtr domain, const char * xml )`

`virDomainDetachDevice`: : pointer to domain object : pointer to XML description of one device

Destroy a virtual device attachment to backend. This function, having hot-unplug semantics, is only allowed on an active domain.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.84 `int virDomainDetachDeviceFlags ( virDomainPtr domain, const char * xml, unsigned int flags )`

`virDomainDetachDeviceFlags`: : pointer to domain object : pointer to XML description of one device : bitwise-OR of `virDomainDeviceModifyFlags`

Detach a virtual device from a domain, using the flags parameter to control how the device is detached. `VIR_DOMAIN_AFFECT_CURRENT` specifies that the device allocation is removed based on current domain state. `VIR_DOMAIN_AFFECT_LIVE` specifies that the device shall be deallocated from the active domain instance only and is not from the persisted domain configuration. `VIR_DOMAIN_AFFECT_CONFIG` specifies that the device shall be deallocated from the persisted domain configuration only. Note that the target hypervisor must return an error if unable to satisfy flags. E.g. the hypervisor driver will return failure if `LIVE` is specified but it only supports removing the persisted device allocation.

Some hypervisors may prevent this operation if there is a current block copy operation on the device being detached; in that case, use `virDomainBlockJobAbort()` to stop the block copy first.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.85 `int virDomainFree ( virDomainPtr domain )`

`virDomainFree`: : a domain object

Free the domain object. The running instance is kept alive. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.86 `int virDomainGetAutostart ( virDomainPtr domain, int * autostart )`

`virDomainGetAutostart`: : a domain object : the value returned

Provides a boolean value indicating whether the domain configured to be automatically started when the host machine boots.

Returns -1 in case of error, 0 in case of success

**4.7.2.87** `int virDomainGetBlkioParameters ( virDomainPtr domain, virTypedParameterPtr params, int * nparams, unsigned int flags )`

virDomainGetBlkioParameters: : pointer to domain object : pointer to blkio parameter object (return value, allocated by the caller) : pointer to number of blkio parameters; input and output : bitwise-OR of virDomainModificationImpact and virTypedParameterFlags

Get all blkio parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. The caller should then allocate array, i.e. (sizeof() \* ) bytes and call the API again.

See [virDomainGetMemoryParameters\(\)](#) for an equivalent usage example.

This function may require privileged access to the hypervisor. This function expects the caller to allocate the .

Returns -1 in case of error, 0 in case of success.

**4.7.2.88** `int virDomainGetBlockInfo ( virDomainPtr domain, const char * disk, virDomainBlockInfoPtr info, unsigned int flags )`

virDomainGetBlockInfo: : a domain object : path to the block device, or device shorthand : pointer to a virDomainBlockInfo structure allocated by the user : extra flags; not used yet, so callers should always pass 0

Extract information about a domain's block device.

The parameter is either an unambiguous source name of the block device (the <source file="..."> sub-element, such as "/path/to/image"), or (since 0.9.5) the device target shorthand (the <target dev="..."> sub-element, such as "xvda"). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within //domain/devices/disk.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.89** `int virDomainGetBlockioTune ( virDomainPtr dom, const char * disk, virTypedParameterPtr params, int * nparams, unsigned int flags )`

virDomainGetBlockioTune: : pointer to domain object : path to the block device, or device shorthand : Pointer to blkio parameter object (return value, allocated by the caller) : Pointer to number of blkio parameters : bitwise-OR of virDomainModificationImpact and virTypedParameterFlags

Get all block IO tunable parameters for a given device. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor, either for the given (note that block devices of different types might support different parameters), or if is NULL, for all possible disks. The caller should then allocate array, i.e. (sizeof() \* ) bytes and call the API again. See [virDomainGetMemoryParameters\(\)](#) for more details.

The parameter is either an unambiguous source name of the block device (the <source file="..."> sub-element, such as "/path/to/image"), or the device target shorthand (the <target dev="..."> sub-element, such as "xvda"). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within //domain/devices/disk. This parameter cannot be NULL unless is 0 on input.

Returns -1 in case of error, 0 in case of success.

**4.7.2.90** `int virDomainGetBlockJobInfo ( virDomainPtr dom, const char * disk, virDomainBlockJobInfoPtr info, unsigned int flags )`

virDomainGetBlockJobInfo: : pointer to domain object : path to the block device, or device shorthand : pointer to a virDomainBlockJobInfo structure : extra flags; not used yet, so callers should always pass 0

Request block job information for the given disk. If an operation is active will be updated with the current progress.

The parameter is either an unambiguous source name of the block device (the <source file="..."> sub-element, such as "/path/to/image"), or (since 0.9.5) the device target shorthand (the <target dev="..."> sub-element, such as "xvda"). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within //domain/devices/disk.

Returns -1 in case of failure, 0 when nothing found, 1 when info was found.

**4.7.2.91** `virConnectPtr virDomainGetConnect ( virDomainPtr dom )`

virDomainGetConnect: : pointer to a domain

Provides the connection pointer associated with a domain. The reference counter on the connection is not increased by this call.

**WARNING:** When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the domain object together.

Returns the virConnectPtr or NULL in case of failure.

**4.7.2.92** `int virDomainGetControllInfo ( virDomainPtr domain, virDomainControllInfoPtr info, unsigned int flags )`

virDomainGetControllInfo: : a domain object : pointer to a virDomainControllInfo structure allocated by the user : extra flags; not used yet, so callers should always pass 0

Extract details about current state of control interface to a domain.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.93** `int virDomainGetCPUStats ( virDomainPtr domain, virTypedParameterPtr params, unsigned int nparams, int start_cpu, unsigned int ncpus, unsigned int flags )`

virDomainGetCPUStats: : domain to query : array to populate on output : number of parameters per cpu : which cpu to start with, or -1 for summary : how many cpus to query : bitwise-OR of virTypedParameterFlags

Get statistics relating to CPU usage attributable to a single domain (in contrast to the statistics returned by [virNodeGetCPUStats\(\)](#) for all processes on the host). must be running (an inactive domain has no attributable cpu usage). On input, must contain at least \* entries, allocated by the caller.

If is -1, then must be 1, and the returned results reflect the statistics attributable to the entire domain (such as user and system time for the process as a whole). Otherwise, represents which cpu to start with, and represents how many consecutive processors to query, with statistics attributable per processor (such as per-cpu usage). If is larger than the number of cpus available to query, then the trailing part of the array will be unpopulated.

The remote driver imposes a limit of 128 and 16 ; the number of parameters per cpu should not exceed 16, but if you have a host with more than 128 CPUs, your program should split the request into multiple calls.

As special cases, if is NULL and is 0 and is 1, and the return value will be how many statistics are available for the given . This number may be different for of -1 than for any non-negative value, but will be the same for all non-negative . Likewise, if is NULL and is 0 and is 0, the number of cpus available to query is returned. From the host perspective, this would typically match the cpus member of [virNodeGetInfo\(\)](#), but might be less due to host cpu hotplug.

For now, is unused, and the statistics all relate to the usage from the host perspective. It is possible that a future version will support a flag that queries the cpu usage from the guest's perspective, where the maximum cpu to query would be related to [virDomainGetVcpusFlags\(\)](#) rather than [virNodeGetInfo\(\)](#). An individual guest vcpu cannot be

reliably mapped back to a specific host cpu unless a single-processor vcpu pinning was used, but when is -1, any difference in usage between a host and guest perspective would serve as a measure of hypervisor overhead.

Typical use sequence is below.

getting total stats: set start\_cpu as -1, ncpus 1 virDomainGetCPUStats(dom, NULL, 0, -1, 1, 0) => nparams  
params = calloc(nparams, sizeof(virTypedParameter)) virDomainGetCPUStats(dom, params, nparams, -1, 1, 0) => total stats.

getting per-cpu stats: virDomainGetCPUStats(dom, NULL, 0, 0, 0, 0) => ncpus virDomainGetCPUStats(dom, NULL, 0, 0, 1, 0) => nparams params = calloc(ncpus \* nparams, sizeof(virTypedParameter)) virDomainGetCPUStats(dom, params, nparams, 0, ncpus, 0) => per-cpu stats

Returns -1 on failure, or the number of statistics that were populated per cpu on success (this will be less than the total number of populated , unless was 1; and may be less than ). The populated parameters start at each stride of , which means the results may be discontinuous; any unpopulated parameters will be zeroed on success (this includes skipped elements if is too large, and tail elements if is too large). The caller is responsible for freeing any returned string parameters.

**4.7.2.94** int virDomainGetDiskErrors ( virDomainPtr dom, virDomainDiskErrorPtr errors, unsigned int maxerrors, unsigned int flags )

virDomainGetDiskErrors : a domain object : array to populate on output : size of array : extra flags; not used yet, so callers should always pass 0

The function populates array with all disks that encountered an I/O error. Disks with no error will not be returned in the array. Each disk is identified by its target (the dev attribute of target subelement in domain XML), such as "vda", and accompanied with the error that was seen on it. The caller is also responsible for calling free() on each disk name returned.

In a special case when is NULL and is 0, the function returns preferred size of that the caller should use to get all disk errors.

Since calling virDomainGetDiskErrors(dom, NULL, 0, 0) to get preferred size of array and getting the errors are two separate operations, new disks may be hotplugged to the domain and new errors may be encountered between the two calls. Thus, this function may not return all disk errors because the supplied array is not large enough. Such errors may, however, be detected by listening to domain events.

Returns number of disks with errors filled in the array or -1 on error.

**4.7.2.95** int virDomainGetEmulatorPinInfo ( virDomainPtr domain, unsigned char \* cpumap, int maplen, unsigned int flags )

virDomainGetEmulatorPinInfo : pointer to domain object, or NULL for Domain0 : pointer to a bit map of real CPUs for all emulator threads of this domain (in 8-bit bytes) (OUT) There is only one cpumap for all emulator threads. Must not be NULL. : the number of bytes in one cpumap, from 1 up to size of CPU map. Must be positive. : bitwise-OR of virDomainModificationImpact Must not be VIR\_DOMAIN\_AFFECT\_LIVE and VIR\_DOMAIN\_AFFECT\_CONFIG concurrently.

Query the CPU affinity setting of all emulator threads of domain, store it in cpumap.

Returns 1 in case of success, 0 in case of no emulator threads are pinned to pcpus, -1 in case of failure.

**4.7.2.96** char\* virDomainGetHostname ( virDomainPtr domain, unsigned int flags )

virDomainGetHostname : a domain object : extra flags; not used yet, so callers should always pass 0

Get the hostname for that domain.

Dependent on hypervisor used, this may require a guest agent to be available.

Returns the hostname which must be freed by the caller, or NULL if there was an error.

**4.7.2.97 unsigned int virDomainGetID ( virDomainPtr domain )**

virDomainGetID: : a domain object

Get the hypervisor ID number for the domain

Returns the domain ID number or (unsigned int) -1 in case of error

**4.7.2.98 int virDomainGetInfo ( virDomainPtr domain, virDomainInfoPtr info )**

virDomainGetInfo: : a domain object : pointer to a virDomainInfo structure allocated by the user

Extract information about a domain. Note that if the connection used to get the domain is limited only a partial set of the information can be extracted.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.99 int virDomainGetInterfaceParameters ( virDomainPtr domain, const char \* device, virTypedParameterPtr params, int \* nparams, unsigned int flags )**

virDomainGetInterfaceParameters: : pointer to domain object : the interface name or mac address : pointer to interface parameter objects (return value, allocated by the caller) : pointer to number of interface parameter; input and output : bitwise-OR of virDomainModificationImpact and virTypedParameterFlags

Get all interface parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. The caller should then allocate array, i.e. (sizeof() \* ) bytes and call the API again. See [virDomainGetMemoryParameters\(\)](#) for an equivalent usage example.

This function may require privileged access to the hypervisor. This function expects the caller to allocate the .

Returns -1 in case of error, 0 in case of success.

**4.7.2.100 int virDomainGetJobInfo ( virDomainPtr domain, virDomainJobInfoPtr info )**

virDomainGetJobInfo: : a domain object : pointer to a virDomainJobInfo structure allocated by the user

Extract information about progress of a background job on a domain. Will return an error if the domain is not active.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.101 unsigned long virDomainGetMaxMemory ( virDomainPtr domain )**

virDomainGetMaxMemory: : a domain object or NULL

Retrieve the maximum amount of physical memory allocated to a domain. If domain is NULL, then this get the amount of memory reserved to Domain0 i.e. the domain where the application runs.

Returns the memory size in kibibytes (blocks of 1024 bytes), or 0 in case of error.

**4.7.2.102 int virDomainGetMaxVcpus ( virDomainPtr domain )**

virDomainGetMaxVcpus: : pointer to domain object

Provides the maximum number of virtual CPUs supported for the guest VM. If the guest is inactive, this is basically the same as [virConnectGetMaxVcpus\(\)](#). If the guest is running this will reflect the maximum number of virtual CPUs the guest was booted with. For more details, see [virDomainGetVcpusFlags\(\)](#).

Returns the maximum of virtual CPU or -1 in case of error.

**4.7.2.103** `int virDomainGetMemoryParameters ( virDomainPtr domain, virTypedParameterPtr params, int * nparams, unsigned int flags )`

virDomainGetMemoryParameters: : pointer to domain object : pointer to memory parameter object (return value, allocated by the caller) : pointer to number of memory parameters; input and output : bitwise-OR of virDomainModificationImpact and virTypedParameterFlags

Get all memory parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. The caller should then allocate array, i.e. (sizeof() \* ) bytes and call the API again.

Here is a sample code snippet:

```
if ((virDomainGetMemoryParameters(dom, NULL, &nparams, 0) == 0) && (nparams != 0)) { if ((params = malloc(sizeof(*params) * nparams)) == NULL) goto error; memset(params, 0, sizeof(*params) * nparams); if (virDomainGetMemoryParameters(dom, params, &nparams, 0)) goto error; }
```

This function may require privileged access to the hypervisor. This function expects the caller to allocate the .

Returns -1 in case of error, 0 in case of success.

**4.7.2.104** `char* virDomainGetMetadata ( virDomainPtr domain, int type, const char * uri, unsigned int flags )`

virDomainGetMetadata: : a domain object : type of description, from virDomainMetadataType : XML namespace identifier : bitwise-OR of virDomainModificationImpact

Retrieves the appropriate domain element given by . If VIR\_DOMAIN\_METADATA\_ELEMENT is requested parameter must be set to the name of the namespace the requested elements belong to, otherwise must be NULL.

If an element of the domain XML is not present, the resulting error will be VIR\_ERR\_NO\_DOMAIN\_METADATA. This method forms a shortcut for seeing information from [virDomainSetMetadata\(\)](#) without having to go through [virDomainGetXMLDesc\(\)](#).

controls whether the live domain or persistent configuration will be queried.

Returns the metadata string on success (caller must free), or NULL in case of failure.

**4.7.2.105** `const char* virDomainGetName ( virDomainPtr domain )`

virDomainGetName: : a domain object

Get the public name for that domain

Returns a pointer to the name or NULL, the string need not be deallocated its lifetime will be the same as the domain object.

**4.7.2.106** `int virDomainGetNumaParameters ( virDomainPtr domain, virTypedParameterPtr params, int * nparams, unsigned int flags )`

virDomainGetNumaParameters: : pointer to domain object : pointer to numa parameter object (return value, allocated by the caller) : pointer to number of numa parameters : bitwise-OR of virDomainModificationImpact and virTypedParameterFlags

Get all numa parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value.

As a special case, calling with as NULL and as 0 on input will cause on output to contain the number of parameters supported by the hypervisor. The caller should then allocate array, i.e. (sizeof() \* ) bytes and call the API again.

See [virDomainGetMemoryParameters\(\)](#) for an equivalent usage example.

This function may require privileged access to the hypervisor. This function expects the caller to allocate the .

Returns -1 in case of error, 0 in case of success.

#### 4.7.2.107 `char* virDomainGetOSType ( virDomainPtr domain )`

`virDomainGetOSType`: : a domain object

Get the type of domain operation system.

Returns the new string or NULL in case of error, the string must be freed by the caller.

#### 4.7.2.108 `int virDomainGetSchedulerParameters ( virDomainPtr domain, virTypedParameterPtr params, int * nparams )`

`virDomainGetSchedulerParameters`: : pointer to domain object : pointer to scheduler parameter objects (return value) : pointer to number of scheduler parameter objects (this value should generally be as large as the returned value `nparams` of `virDomainGetSchedulerType()`); input and output

Get all scheduler parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value. cannot be 0.

It is hypervisor specific whether this returns the live or persistent state; for more control, use `virDomainGetSchedulerParametersFlags()`.

Returns -1 in case of error, 0 in case of success.

#### 4.7.2.109 `int virDomainGetSchedulerParametersFlags ( virDomainPtr domain, virTypedParameterPtr params, int * nparams, unsigned int flags )`

`virDomainGetSchedulerParametersFlags`: : pointer to domain object : pointer to scheduler parameter object (return value) : pointer to number of scheduler parameter (this value should be same than the returned value `nparams` of `virDomainGetSchedulerType()`); input and output : bitwise-OR of `virDomainModificationImpact` and `virTypedParameterFlags`

Get all scheduler parameters. On input, gives the size of the array; on output, gives how many slots were filled with parameter information, which might be less but will not exceed the input value. cannot be 0.

The value of can be exactly `VIR_DOMAIN_AFFECT_CURRENT`, `VIR_DOMAIN_AFFECT_LIVE`, or `VIR_DOMAIN_AFFECT_CONFIG`.

Here is a sample code snippet:

```
char *ret = virDomainGetSchedulerType(dom, &nparams); if (ret && nparams != 0) { if ((params = malloc(sizeof(*params) * nparams)) == NULL) goto error; memset(params, 0, sizeof(*params) * nparams); if (virDomainGetSchedulerParametersFlags(dom, params, &nparams, 0)) goto error; }
```

Returns -1 in case of error, 0 in case of success.

#### 4.7.2.110 `char* virDomainGetSchedulerType ( virDomainPtr domain, int * nparams )`

`virDomainGetSchedulerType`: : pointer to domain object : pointer to number of scheduler parameters, can be NULL (return value)

Get the scheduler type and the number of scheduler parameters.

Returns NULL in case of error. The caller must free the returned string.

#### 4.7.2.111 `int virDomainGetSecurityLabel ( virDomainPtr domain, virSecurityLabelPtr seclabel )`

`virDomainGetSecurityLabel`: : a domain object : pointer to a `virSecurityLabel` structure

Extract security label of an active domain. The 'label' field in the argument will be initialized to the empty string if the domain is not running under a security model.



Returns 0 in case of success, -1 in case of failure

#### 4.7.2.112 int virDomainGetSecurityLabelList ( virDomainPtr domain, virSecurityLabelPtr \* seclabels )

virDomainGetSecurityLabelList: : a domain object : will be auto-allocated and filled with domains' security labels. Caller must free memory on return.

Extract the security labels of an active domain. The 'label' field in the argument will be initialized to the empty string if the domain is not running under a security model.

Returns number of elements in on success, -1 in case of failure.

#### 4.7.2.113 int virDomainGetState ( virDomainPtr domain, int \* state, int \* reason, unsigned int flags )

virDomainGetState: : a domain object : returned state of the domain (one of virDomainState) : returned reason which led to (one of virDomainReason corresponding to the current state); it is allowed to be NULL : extra flags; not used yet, so callers should always pass 0

Extract domain state. Each state can be accompanied with a reason (if known) which led to the state.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.114 int virDomainGetUUID ( virDomainPtr domain, unsigned char \* uuid )

virDomainGetUUID: : a domain object : pointer to a VIR\_UUID\_BUFLen bytes array

Get the UUID for a domain

Returns -1 in case of error, 0 in case of success

#### 4.7.2.115 int virDomainGetUUIDString ( virDomainPtr domain, char \* buf )

virDomainGetUUIDString: : a domain object : pointer to a VIR\_UUID\_STRING\_BUFLen bytes array

Get the UUID for a domain as string. For more information about UUID see RFC4122.

Returns -1 in case of error, 0 in case of success

#### 4.7.2.116 int virDomainGetVcpuPinInfo ( virDomainPtr domain, int ncpumaps, unsigned char \* cpumaps, int maplen, unsigned int flags )

virDomainGetVcpuPinInfo: : pointer to domain object, or NULL for Domain0 : the number of cpumap (listed first to match virDomainGetVcpus) : pointer to a bit map of real CPUs for all vcpus of this domain (in 8-bit bytes) (OUT) It's assumed there is <ncpumaps> cpumap in cpumaps array. The memory allocated to cpumaps must be (ncpumaps \* maplen) bytes (ie: calloc(ncpumaps, maplen)). One cpumap inside cpumaps has the format described in [virDomainPinVcpu\(\)](#) API. Must not be NULL. : the number of bytes in one cpumap, from 1 up to size of CPU map. Must be positive. : bitwise-OR of virDomainModificationImpact Must not be VIR\_DOMAIN\_AFFECT\_LIVE and VIR\_DOMAIN\_AFFECT\_CONFIG concurrently.

Query the CPU affinity setting of all virtual CPUs of domain, store it in cpumaps.

Returns the number of virtual CPUs in case of success, -1 in case of failure.

#### 4.7.2.117 int virDomainGetVcpus ( virDomainPtr domain, virVcpuInfoPtr info, int maxinfo, unsigned char \* cpumaps, int maplen )

virDomainGetVcpus: : pointer to domain object, or NULL for Domain0 : pointer to an array of virVcpuInfo structures (OUT) : number of structures in info array : pointer to a bit map of real CPUs for all vcpus of this domain (in 8-bit bytes) (OUT) If cpumaps is NULL, then no cpumap information is returned by the API. It's assumed there is

<maxinfo> cpumap in cpumaps array. The memory allocated to cpumaps must be (maxinfo \* maplen) bytes (ie: calloc(maxinfo, maplen)). One cpumap inside cpumaps has the format described in [virDomainPinVcpu\(\)](#) API. : number of bytes in one cpumap, from 1 up to size of CPU map in underlying virtualization system (Xen...). Must be zero when cpumaps is NULL and positive when it is non-NULL.

Extract information about virtual CPUs of domain, store it in info array and also in cpumaps if this pointer isn't NULL. This call may fail on an inactive domain.

See also [virDomainGetVcpuPinInfo](#) for querying just cpumaps, including on an inactive domain.

Returns the number of info filled in case of success, -1 in case of failure.

#### 4.7.2.118 int virDomainGetVcpusFlags ( virDomainPtr domain, unsigned int flags )

virDomainGetVcpusFlags: : pointer to domain object, or NULL for Domain0 : bitwise-OR of virDomainVcpuFlags

Query the number of virtual CPUs used by the domain. Note that this call may fail if the underlying virtualization hypervisor does not support it. This function may require privileged access to the hypervisor.

If includes VIR\_DOMAIN\_AFFECT\_LIVE, this will query a running domain (which will fail if domain is not active); if it includes VIR\_DOMAIN\_AFFECT\_CONFIG, this will query the XML description of the domain. It is an error to set both flags. If neither flag is set (that is, VIR\_DOMAIN\_AFFECT\_CURRENT), then the configuration queried depends on whether the domain is currently running.

If includes VIR\_DOMAIN\_VCPU\_MAXIMUM, then the maximum virtual CPU limit is queried. Otherwise, this call queries the current virtual CPU limit.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.119 char\* virDomainGetXMLDesc ( virDomainPtr domain, unsigned int flags )

virDomainGetXMLDesc: : a domain object : bitwise-OR of virDomainXMLFlags

Provide an XML description of the domain. The description may be reused later to relaunch the domain with [virDomainCreateXML\(\)](#).

No security-sensitive data will be included unless contains VIR\_DOMAIN\_XML\_SECURE; this flag is rejected on read-only connections. If includes VIR\_DOMAIN\_XML\_INACTIVE, then the XML represents the configuration that will be used on the next boot of a persistent domain; otherwise, the configuration represents the currently running domain. If contains VIR\_DOMAIN\_XML\_UPDATE\_CPU, then the portion of the domain XML describing CPU capabilities is modified to match actual capabilities of the host.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

#### 4.7.2.120 int virDomainHasCurrentSnapshot ( virDomainPtr domain, unsigned int flags )

virDomainHasCurrentSnapshot: : pointer to the domain object : extra flags; not used yet, so callers should always pass 0

Determine if the domain has a current snapshot.

Returns 1 if such snapshot exists, 0 if it doesn't, -1 on error.

#### 4.7.2.121 int virDomainHasManagedSaveImage ( virDomainPtr dom, unsigned int flags )

virDomainHasManagedSaveImage: : pointer to the domain : extra flags; not used yet, so callers should always pass 0

Check if a domain has a managed save image as created by [virDomainManagedSave\(\)](#). Note that any running domain should not have such an image, as it should have been removed on restart.

Returns 0 if no image is present, 1 if an image is present, and -1 in case of error

**4.7.2.122 int virDomainInjectNMI ( virDomainPtr domain, unsigned int flags )**

virDomainInjectNMI: : pointer to domain object, or NULL for Domain0 : extra flags; not used yet, so callers should always pass 0

Send NMI to the guest

Returns 0 in case of success, -1 in case of failure.

**4.7.2.123 int virDomainInterfaceStats ( virDomainPtr dom, const char \* path, virDomainInterfaceStatsPtr stats, size\_t size )**

virDomainInterfaceStats: : pointer to the domain object : path to the interface : network interface stats (returned) : size of stats structure

This function returns network interface stats for interfaces attached to the domain.

The path parameter is the name of the network interface.

Domains may have more than one network interface. To get stats for each you should make multiple calls to this function.

Individual fields within the stats structure may be returned as -1, which indicates that the hypervisor does not support that particular statistic.

Returns: 0 in case of success or -1 in case of failure.

**4.7.2.124 int virDomainsActive ( virDomainPtr dom )**

virDomainsActive: : pointer to the domain object

Determine if the domain is currently running

Returns 1 if running, 0 if inactive, -1 on error

**4.7.2.125 int virDomainsPersistent ( virDomainPtr dom )**

virDomainsPersistent: : pointer to the domain object

Determine if the domain has a persistent configuration which means it will still exist after shutting down

Returns 1 if persistent, 0 if transient, -1 on error

**4.7.2.126 int virDomainsUpdated ( virDomainPtr dom )**

virDomainsUpdated: : pointer to the domain object

Determine if the domain has been updated.

Returns 1 if updated, 0 if not, -1 on error

**4.7.2.127 int virDomainListAllSnapshots ( virDomainPtr domain, virDomainSnapshotPtr \*\* snaps, unsigned int flags )**

virDomainListAllSnapshots: : a domain object : pointer to variable to store the array containing snapshot objects, or NULL if the list is not required (just returns number of snapshots) : bitwise-OR of supported virDomainSnapshotListFlags

Collect the list of domain snapshots for the given domain, and allocate an array to store those objects. This API solves the race inherent in [virDomainSnapshotListNames\(\)](#).

By default, this command covers all snapshots; it is also possible to limit things to just snapshots with no parents, when includes VIR\_DOMAIN\_SNAPSHOT\_LIST\_ROOTS. Additional filters are provided in groups, where each

group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling `virDomainSnapshotFree()` on each array element, then calling `free()` on .

#### 4.7.2.128 `virDomainPtr virDomainLookupByID ( virConnectPtr conn, int id )`

`virDomainLookupByID`: : pointer to the hypervisor connection : the domain ID number

Try to find a domain based on the hypervisor ID number Note that this won't work for inactive domains which have an ID of -1, in that case a lookup based on the Name or UUID need to be done instead.

Returns a new domain object or NULL in case of failure. If the domain cannot be found, then `VIR_ERR_NO_DOMAIN` error is raised.

#### 4.7.2.129 `virDomainPtr virDomainLookupByName ( virConnectPtr conn, const char * name )`

#### 4.7.2.130 `virDomainPtr virDomainLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

`virDomainLookupByUUID`: : pointer to the hypervisor connection : the raw UUID for the domain

Try to lookup a domain on the given hypervisor based on its UUID.

Returns a new domain object or NULL in case of failure. If the domain cannot be found, then `VIR_ERR_NO_DOMAIN` error is raised.

#### 4.7.2.131 `virDomainPtr virDomainLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

`virDomainLookupByUUIDString`: : pointer to the hypervisor connection : the string UUID for the domain

Try to lookup a domain on the given hypervisor based on its UUID.

Returns a new domain object or NULL in case of failure. If the domain cannot be found, then `VIR_ERR_NO_DOMAIN` error is raised.

#### 4.7.2.132 `int virDomainManagedSave ( virDomainPtr dom, unsigned int flags )`

`virDomainManagedSave`: : pointer to the domain : bitwise-OR of `virDomainSaveRestoreFlags`

This method will suspend a domain and save its memory contents to a file on disk. After the call, if successful, the domain is not listed as running anymore. The difference from `virDomainSave()` is that libvirt is keeping track of the saved state itself, and will reuse it once the domain is being restarted (automatically or via an explicit libvirt call). As a result any running domain is sure to not have a managed saved image. This also implies that managed save only works on persistent domains, since the domain must still exist in order to use `virDomainCreate()` to restart it.

If includes `VIR_DOMAIN_SAVE_BYPASS_CACHE`, then libvirt will attempt to bypass the file system cache while creating the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing saves to NFS.

Normally, the managed saved state will remember whether the domain was running or paused, and start will resume to the same state. Specifying `VIR_DOMAIN_SAVE_RUNNING` or `VIR_DOMAIN_SAVE_PAUSED` in will override the default saved into the file. These two flags are mutually exclusive.

Returns 0 in case of success or -1 in case of failure

#### 4.7.2.133 `int virDomainManagedSaveRemove ( virDomainPtr dom, unsigned int flags )`

`virDomainManagedSaveRemove`: : pointer to the domain : extra flags; not used yet, so callers should always pass 0

Remove any managed save image for this domain.

Returns 0 in case of success, and -1 in case of error

#### 4.7.2.134 `int virDomainMemoryPeek ( virDomainPtr dom, unsigned long long start, size_t size, void * buffer, unsigned int flags )`

`virDomainMemoryPeek`: : pointer to the domain object : start of memory to peek : size of memory to peek : return buffer (must be at least size bytes) : bitwise-OR of `virDomainMemoryFlags`

This function allows you to read the contents of a domain's memory.

The memory which is read is controlled by the 'start', 'size' and 'flags' parameters.

If 'flags' is `VIR_MEMORY_VIRTUAL` then the 'start' and 'size' parameters are interpreted as virtual memory addresses for whichever task happens to be running on the domain at the moment. Although this sounds haphazard it is in fact what you want in order to read Linux kernel state, because it ensures that pointers in the kernel image can be interpreted coherently.

'buffer' is the return buffer and must be at least 'size' bytes. 'size' may be 0 to test if the call would succeed.

NB. The remote driver imposes a 64K byte limit on 'size'. For your program to be able to work reliably over a remote connection you should split large requests to  $\leq 65536$  bytes. However, with 0.9.13 this RPC limit has been raised to 1M byte.

Returns: 0 in case of success or -1 in case of failure.

#### 4.7.2.135 `int virDomainMemoryStats ( virDomainPtr dom, virDomainMemoryStatPtr stats, unsigned int nr_stats, unsigned int flags )`

`virDomainMemoryStats`: : pointer to the domain object : nr\_stats-sized array of stat structures (returned) : number of memory statistics requested : extra flags; not used yet, so callers should always pass 0

This function provides memory statistics for the domain.

Up to 'nr\_stats' elements of 'stats' will be populated with memory statistics from the domain. Only statistics supported by the domain, the driver, and this version of libvirt will be returned.

Memory Statistics:

`VIR_DOMAIN_MEMORY_STAT_SWAP_IN`: The total amount of data read from swap space (in kb). `VIR_DOMAIN_MEMORY_STAT_SWAP_OUT`: The total amount of memory written out to swap space (in kb). `VIR_DOMAIN_MEMORY_STAT_MAJOR_FAULT`: The number of page faults that required disk IO to service. `VIR_DOMAIN_MEMORY_STAT_MINOR_FAULT`: The number of page faults serviced without disk IO. `VIR_DOMAIN_MEMORY_STAT_UNUSED`: The amount of memory which is not being used for any purpose (in kb). `VIR_DOMAIN_MEMORY_STAT_AVAILABLE`: The total amount of memory available to the domain's OS (in kb). `VIR_DOMAIN_MEMORY_STAT_ACTUAL_BALLOON`: Current balloon value (in kb).

Returns: The number of stats provided or -1 in case of failure.

**4.7.2.136** `virDomainPtr virDomainMigrate ( virDomainPtr domain, virConnectPtr dconn, unsigned long flags, const char * dname, const char * uri, unsigned long bandwidth )`

`virDomainMigrate`: : a domain object : destination host (a connection object) : bitwise-OR of `virDomainMigrate-Flags` : (optional) rename domain to this at destination : (optional) dest hostname/URI as seen from the source host : (optional) specify migration bandwidth limit in Mbps

Migrate the domain object from its current host to the destination host given by `dconn` (a connection to the destination host).

Flags may be one or more of the following: `VIR_MIGRATE_LIVE` Do not pause the VM during migration `VIR_MIGRATE_PEER2PEER` Direct connection between source & destination hosts `VIR_MIGRATE_TUNNELLED` Tunnel migration data over the libvirt RPC channel `VIR_MIGRATE_PERSIST_DEST` If the migration is successful, persist the domain on the destination host. `VIR_MIGRATE_UNDEFINE_SOURCE` If the migration is successful, undefine the domain on the source host. `VIR_MIGRATE_PAUSED` Leave the domain suspended on the remote side. `VIR_MIGRATE_CHANGE_PROTECTION` Protect against domain configuration changes during the migration process (set automatically when supported). `VIR_MIGRATE_UNSAFE` Force migration even if it is considered unsafe.

`VIR_MIGRATE_TUNNELLED` requires that `VIR_MIGRATE_PEER2PEER` be set. Applications using the `VIR_MIGRATE_PEER2PEER` flag will probably prefer to invoke `virDomainMigrateToURI`, avoiding the need to open connection to the destination host themselves.

If a hypervisor supports renaming domains during migration, then you may set the `dname` parameter to the new name (otherwise it keeps the same name). If this is not supported by the hypervisor, `dname` must be `NULL` or else you will get an error.

If the `VIR_MIGRATE_PEER2PEER` flag is set, the `uri` parameter must be a valid libvirt connection URI, by which the source libvirt driver can connect to the destination libvirt. If omitted, the `dconn` connection object will be queried for its current URI.

If the `VIR_MIGRATE_PEER2PEER` flag is NOT set, the URI parameter takes a hypervisor specific format. The hypervisor capabilities XML includes details of the support URI schemes. If omitted the `dconn` will be asked for a default URI.

In either case it is typically only necessary to specify a URI if the destination host has multiple interfaces and a specific interface is required to transmit migration data.

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the `bandwidth` parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0.

To see which features are supported by the current hypervisor, see `virConnectGetCapabilities`, `/capabilities/host/migration_features`.

There are many limitations on migration imposed by the underlying technology - for example it may not be possible to migrate between different processors even with the same architecture, or between different types of hypervisor.

Returns the new domain object if the migration was successful, or `NULL` in case of error. Note that the new domain object exists in the scope of the destination connection (`dconn`).

**4.7.2.137** `virDomainPtr virDomainMigrate2 ( virDomainPtr domain, virConnectPtr dconn, const char * dxml, unsigned long flags, const char * dname, const char * uri, unsigned long bandwidth )`

`virDomainMigrate2`: : a domain object : destination host (a connection object) : bitwise-OR of `virDomainMigrate-Flags` : (optional) XML config for launching guest on target : (optional) rename domain to this at destination : (optional) dest hostname/URI as seen from the source host : (optional) specify migration bandwidth limit in Mbps

Migrate the domain object from its current host to the destination host given by `dconn` (a connection to the destination host).

Flags may be one or more of the following: `VIR_MIGRATE_LIVE` Do not pause the VM during migration `VIR_MIGRATE_PEER2PEER` Direct connection between source & destination hosts `VIR_MIGRATE_TUNNELLED` Tunnel migration data over the libvirt RPC channel `VIR_MIGRATE_PERSIST_DEST` If the migration is successful, persist the domain on the destination host. `VIR_MIGRATE_UNDEFINE_SOURCE` If the migration is successful, undefine

the domain on the source host. `VIR_MIGRATE_PAUSED` Leave the domain suspended on the remote side. `VIR_MIGRATE_CHANGE_PROTECTION` Protect against domain configuration changes during the migration process (set automatically when supported). `VIR_MIGRATE_UNSAFE` Force migration even if it is considered unsafe.

`VIR_MIGRATE_TUNNELLED` requires that `VIR_MIGRATE_PEER2PEER` be set. Applications using the `VIR_MIGRATE_PEER2PEER` flag will probably prefer to invoke `virDomainMigrateToURI`, avoiding the need to open connection to the destination host themselves.

If a hypervisor supports renaming domains during migration, then you may set the `dname` parameter to the new name (otherwise it keeps the same name). If this is not supported by the hypervisor, `dname` must be `NULL` or else you will get an error.

If the `VIR_MIGRATE_PEER2PEER` flag is set, the `uri` parameter must be a valid libvirt connection URI, by which the source libvirt driver can connect to the destination libvirt. If omitted, the `dconn` connection object will be queried for its current URI.

If the `VIR_MIGRATE_PEER2PEER` flag is NOT set, the URI parameter takes a hypervisor specific format. The hypervisor capabilities XML includes details of the support URI schemes. If omitted the `dconn` will be asked for a default URI.

In either case it is typically only necessary to specify a URI if the destination host has multiple interfaces and a specific interface is required to transmit migration data.

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the `bandwidth` parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0.

To see which features are supported by the current hypervisor, see `virConnectGetCapabilities`, `/capabilities/host/migration-features`.

There are many limitations on migration imposed by the underlying technology - for example it may not be possible to migrate between different processors even with the same architecture, or between different types of hypervisor.

If the hypervisor supports it, can be used to alter host-specific portions of the domain XML that will be used on the destination. For example, it is possible to alter the backing filename that is associated with a disk device, in order to account for naming differences between source and destination in accessing the underlying storage. The migration will fail if would cause any guest-visible changes. Pass `NULL` if no changes are needed to the XML between source and destination. cannot be used to rename the domain during migration (use for that purpose). Domain name in must match the original domain name.

Returns the new domain object if the migration was successful, or `NULL` in case of error. Note that the new domain object exists in the scope of the destination connection (`dconn`).

**4.7.2.138** `char* virDomainMigrateBegin3 ( virDomainPtr domain, const char * xmlin, char ** cookieout, int * cookieoutlen, unsigned long flags, const char * dname, unsigned long bandwidth )`

**4.7.2.139** `int virDomainMigrateConfirm3 ( virDomainPtr domain, const char * cookiein, int cookieinlen, unsigned long flags, int cancelled )`

**4.7.2.140** `static int virDomainMigrateDirect ( virDomainPtr domain, const char * xmlin, unsigned long flags, const char * dname, const char * uri, unsigned long bandwidth ) [static]`

**4.7.2.141** `virDomainPtr virDomainMigrateFinish ( virConnectPtr dconn, const char * dname, const char * cookie, int cookielen, const char * uri, unsigned long flags )`

**4.7.2.142** `virDomainPtr virDomainMigrateFinish2 ( virConnectPtr dconn, const char * dname, const char * cookie, int cookielen, const char * uri, unsigned long flags, int retcode )`

**4.7.2.143** `virDomainPtr virDomainMigrateFinish3 ( virConnectPtr dconn, const char * dname, const char * cookiein, int cookieinlen, char ** cookieout, int * cookieoutlen, const char * dconnuri, const char * uri, unsigned long flags, int cancelled )`

4.7.2.144 `int virDomainMigrateGetMaxSpeed ( virDomainPtr domain, unsigned long * bandwidth, unsigned int flags )`

`virDomainMigrateGetMaxSpeed`: : a domain object : return value of current migration bandwidth limit in Mbps : extra flags; not used yet, so callers should always pass 0

Get the current maximum bandwidth (in Mbps) that will be used if the domain is migrated. Not all hypervisors will support a bandwidth limit.

Returns 0 in case of success, -1 otherwise.

4.7.2.145 `static int virDomainMigratePeer2Peer ( virDomainPtr domain, const char * xmlin, unsigned long flags, const char * dname, const char * dconnuri, const char * uri, unsigned long bandwidth ) [static]`

4.7.2.146 `int virDomainMigratePerform ( virDomainPtr domain, const char * cookie, int cookieinlen, const char * uri, unsigned long flags, const char * dname, unsigned long bandwidth )`

4.7.2.147 `int virDomainMigratePerform3 ( virDomainPtr domain, const char * xmlin, const char * cookiein, int cookieinlen, char ** cookieout, int * cookieoutlen, const char * dconnuri, const char * uri, unsigned long flags, const char * dname, unsigned long bandwidth )`

4.7.2.148 `int virDomainMigratePrepare ( virConnectPtr dconn, char ** cookie, int * cookieinlen, const char * uri_in, char ** uri_out, unsigned long flags, const char * dname, unsigned long bandwidth )`

4.7.2.149 `int virDomainMigratePrepare2 ( virConnectPtr dconn, char ** cookie, int * cookieinlen, const char * uri_in, char ** uri_out, unsigned long flags, const char * dname, unsigned long bandwidth, const char * dom_xml )`

4.7.2.150 `int virDomainMigratePrepare3 ( virConnectPtr dconn, const char * cookiein, int cookieinlen, char ** cookieout, int * cookieoutlen, const char * uri_in, char ** uri_out, unsigned long flags, const char * dname, unsigned long bandwidth, const char * dom_xml )`

4.7.2.151 `int virDomainMigratePrepareTunnel ( virConnectPtr conn, virStreamPtr st, unsigned long flags, const char * dname, unsigned long bandwidth, const char * dom_xml )`

4.7.2.152 `int virDomainMigratePrepareTunnel3 ( virConnectPtr conn, virStreamPtr st, const char * cookiein, int cookieinlen, char ** cookieout, int * cookieoutlen, unsigned long flags, const char * dname, unsigned long bandwidth, const char * dom_xml )`

4.7.2.153 `int virDomainMigrateSetMaxDowntime ( virDomainPtr domain, unsigned long long downtime, unsigned int flags )`

`virDomainMigrateSetMaxDowntime`: : a domain object : maximum tolerable downtime for live migration, in milliseconds : extra flags; not used yet, so callers should always pass 0

Sets maximum tolerable time for which the domain is allowed to be paused at the end of live migration. It's supposed to be called while the domain is being live-migrated as a reaction to migration progress.

Returns 0 in case of success, -1 otherwise.

4.7.2.154 `int virDomainMigrateSetMaxSpeed ( virDomainPtr domain, unsigned long bandwidth, unsigned int flags )`

`virDomainMigrateSetMaxSpeed`: : a domain object : migration bandwidth limit in Mbps : extra flags; not used yet, so callers should always pass 0

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the bandwidth parameter. Not all hypervisors will support a bandwidth cap

Returns 0 in case of success, -1 otherwise.



**4.7.2.155** `int virDomainMigrateToURI ( virDomainPtr domain, const char * duri, unsigned long flags, const char * dname, unsigned long bandwidth )`

virDomainMigrateToURI: : a domain object : mandatory URI for the destination host : bitwise-OR of virDomainMigrateFlags : (optional) rename domain to this at destination : (optional) specify migration bandwidth limit in Mbps

Migrate the domain object from its current host to the destination host given by duri.

Flags may be one of more of the following: VIR\_MIGRATE\_LIVE Do not pause the VM during migration VIR\_MIGRATE\_PEER2PEER Direct connection between source & destination hosts VIR\_MIGRATE\_TUNNELLED Tunnel migration data over the libvirt RPC channel VIR\_MIGRATE\_PERSIST\_DEST If the migration is successful, persist the domain on the destination host. VIR\_MIGRATE\_UNDEFINE\_SOURCE If the migration is successful, undefine the domain on the source host. VIR\_MIGRATE\_CHANGE\_PROTECTION Protect against domain configuration changes during the migration process (set automatically when supported). VIR\_MIGRATE\_UNSAFE Force migration even if it is considered unsafe.

The operation of this API hinges on the VIR\_MIGRATE\_PEER2PEER flag. If the VIR\_MIGRATE\_PEER2PEER flag is NOT set, the duri parameter takes a hypervisor specific format. The uri\_transports element of the hypervisor capabilities XML includes details of the supported URI schemes. Not all hypervisors will support this mode of migration, so if the VIR\_MIGRATE\_PEER2PEER flag is not set, then it may be necessary to use the alternative virDomainMigrate API providing and explicit virConnectPtr for the destination host.

If the VIR\_MIGRATE\_PEER2PEER flag IS set, the duri parameter must be a valid libvirt connection URI, by which the source libvirt driver can connect to the destination libvirt.

VIR\_MIGRATE\_TUNNELLED requires that VIR\_MIGRATE\_PEER2PEER be set.

If a hypervisor supports renaming domains during migration, the dname parameter specifies the new name for the domain. Setting dname to NULL keeps the domain name the same. If domain renaming is not supported by the hypervisor, dname must be NULL or else an error will be returned.

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the bandwidth parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0.

To see which features are supported by the current hypervisor, see virConnectGetCapabilities, /capabilities/host/migration\_features.

There are many limitations on migration imposed by the underlying technology - for example it may not be possible to migrate between different processors even with the same architecture, or between different types of hypervisor.

Returns 0 if the migration succeeded, -1 upon error.

**4.7.2.156** `int virDomainMigrateToURI2 ( virDomainPtr domain, const char * dconnuri, const char * miguri, const char * dxml, unsigned long flags, const char * dname, unsigned long bandwidth )`

virDomainMigrateToURI2: : a domain object : (optional) URI for target libvirtd if includes VIR\_MIGRATE\_PEER2PEER : (optional) URI for invoking the migration, not if includes VIR\_MIGRATE\_TUNNELLED : (optional) XML config for launching guest on target : bitwise-OR of virDomainMigrateFlags : (optional) rename domain to this at destination : (optional) specify migration bandwidth limit in Mbps

Migrate the domain object from its current host to the destination host given by duri.

Flags may be one of more of the following: VIR\_MIGRATE\_LIVE Do not pause the VM during migration VIR\_MIGRATE\_PEER2PEER Direct connection between source & destination hosts VIR\_MIGRATE\_TUNNELLED Tunnel migration data over the libvirt RPC channel VIR\_MIGRATE\_PERSIST\_DEST If the migration is successful, persist the domain on the destination host. VIR\_MIGRATE\_UNDEFINE\_SOURCE If the migration is successful, undefine the domain on the source host. VIR\_MIGRATE\_CHANGE\_PROTECTION Protect against domain configuration changes during the migration process (set automatically when supported). VIR\_MIGRATE\_UNSAFE Force migration even if it is considered unsafe.

The operation of this API hinges on the VIR\_MIGRATE\_PEER2PEER flag.

If the VIR\_MIGRATE\_PEER2PEER flag is set, the parameter must be a valid libvirt connection URI, by which the source libvirt driver can connect to the destination libvirt. If the VIR\_MIGRATE\_PEER2PEER flag is NOT set, then

must be NULL.

If the VIR\_MIGRATE\_TUNNELLED flag is NOT set, then the parameter allows specification of a URI to use to initiate the VM migration. It takes a hypervisor specific format. The uri\_transports element of the hypervisor capabilities XML includes details of the supported URI schemes.

VIR\_MIGRATE\_TUNNELLED requires that VIR\_MIGRATE\_PEER2PEER be set.

If a hypervisor supports changing the configuration of the guest during migration, the parameter specifies the new config for the guest. The configuration must include an identical set of virtual devices, to ensure a stable guest ABI across migration. Only parameters related to host side configuration can be changed in the XML. Hypervisors will validate this and refuse to allow migration if the provided XML would cause a change in the guest ABI,

If a hypervisor supports renaming domains during migration, the dname parameter specifies the new name for the domain. Setting dname to NULL keeps the domain name the same. If domain renaming is not supported by the hypervisor, dname must be NULL or else an error will be returned.

The maximum bandwidth (in Mbps) that will be used to do migration can be specified with the bandwidth parameter. If set to 0, libvirt will choose a suitable default. Some hypervisors do not support this feature and will return an error if bandwidth is not 0.

To see which features are supported by the current hypervisor, see virConnectGetCapabilities, /capabilities/host/migration-features.

There are many limitations on migration imposed by the underlying technology - for example it may not be possible to migrate between different processors even with the same architecture, or between different types of hypervisor.

Returns 0 if the migration succeeded, -1 upon error.

4.7.2.157 static virDomainPtr virDomainMigrateVersion1 ( virDomainPtr domain, virConnectPtr dconn, unsigned long flags, const char \* dname, const char \* uri, unsigned long bandwidth ) [static]

4.7.2.158 static virDomainPtr virDomainMigrateVersion2 ( virDomainPtr domain, virConnectPtr dconn, unsigned long flags, const char \* dname, const char \* uri, unsigned long bandwidth ) [static]

4.7.2.159 static virDomainPtr virDomainMigrateVersion3 ( virDomainPtr domain, virConnectPtr dconn, const char \* xmlin, unsigned long flags, const char \* dname, const char \* uri, unsigned long bandwidth ) [static]

4.7.2.160 int virDomainOpenConsole ( virDomainPtr dom, const char \* dev\_name, virStreamPtr st, unsigned int flags )

virDomainOpenConsole: : a domain object : the console, serial or parallel port device alias, or NULL : a stream to associate with the console : bitwise-OR of virDomainConsoleFlags

This opens the backend associated with a console, serial or parallel port device on a guest, if the backend is supported. If the is omitted, then the first console or serial device is opened. The console is associated with the passed in stream, which should have been opened in non-blocking mode for bi-directional I/O.

By default, when is 0, the open will fail if libvirt detects that the console is already in use by another client; passing VIR\_DOMAIN\_CONSOLE\_FORCE will cause libvirt to forcefully remove the other client prior to opening this console.

If flag VIR\_DOMAIN\_CONSOLE\_SAFE the console is opened only in the case where the hypervisor driver supports safe (mutually exclusive) console handling.

Older servers did not support either flag, and also did not forbid simultaneous clients on a console, with potentially confusing results. When passing of 0 in order to support a wider range of server versions, it is up to the client to ensure mutual exclusion.

Returns 0 if the console was opened, -1 on error

4.7.2.161 int virDomainOpenGraphics ( virDomainPtr dom, unsigned int idx, int fd, unsigned int flags )

virDomainOpenGraphics: : pointer to domain object : index of graphics config to open : file descriptor to attach graphics to : bitwise-OR of virDomainOpenGraphicsFlags

This will attempt to connect the file descriptor , to the graphics backend of . If has multiple graphics backends configured, then will determine which one is opened, starting from 0.

To disable any authentication, pass the VIR\_DOMAIN\_OPEN\_GRAPHICS\_SKIPAUTH constant for .

The caller should use an anonymous socketpair to open before invocation.

This method can only be used when connected to a local libvirt hypervisor, over a UNIX domain socket. Attempts to use this method over a TCP connection will always fail

Returns 0 on success, -1 on failure

#### 4.7.2.162 int virDomainPinEmulator ( virDomainPtr domain, unsigned char \* cpumap, int maplen, unsigned int flags )

virDomainPinEmulator: : pointer to domain object, or NULL for Domain0 : pointer to a bit map of real CPUs (in 8-bit bytes) (IN) Each bit set to 1 means that corresponding CPU is usable. Bytes are stored in little-endian order: CPU0-7, 8-15... In each byte, lowest CPU number is least significant bit. : number of bytes in cpumap, from 1 up to size of CPU map in underlying virtualization system (Xen...). If maplen < size, missing bytes are set to zero. If maplen > size, failure code is returned. : bitwise-OR of virDomainModificationImpact

Dynamically change the real CPUs which can be allocated to all emulator threads. This function may require privileged access to the hypervisor.

may include VIR\_DOMAIN\_AFFECT\_LIVE or VIR\_DOMAIN\_AFFECT\_CONFIG. Both flags may be set. If VIR\_DOMAIN\_AFFECT\_LIVE is set, the change affects a running domain and may fail if domain is not alive. If VIR\_DOMAIN\_AFFECT\_CONFIG is set, the change affects persistent state, and will fail for transient domains. If neither flag is specified (that is, is VIR\_DOMAIN\_AFFECT\_CURRENT), then an inactive domain modifies persistent setup, while an active domain is hypervisor-dependent on whether just live or both live and persistent state is changed. Not all hypervisors can support all flag combinations.

See also virDomainGetEmulatorPinInfo for querying this information.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.163 int virDomainPinVcpu ( virDomainPtr domain, unsigned int vcpu, unsigned char \* cpumap, int maplen )

virDomainPinVcpu: : pointer to domain object, or NULL for Domain0 : virtual CPU number : pointer to a bit map of real CPUs (in 8-bit bytes) (IN) Each bit set to 1 means that corresponding CPU is usable. Bytes are stored in little-endian order: CPU0-7, 8-15... In each byte, lowest CPU number is least significant bit. : number of bytes in cpumap, from 1 up to size of CPU map in underlying virtualization system (Xen...). If maplen < size, missing bytes are set to zero. If maplen > size, failure code is returned.

Dynamically change the real CPUs which can be allocated to a virtual CPU. This function may require privileged access to the hypervisor.

This command only changes the runtime configuration of the domain, so can only be called on an active domain.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.164 int virDomainPinVcpuFlags ( virDomainPtr domain, unsigned int vcpu, unsigned char \* cpumap, int maplen, unsigned int flags )

virDomainPinVcpuFlags: : pointer to domain object, or NULL for Domain0 : virtual CPU number : pointer to a bit map of real CPUs (in 8-bit bytes) (IN) Each bit set to 1 means that corresponding CPU is usable. Bytes are stored in little-endian order: CPU0-7, 8-15... In each byte, lowest CPU number is least significant bit. : number of bytes in cpumap, from 1 up to size of CPU map in underlying virtualization system (Xen...). If maplen < size, missing bytes are set to zero. If maplen > size, failure code is returned. : bitwise-OR of virDomainModificationImpact

Dynamically change the real CPUs which can be allocated to a virtual CPU. This function may require privileged access to the hypervisor.

may include VIR\_DOMAIN\_AFFECT\_LIVE or VIR\_DOMAIN\_AFFECT\_CONFIG. Both flags may be set. If VIR\_DOMAIN\_AFFECT\_LIVE is set, the change affects a running domain and may fail if domain is not alive. If VIR\_D-

OMAIN\_AFFECT\_CONFIG is set, the change affects persistent state, and will fail for transient domains. If neither flag is specified (that is, is VIR\_DOMAIN\_AFFECT\_CURRENT), then an inactive domain modifies persistent setup, while an active domain is hypervisor-dependent on whether just live or both live and persistent state is changed. Not all hypervisors can support all flag combinations.

See also virDomainGetVcpuPinInfo for querying this information.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.165** int virDomainPMSuspendForDuration ( virDomainPtr dom, unsigned int target, unsigned long long duration, unsigned int flags )

virDomainPMSuspendForDuration: : a domain object : a value from virNodeSuspendTarget : duration in seconds to suspend, or 0 for indefinite : extra flags; not used yet, so callers should always pass 0

Attempt to have the guest enter the given power management suspension level. If is non-zero, also schedule the guest to resume normal operation after that many seconds, if nothing else has resumed it earlier. Some hypervisors require that be 0, for an indefinite suspension.

Dependent on hypervisor used, this may require a guest agent to be available, e.g. QEMU.

Returns: 0 on success, -1 on failure.

**4.7.2.166** int virDomainPMWakeup ( virDomainPtr dom, unsigned int flags )

virDomainPMWakeup: : a domain object : extra flags; not used yet, so callers should always pass 0

Inject a wakeup into the guest that previously used virDomainPMSuspendForDuration, rather than waiting for the previously requested duration (if any) to elapse.

Returns: 0 on success, -1 on failure.

**4.7.2.167** int virDomainReboot ( virDomainPtr domain, unsigned int flags )

virDomainReboot: : a domain object : bitwise-OR of virDomainRebootFlagValues

Reboot a domain, the domain object is still usable there after but the domain OS is being stopped for a restart. Note that the guest OS may ignore the request.

If is set to zero, then the hypervisor will choose the method of shutdown it considers best. To have greater control pass exactly one of the virDomainRebootFlagValues.

To use guest agent (VIR\_DOMAIN\_REBOOT\_GUEST\_AGENT) the domain XML must have <channel> configured.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.168** int virDomainRef ( virDomainPtr domain )

virDomainRef: : the domain to hold a reference on

Increment the reference count on the domain. For each additional call to this method, there shall be a corresponding call to virDomainFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a domain would increment the reference count.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.169** int virDomainReset ( virDomainPtr domain, unsigned int flags )

virDomainReset: : a domain object : extra flags; not used yet, so callers should always pass 0

Reset a domain immediately without any guest OS shutdown. Reset emulates the power reset button on a machine, where all hardware sees the RST line set and reinitializes internal state.

Note that there is a risk of data loss caused by reset without any guest OS shutdown.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.170** int virDomainRestore ( virConnectPtr conn, const char \* from )

virDomainRestore: : pointer to the hypervisor connection : path to the input file

This method will restore a domain saved to disk by [virDomainSave\(\)](#).

See [virDomainRestoreFlags\(\)](#) for more control.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.171** int virDomainRestoreFlags ( virConnectPtr conn, const char \* from, const char \* dxml, unsigned int flags )

virDomainRestoreFlags: : pointer to the hypervisor connection : path to the input file : (optional) XML config for adjusting guest xml used on restore : bitwise-OR of virDomainSaveRestoreFlags

This method will restore a domain saved to disk by [virDomainSave\(\)](#).

If the hypervisor supports it, can be used to alter host-specific portions of the domain XML that will be used when restoring an image. For example, it is possible to alter the backing filename that is associated with a disk device, in order to prepare for file renaming done as part of backing up the disk device while the domain is stopped.

If includes VIR\_DOMAIN\_SAVE\_BYPASS\_CACHE, then libvirt will attempt to bypass the file system cache while restoring the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing saves to NFS.

Normally, the saved state file will remember whether the domain was running or paused, and restore defaults to the same state. Specifying VIR\_DOMAIN\_SAVE\_RUNNING or VIR\_DOMAIN\_SAVE\_PAUSED in will override the default read from the file. These two flags are mutually exclusive.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.172** int virDomainResume ( virDomainPtr domain )

virDomainResume: : a domain object

Resume a suspended domain, the process is restarted from the state where it was frozen by calling [virDomainSuspend\(\)](#). This function may require privileged access. Moreover, resume may not be supported if domain is in some special state like VIR\_DOMAIN\_PMSUSPENDED.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.173** int virDomainRevertToSnapshot ( virDomainSnapshotPtr snapshot, unsigned int flags )

virDomainRevertToSnapshot: : a domain snapshot object : bitwise-OR of virDomainSnapshotRevertFlags

Revert the domain to a given snapshot.

Normally, the domain will revert to the same state the domain was in while the snapshot was taken (whether inactive, running, or paused), except that disk snapshots default to reverting to inactive state. Including VIR\_DOMAIN\_SNAPSHOT\_REVERT\_RUNNING in overrides the snapshot state to guarantee a running domain after the revert; or including VIR\_DOMAIN\_SNAPSHOT\_REVERT\_PAUSED in guarantees a paused domain after the revert. These

two flags are mutually exclusive. While a persistent domain does not need either flag, it is not possible to revert a transient domain into an inactive state, so transient domains require the use of one of these two flags.

Reverting to any snapshot discards all configuration changes made since the last snapshot. Additionally, reverting to a snapshot from a running domain is a form of data loss, since it discards whatever is in the guest's RAM at the time. Since the very nature of keeping snapshots implies the intent to roll back state, no additional confirmation is normally required for these lossy effects.

However, there are two particular situations where reverting will be refused by default, and where must include `VIR_DOMAIN_SNAPSHOT_REVERT_FORCE` to acknowledge the risks. 1) Any attempt to revert to a snapshot that lacks the metadata to perform ABI compatibility checks (generally the case for snapshots that lack a full <domain> when listed by [virDomainSnapshotGetXMLDesc\(\)](#), such as those created prior to libvirt 0.9.5). 2) Any attempt to revert a running domain to an active state that requires starting a new hypervisor instance rather than reusing the existing hypervisor (since this would terminate all connections to the domain, such as VNC or Spice graphics) - this condition arises from active snapshots that are provably ABI incompatible, as well as from inactive snapshots with a request to start the domain after the revert.

Returns 0 if the creation is successful, -1 on error.

#### 4.7.2.174 `int virDomainSave ( virDomainPtr domain, const char * to )`

`virDomainSave`: : a domain object : path for the output file

This method will suspend a domain and save its memory contents to a file on disk. After the call, if successful, the domain is not listed as running anymore (this ends the life of a transient domain). Use [virDomainRestore\(\)](#) to restore a domain after saving.

See [virDomainSaveFlags\(\)](#) for more control. Also, a save file can be inspected or modified slightly with [virDomainSaveImageGetXMLDesc\(\)](#) and [virDomainSaveImageDefineXML\(\)](#).

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.175 `int virDomainSaveFlags ( virDomainPtr domain, const char * to, const char * dxml, unsigned int flags )`

`virDomainSaveFlags`: : a domain object : path for the output file : (optional) XML config for adjusting guest xml used on restore : bitwise-OR of `virDomainSaveRestoreFlags`

This method will suspend a domain and save its memory contents to a file on disk. After the call, if successful, the domain is not listed as running anymore (this ends the life of a transient domain). Use [virDomainRestore\(\)](#) to restore a domain after saving.

If the hypervisor supports it, can be used to alter host-specific portions of the domain XML that will be used when restoring an image. For example, it is possible to alter the backing filename that is associated with a disk device, in order to prepare for file renaming done as part of backing up the disk device while the domain is stopped.

If includes `VIR_DOMAIN_SAVE_BYPASS_CACHE`, then libvirt will attempt to bypass the file system cache while creating the file, or fail if it cannot do so for the given system; this can allow less pressure on file system cache, but also risks slowing saves to NFS.

Normally, the saved state file will remember whether the domain was running or paused, and restore defaults to the same state. Specifying `VIR_DOMAIN_SAVE_RUNNING` or `VIR_DOMAIN_SAVE_PAUSED` in will override what state gets saved into the file. These two flags are mutually exclusive.

A save file can be inspected or modified slightly with [virDomainSaveImageGetXMLDesc\(\)](#) and [virDomainSaveImageDefineXML\(\)](#).

Some hypervisors may prevent this operation if there is a current block copy operation; in that case, use [virDomainBlockJobAbort\(\)](#) to stop the block copy first.

Returns 0 in case of success and -1 in case of failure.

4.7.2.176 `int virDomainSavImageDefineXML ( virConnectPtr conn, const char * file, const char * dxml, unsigned int flags )`

4.7.2.177 `char* virDomainSavImageGetXMLDesc ( virConnectPtr conn, const char * file, unsigned int flags )`

4.7.2.178 `char* virDomainScreenshot ( virDomainPtr domain, virStreamPtr stream, unsigned int screen, unsigned int flags )`

virDomainScreenshot: : a domain object : stream to use as output : monitor ID to take screenshot from : extra flags; not used yet, so callers should always pass 0

Take a screenshot of current domain console as a stream. The image format is hypervisor specific. Moreover, some hypervisors supports multiple displays per domain. These can be distinguished by argument.

This call sets up a stream; subsequent use of stream API is necessary to transfer actual data, determine how much data is successfully transfered, and detect any errors.

The screen ID is the sequential number of screen. In case of multiple graphics cards, heads are enumerated before devices, e.g. having two graphics cards, both with four heads, screen ID 5 addresses the second head on the second card.

Returns a string representing the mime-type of the image format, or NULL upon error. The caller must free() the returned value.

4.7.2.179 `int virDomainSendKey ( virDomainPtr domain, unsigned int codeset, unsigned int holdtime, unsigned int * keycodes, int nkeycodes, unsigned int flags )`

virDomainSendKey: : pointer to domain object, or NULL for Domain0 : the code set of keycodes, from virKeycode-Set : the duration (in milliseconds) that the keys will be held : array of keycodes : number of keycodes, up to VIR\_DOMAIN\_SEND\_KEY\_MAX\_KEYS : extra flags; not used yet, so callers should always pass 0

Send key(s) to the guest.

Returns 0 in case of success, -1 in case of failure.

4.7.2.180 `int virDomainSetAutostart ( virDomainPtr domain, int autostart )`

virDomainSetAutostart: : a domain object : whether the domain should be automatically started 0 or 1

Configure the domain to be automatically started when the host machine boots.

Returns -1 in case of error, 0 in case of success

4.7.2.181 `int virDomainSetBlkioParameters ( virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags )`

virDomainSetBlkioParameters: : pointer to domain object : pointer to blkio parameter objects : number of blkio parameters (this value can be the same or less than the number of parameters supported) : bitwise-OR of virDomainModificationImpact

Change all or a subset of the blkio tunables. This function may require privileged access to the hypervisor.

Returns -1 in case of error, 0 in case of success.

4.7.2.182 `int virDomainSetBlockIoTune ( virDomainPtr dom, const char * disk, virTypedParameterPtr params, int nparams, unsigned int flags )`

virDomainSetBlockIoTune: : pointer to domain object : path to the block device, or device shorthand : Pointer to blkio parameter objects : Number of blkio parameters (this value can be the same or less than the number of parameters supported) : bitwise-OR of virDomainModificationImpact

Change all or a subset of the per-device block IO tunables.

The parameter is either an unambiguous source name of the block device (the `<source file=...">` sub-element, such as `/path/to/image`), or the device target shorthand (the `<target dev=...">` sub-element, such as `xvda`). Valid names can be found by calling [virDomainGetXMLDesc\(\)](#) and inspecting elements within `//domain/devices/disk`.

Returns -1 in case of error, 0 in case of success.

**4.7.2.183** `int virDomainSetInterfaceParameters ( virDomainPtr domain, const char * device, virTypedParameterPtr params, int nparams, unsigned int flags )`

`virDomainSetInterfaceParameters`: : pointer to domain object : the interface name or mac address : pointer to interface parameter objects : number of interface parameter (this value can be the same or less than the number of parameters supported) : bitwise-OR of `virDomainModificationImpact`

Change a subset or all parameters of interface; currently this includes bandwidth parameters. The value of should be either `VIR_DOMAIN_AFFECT_CURRENT`, or a bitwise-or of values `VIR_DOMAIN_AFFECT_LIVE` and `VIR_DOMAIN_AFFECT_CONFIG`, although hypervisors vary in which flags are supported.

This function may require privileged access to the hypervisor.

Returns -1 in case of error, 0 in case of success.

**4.7.2.184** `int virDomainSetMaxMemory ( virDomainPtr domain, unsigned long memory )`

`virDomainSetMaxMemory`: : a domain object or NULL : the memory size in kibibytes (blocks of 1024 bytes)

Dynamically change the maximum amount of physical memory allocated to a domain. If domain is NULL, then this change the amount of memory reserved to Domain0 i.e. the domain where the application runs. This function may require privileged access to the hypervisor.

This command is hypervisor-specific for whether active, persistent, or both configurations are changed; for more control, use [virDomainSetMemoryFlags\(\)](#).

Returns 0 in case of success and -1 in case of failure.

**4.7.2.185** `int virDomainSetMemory ( virDomainPtr domain, unsigned long memory )`

`virDomainSetMemory`: : a domain object or NULL : the memory size in kibibytes (blocks of 1024 bytes)

Dynamically change the target amount of physical memory allocated to a domain. If domain is NULL, then this change the amount of memory reserved to Domain0 i.e. the domain where the application runs. This function may require privileged access to the hypervisor.

This command only changes the runtime configuration of the domain, so can only be called on an active domain.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.186** `int virDomainSetMemoryFlags ( virDomainPtr domain, unsigned long memory, unsigned int flags )`

`virDomainSetMemoryFlags`: : a domain object or NULL : the memory size in kibibytes (blocks of 1024 bytes) : bitwise-OR of `virDomainMemoryModFlags`

Dynamically change the target amount of physical memory allocated to a domain. If domain is NULL, then this change the amount of memory reserved to Domain0 i.e. the domain where the application runs. This function may require privileged access to the hypervisor.

may include `VIR_DOMAIN_AFFECT_LIVE` or `VIR_DOMAIN_AFFECT_CONFIG`. Both flags may be set. If `VIR_DOMAIN_AFFECT_LIVE` is set, the change affects a running domain and will fail if domain is not active. If `VIR_DOMAIN_AFFECT_CONFIG` is set, the change affects persistent state, and will fail for transient domains. If neither flag is specified (that is, is `VIR_DOMAIN_AFFECT_CURRENT`), then an inactive domain modifies persistent setup, while an active domain is hypervisor-dependent on whether just live or both live and persistent state is changed.



If VIR\_DOMAIN\_MEM\_MAXIMUM is set, the change affects domain's maximum memory size rather than current memory size. Not all hypervisors can support all flag combinations.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.187** int virDomainSetMemoryParameters ( virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags )

virDomainSetMemoryParameters: : pointer to domain object : pointer to memory parameter objects : number of memory parameter (this value can be the same or less than the number of parameters supported) : bitwise-OR of virDomainModificationImpact

Change all or a subset of the memory tunables. This function may require privileged access to the hypervisor.

Returns -1 in case of error, 0 in case of success.

**4.7.2.188** int virDomainSetMetadata ( virDomainPtr domain, int type, const char \* metadata, const char \* key, const char \* uri, unsigned int flags )

virDomainSetMetadata: : a domain object : type of description, from virDomainMetadataType : new metadata text : XML namespace key, or NULL : XML namespace URI, or NULL : bitwise-OR of virDomainModificationImpact

Sets the appropriate domain element given by to the value of . A of VIR\_DOMAIN\_METADATA\_DESCRIPTION is free-form text; VIR\_DOMAIN\_METADATA\_TITLE is free-form, but no newlines are permitted, and should be short (although the length is not enforced). For these two options and are irrelevant and must be set to NULL.

For type VIR\_DOMAIN\_METADATA\_ELEMENT must be well-formed XML belonging to namespace defined by with local name .

Passing NULL for says to remove that element from the domain XML (passing the empty string leaves the element present).

The resulting metadata will be present in [virDomainGetXMLDesc\(\)](#), as well as quick access through [virDomainGetMetadata\(\)](#).

controls whether the live domain, persistent configuration, or both will be modified.

Returns 0 on success, -1 in case of failure.

**4.7.2.189** int virDomainSetNumaParameters ( virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags )

virDomainSetNumaParameters: : pointer to domain object : pointer to numa parameter objects : number of numa parameters (this value can be the same or less than the number of parameters supported) : bitwise-OR of virDomainModificationImpact

Change all or a subset of the numa tunables. This function may require privileged access to the hypervisor.

Returns -1 in case of error, 0 in case of success.

**4.7.2.190** int virDomainSetSchedulerParameters ( virDomainPtr domain, virTypedParameterPtr params, int nparams )

virDomainSetSchedulerParameters: : pointer to domain object : pointer to scheduler parameter objects : number of scheduler parameter objects (this value can be the same or less than the returned value nparams of virDomainGetSchedulerType)

Change all or a subset of the scheduler parameters. It is hypervisor-specific whether this sets live, persistent, or both settings; for more control, use virDomainSetSchedulerParametersFlags.

Returns -1 in case of error, 0 in case of success.

**4.7.2.191** `int virDomainSetSchedulerParametersFlags ( virDomainPtr domain, virTypedParameterPtr params, int nparams, unsigned int flags )`

`virDomainSetSchedulerParametersFlags`: : pointer to domain object : pointer to scheduler parameter objects : number of scheduler parameter objects (this value can be the same or less than the returned value `nparams` of `virDomainGetSchedulerType`) : bitwise-OR of `virDomainModificationImpact`

Change a subset or all scheduler parameters. The value of should be either `VIR_DOMAIN_AFFECT_CURRENT`, or a bitwise-or of values from `VIR_DOMAIN_AFFECT_LIVE` and `VIR_DOMAIN_AFFECT_CURRENT`, although hypervisors vary in which flags are supported.

Returns -1 in case of error, 0 in case of success.

**4.7.2.192** `int virDomainSetVcpus ( virDomainPtr domain, unsigned int nvcpus )`

`virDomainSetVcpus`: : pointer to domain object, or NULL for Domain0 : the new number of virtual CPUs for this domain

Dynamically change the number of virtual CPUs used by the domain. Note that this call may fail if the underlying virtualization hypervisor does not support it or if growing the number is arbitrary limited. This function may require privileged access to the hypervisor.

This command only changes the runtime configuration of the domain, so can only be called on an active domain. It is hypervisor-dependent whether it also affects persistent configuration; for more control, use [virDomainSetVcpusFlags\(\)](#).

Returns 0 in case of success, -1 in case of failure.

**4.7.2.193** `int virDomainSetVcpusFlags ( virDomainPtr domain, unsigned int nvcpus, unsigned int flags )`

`virDomainSetVcpusFlags`: : pointer to domain object, or NULL for Domain0 : the new number of virtual CPUs for this domain, must be at least 1 : bitwise-OR of `virDomainVcpuFlags`

Dynamically change the number of virtual CPUs used by the domain. Note that this call may fail if the underlying virtualization hypervisor does not support it or if growing the number is arbitrary limited. This function may require privileged access to the hypervisor.

may include `VIR_DOMAIN_AFFECT_LIVE` to affect a running domain (which may fail if domain is not active), or `VIR_DOMAIN_AFFECT_CONFIG` to affect the next boot via the XML description of the domain. Both flags may be set. If neither flag is specified (that is, is `VIR_DOMAIN_AFFECT_CURRENT`), then an inactive domain modifies persistent setup, while an active domain is hypervisor-dependent on whether just live or both live and persistent state is changed.

If includes `VIR_DOMAIN_VCPU_MAXIMUM`, then `VIR_DOMAIN_AFFECT_LIVE` must be clear, and only the maximum virtual CPU limit is altered; generally, this value must be less than or equal to [virConnectGetMaxVcpus\(\)](#). Otherwise, this call affects the current virtual CPU limit, which must be less than or equal to the maximum limit. Not all hypervisors can support all flag combinations.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.194** `int virDomainShutdown ( virDomainPtr domain )`

`virDomainShutdown`: : a domain object

Shutdown a domain, the domain object is still usable thereafter but the domain OS is being stopped. Note that the guest OS may ignore the request. For guests that react to a shutdown request, the differences from [virDomainDestroy\(\)](#) are that the guests disk storage will be in a stable state rather than having the (virtual) power cord pulled, and this command returns as soon as the shutdown request is issued rather than blocking until the guest is no longer running.

If the domain is transient and has any snapshot metadata (see [virDomainSnapshotNum\(\)](#)), then that metadata will automatically be deleted when the domain quits.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.195 `int virDomainShutdownFlags ( virDomainPtr domain, unsigned int flags )`

`virDomainShutdownFlags`: : a domain object : bitwise-OR of `virDomainShutdownFlagValues`

Shutdown a domain, the domain object is still usable thereafter but the domain OS is being stopped. Note that the guest OS may ignore the request. For guests that react to a shutdown request, the differences from `virDomainDestroy()` are that the guest's disk storage will be in a stable state rather than having the (virtual) power cord pulled, and this command returns as soon as the shutdown request is issued rather than blocking until the guest is no longer running.

If the domain is transient and has any snapshot metadata (see `virDomainSnapshotNum()`), then that metadata will automatically be deleted when the domain quits.

If is set to zero, then the hypervisor will choose the method of shutdown it considers best. To have greater control pass exactly one of the `virDomainShutdownFlagValues`.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.196 `virDomainSnapshotPtr virDomainSnapshotCreateXML ( virDomainPtr domain, const char * xmlDesc, unsigned int flags )`

`virDomainSnapshotCreateXML`: : a domain object : string containing an XML description of the domain : bitwise-OR of `virDomainSnapshotCreateFlags`

Creates a new snapshot of a domain based on the snapshot xml contained in `xmlDesc`.

If is 0, the domain can be active, in which case the snapshot will be a system checkpoint (both disk state and runtime VM state such as RAM contents), where reverting to the snapshot is the same as resuming from hibernation (TCP connections may have timed out, but everything else picks up where it left off); or the domain can be inactive, in which case the snapshot includes just the disk state prior to booting. The newly created snapshot becomes current (see `virDomainSnapshotCurrent()`), and is a child of any previous current snapshot.

If includes `VIR_DOMAIN_SNAPSHOT_CREATE_REDEFINE`, then this is a request to reinstate snapshot metadata that was previously discarded, rather than creating a new snapshot. This can be used to recreate a snapshot hierarchy on a destination, then remove it on the source, in order to allow migration (since migration normally fails if snapshot metadata still remains on the source machine). When redefining snapshot metadata, the current snapshot will not be altered unless the `VIR_DOMAIN_SNAPSHOT_CREATE_CURRENT` flag is also present. It is an error to request the `VIR_DOMAIN_SNAPSHOT_CREATE_CURRENT` flag without `VIR_DOMAIN_SNAPSHOT_CREATE_REDEFINE`. On some hypervisors, redefining an existing snapshot can be used to alter host-specific portions of the domain XML to be used during revert (such as backing filenames associated with disk devices), but must not alter guest-visible layout. When redefining a snapshot name that does not exist, the hypervisor may validate that reverting to the snapshot appears to be possible (for example, disk images have snapshot contents by the requested name). Not all hypervisors support these flags.

If includes `VIR_DOMAIN_SNAPSHOT_CREATE_NO_METADATA`, then the domain's disk images are modified according to , but then the just-created snapshot has its metadata deleted. This flag is incompatible with `VIR_DOMAIN_SNAPSHOT_CREATE_REDEFINE`.

If includes `VIR_DOMAIN_SNAPSHOT_CREATE_HALT`, then the domain will be inactive after the snapshot completes, regardless of whether it was active before; otherwise, a running domain will still be running after the snapshot. This flag is invalid on transient domains, and is incompatible with `VIR_DOMAIN_SNAPSHOT_CREATE_REDEFINE`.

If includes `VIR_DOMAIN_SNAPSHOT_CREATE_DISK_ONLY`, then the snapshot will be limited to the disks described in , and no VM state will be saved. For an active guest, the disk image may be inconsistent (as if power had been pulled), and specifying this with the `VIR_DOMAIN_SNAPSHOT_CREATE_HALT` flag risks data loss.

If includes `VIR_DOMAIN_SNAPSHOT_CREATE_QUIESCE`, then the libvirt will attempt to use guest agent to freeze and thaw all file systems in use within domain OS. However, if the guest agent is not present, an error is thrown. Moreover, this flag requires `VIR_DOMAIN_SNAPSHOT_CREATE_DISK_ONLY` to be passed as well.

By default, if the snapshot involves external files, and any of the destination files already exist as a non-empty regular file, the snapshot is rejected to avoid losing contents of those files. However, if includes `VIR_DOMAIN_SNAPSHOT_CREATE_REUSE_EXT`, then the destination files must already exist and contain content identical to the source files (this allows a management app to pre-create files with relative backing file names, rather than the default of creating with absolute backing file names).

Be aware that although libvirt prefers to report errors up front with no other effect, some hypervisors have certain types of failures where the overall command can easily fail even though the guest configuration was partially altered (for example, if a disk snapshot request for two disks fails on the second disk, but the first disk alteration cannot be rolled back). If this API call fails, it is therefore normally necessary to follow up with `virDomainGetXMLDesc()` and check each disk to determine if any partial changes occurred. However, if contains `VIR_DOMAIN_SNAPSHOT_CREATE_ATOMIC`, then libvirt guarantees that this command will not alter any disks unless the entire set of changes can be done atomically, making failure recovery simpler (note that it is still possible to fail after disks have changed, but only in the much rarer cases of running out of memory or disk space).

Some hypervisors may prevent this operation if there is a current block copy operation; in that case, use `virDomainBlockJobAbort()` to stop the block copy first.

Returns an (opaque) `virDomainSnapshotPtr` on success, `NULL` on failure.

#### 4.7.2.197 `virDomainSnapshotPtr virDomainSnapshotCurrent ( virDomainPtr domain, unsigned int flags )`

`virDomainSnapshotCurrent`: : a domain object : extra flags; not used yet, so callers should always pass 0

Get the current snapshot for a domain, if any.

Returns a domain snapshot object or `NULL` in case of failure. If the current domain snapshot cannot be found, then the `VIR_ERR_NO_DOMAIN_SNAPSHOT` error is raised.

#### 4.7.2.198 `int virDomainSnapshotDelete ( virDomainSnapshotPtr snapshot, unsigned int flags )`

`virDomainSnapshotDelete`: : a domain snapshot object : bitwise-OR of supported `virDomainSnapshotDeleteFlags`

Delete the snapshot.

If is 0, then just this snapshot is deleted, and changes from this snapshot are automatically merged into children snapshots. If includes `VIR_DOMAIN_SNAPSHOT_DELETE_CHILDREN`, then this snapshot and any descendant snapshots are deleted. If includes `VIR_DOMAIN_SNAPSHOT_DELETE_CHILDREN_ONLY`, then any descendant snapshots are deleted, but this snapshot remains. These two flags are mutually exclusive.

If includes `VIR_DOMAIN_SNAPSHOT_DELETE_METADATA_ONLY`, then any snapshot metadata tracked by libvirt is removed while keeping the snapshot contents intact; if a hypervisor does not require any libvirt metadata to track snapshots, then this flag is silently ignored.

Returns 0 if the selected snapshot(s) were successfully deleted, -1 on error.

#### 4.7.2.199 `int virDomainSnapshotFree ( virDomainSnapshotPtr snapshot )`

`virDomainSnapshotFree`: : a domain snapshot object

Free the domain snapshot object. The snapshot itself is not modified. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.200 `virConnectPtr virDomainSnapshotGetConnect ( virDomainSnapshotPtr snapshot )`

`virDomainSnapshotGetConnect`: : a snapshot object

Get the connection that owns the domain that a snapshot was created for

Returns the connection or `NULL`.

**4.7.2.201 virDomainPtr virDomainSnapshotGetDomain ( virDomainSnapshotPtr snapshot )**

virDomainSnapshotGetDomain: : a snapshot object

Get the domain that a snapshot was created for

Returns the domain or NULL.

**4.7.2.202 const char\* virDomainSnapshotGetName ( virDomainSnapshotPtr snapshot )**

virDomainSnapshotGetName: : a snapshot object

Get the public name for that snapshot

Returns a pointer to the name or NULL, the string need not be deallocated as its lifetime will be the same as the snapshot object.

**4.7.2.203 virDomainSnapshotPtr virDomainSnapshotGetParent ( virDomainSnapshotPtr snapshot, unsigned int flags )**

virDomainSnapshotGetParent: : a snapshot object : extra flags; not used yet, so callers should always pass 0

Get the parent snapshot for , if any.

Returns a domain snapshot object or NULL in case of failure. If the given snapshot is a root (no parent), then the VIR\_ERR\_NO\_DOMAIN\_SNAPSHOT error is raised.

**4.7.2.204 char\* virDomainSnapshotGetXMLDesc ( virDomainSnapshotPtr snapshot, unsigned int flags )**

virDomainSnapshotGetXMLDesc: : a domain snapshot object : bitwise-OR of subset of virDomainXMLFlags

Provide an XML description of the domain snapshot.

No security-sensitive data will be included unless contains VIR\_DOMAIN\_XML\_SECURE; this flag is rejected on read-only connections. For this API, should not contain either VIR\_DOMAIN\_XML\_INACTIVE or VIR\_DOMAIN\_XML\_UPDATE\_CPU.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

**4.7.2.205 int virDomainSnapshotHasMetadata ( virDomainSnapshotPtr snapshot, unsigned int flags )**

virDomainSnapshotHasMetadata: : a snapshot object : extra flags; not used yet, so callers should always pass 0

Determine if the given snapshot is associated with libvirt metadata that would prevent the deletion of the domain.

Returns 1 if the snapshot has metadata, 0 if the snapshot exists without help from libvirt, or -1 on error.

**4.7.2.206 int virDomainSnapshotIsCurrent ( virDomainSnapshotPtr snapshot, unsigned int flags )**

virDomainSnapshotIsCurrent: : a snapshot object : extra flags; not used yet, so callers should always pass 0

Determine if the given snapshot is the domain's current snapshot. See also [virDomainHasCurrentSnapshot\(\)](#).

Returns 1 if current, 0 if not current, or -1 on error.

**4.7.2.207 int virDomainSnapshotListAllChildren ( virDomainSnapshotPtr snapshot, virDomainSnapshotPtr \*\* snaps, unsigned int flags )**

virDomainSnapshotListAllChildren: : a domain snapshot object : pointer to variable to store the array containing snapshot objects, or NULL if the list is not required (just returns number of snapshots) : bitwise-OR of supported

### virDomainSnapshotListFlags

Collect the list of domain snapshots that are children of the given snapshot, and allocate an array to store those objects. This API solves the race inherent in [virDomainSnapshotListChildrenNames\(\)](#).

By default, this command covers only direct children; it is also possible to expand things to cover all descendants, when includes `VIR_DOMAIN_SNAPSHOT_LIST_DESCENDANTS`. Also, some filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virDomainSnapshotFree\(\)](#) on each array element, then calling `free()` on .

**4.7.2.208** `int virDomainSnapshotListChildrenNames ( virDomainSnapshotPtr snapshot, char ** names, int nameslen, unsigned int flags )`

`virDomainSnapshotListChildrenNames`: : a domain snapshot object : array to collect the list of names of snapshots  
: size of : bitwise-OR of supported `virDomainSnapshotListFlags`

Collect the list of domain snapshots that are children of the given snapshot, and store their names in . The value to use for can be determined by [virDomainSnapshotNumChildren\(\)](#) with the same .

By default, this command covers only direct children; it is also possible to expand things to cover all descendants, when includes `VIR_DOMAIN_SNAPSHOT_LIST_DESCENDANTS`. Also, some filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 in case of error. Note that this command is inherently racy: another connection can define a new snapshot between a call to [virDomainSnapshotNumChildren\(\)](#) and this call. You are only guaranteed that all currently defined snapshots were listed if the return is less than . Likewise, you should be prepared for [virDomainSnapshotLookupByName\(\)](#) to fail when converting a name from this call into a snapshot object, if another connection deletes the snapshot in the meantime. For more control over the results, see [virDomainSnapshotListAllChildren\(\)](#).

Returns the number of domain snapshots found or -1 in case of error. The caller is responsible for freeing each member of the array.

**4.7.2.209** `int virDomainSnapshotListNames ( virDomainPtr domain, char ** names, int nameslen, unsigned int flags )`

`virDomainSnapshotListNames`: : a domain object : array to collect the list of names of snapshots : size of : bitwise-OR of supported `virDomainSnapshotListFlags`



Collect the list of domain snapshots for the given domain, and store their names in `names`. The value to use for `names` can be determined by `virDomainSnapshotNum()` with the same `domain`.

By default, this command covers all snapshots; it is also possible to limit things to just snapshots with no parents, when includes `VIR_DOMAIN_SNAPSHOT_LIST_ROOTS`. Additional filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Note that this command is inherently racy: another connection can define a new snapshot between a call to `virDomainSnapshotNum()` and this call. You are only guaranteed that all currently defined snapshots were listed if the return is less than `0`. Likewise, you should be prepared for `virDomainSnapshotLookupByName()` to fail when converting a name from this call into a snapshot object, if another connection deletes the snapshot in the meantime. For more control over the results, see `virDomainListAllSnapshots()`.

Returns the number of domain snapshots found or `-1` in case of error. The caller is responsible for freeing each member of the array.

**4.7.2.210** `virDomainSnapshotPtr virDomainSnapshotLookupByName ( virDomainPtr domain, const char * name, unsigned int flags )`

**4.7.2.211** `int virDomainSnapshotNum ( virDomainPtr domain, unsigned int flags )`

`virDomainSnapshotNum`: : a domain object : bitwise-OR of supported `virDomainSnapshotListFlags`

Provides the number of domain snapshots for this domain.

By default, this command covers all snapshots; it is also possible to limit things to just snapshots with no parents, when includes `VIR_DOMAIN_SNAPSHOT_LIST_ROOTS`. Additional filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or `-1` in case of error.

**4.7.2.212** `int virDomainSnapshotNumChildren ( virDomainSnapshotPtr snapshot, unsigned int flags )`

`virDomainSnapshotNumChildren`: : a domain snapshot object : bitwise-OR of supported `virDomainSnapshotListFlags`

Provides the number of child snapshots for this domain snapshot.

By default, this command covers only direct children; it is also possible to expand things to cover all descendants, when includes `VIR_DOMAIN_SNAPSHOT_LIST_DESCENDANTS`. Also, some filters are provided in groups, where each group contains bits that describe mutually exclusive attributes of a snapshot, and where all bits within

a group describe all possible snapshots. Some hypervisors might reject explicit bits from a group where the hypervisor cannot make a distinction. For a group supported by a given hypervisor, the behavior when no bits of a group are set is identical to the behavior when all bits in that group are set. When setting bits from more than one group, it is possible to select an impossible combination, in that case a hypervisor may return either 0 or an error.

The first group of is `VIR_DOMAIN_SNAPSHOT_LIST_LEAVES` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_LEAVES`, to filter based on snapshots that have no further children (a leaf snapshot).

The next group of is `VIR_DOMAIN_SNAPSHOT_LIST_METADATA` and `VIR_DOMAIN_SNAPSHOT_LIST_NO_METADATA`, for filtering snapshots based on whether they have metadata that would prevent the removal of the last reference to a domain.

Returns the number of domain snapshots found or -1 in case of error.

#### 4.7.2.213 `int virDomainSnapshotRef ( virDomainSnapshotPtr snapshot )`

`virDomainSnapshotRef`: : the snapshot to hold a reference on

Increment the reference count on the snapshot. For each additional call to this method, there shall be a corresponding call to `virDomainSnapshotFree` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection and domain remain open until all threads have finished using the snapshot. ie, each new thread using a snapshot would increment the reference count.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.214 `int virDomainSuspend ( virDomainPtr domain )`

`virDomainSuspend`: : a domain object

Suspends an active domain, the process is frozen without further access to CPU resources and I/O but the memory used by the domain at the hypervisor level will stay allocated. Use [virDomainResume\(\)](#) to reactivate the domain. This function may require privileged access. Moreover, suspend may not be supported if domain is in some special state like `VIR_DOMAIN_PMSUSPENDED`.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.215 `int virDomainUndefine ( virDomainPtr domain )`

`virDomainUndefine`: : pointer to a defined domain

Undefine a domain. If the domain is running, it's converted to transient domain, without stopping it. If the domain is inactive, the domain configuration is removed.

If the domain has a managed save image (see [virDomainHasManagedSaveImage\(\)](#)), or if it is inactive and has any snapshot metadata (see [virDomainSnapshotNum\(\)](#)), then the undefine will fail. See [virDomainUndefineFlags\(\)](#) for more control.

Returns 0 in case of success, -1 in case of error

#### 4.7.2.216 `int virDomainUndefineFlags ( virDomainPtr domain, unsigned int flags )`

`virDomainUndefineFlags`: : pointer to a defined domain : bitwise-OR of supported `virDomainUndefineFlagsValues`

Undefine a domain. If the domain is running, it's converted to transient domain, without stopping it. If the domain is inactive, the domain configuration is removed.

If the domain has a managed save image (see [virDomainHasManagedSaveImage\(\)](#)), then including `VIR_DOMAIN_UNDEFINE_MANAGED_SAVE` in will also remove that file, and omitting the flag will cause the undefine process to fail.



If the domain is inactive and has any snapshot metadata (see [virDomainSnapshotNum\(\)](#)), then including `VIR_DOMAIN_UNDEFINE_SNAPSHOTS_METADATA` will also remove that metadata. Omitting the flag will cause the undefine of an inactive domain to fail. Active snapshots will retain snapshot metadata until the (now-transient) domain halts, regardless of whether this flag is present. On hypervisors where snapshots do not use libvirt metadata, this flag has no effect.

Returns 0 in case of success, -1 in case of error

#### 4.7.2.217 `int virDomainUpdateDeviceFlags ( virDomainPtr domain, const char * xml, unsigned int flags )`

`virDomainUpdateDeviceFlags`: : pointer to domain object : pointer to XML description of one device : bitwise-OR of `virDomainDeviceModifyFlags`

Change a virtual device on a domain, using the flags parameter to control how the device is changed. `VIR_DOMAIN_AFFECT_CURRENT` specifies that the device change is made based on current domain state. `VIR_DOMAIN_AFFECT_LIVE` specifies that the device shall be changed on the active domain instance only and is not added to the persisted domain configuration. `VIR_DOMAIN_AFFECT_CONFIG` specifies that the device shall be changed on the persisted domain configuration only. Note that the target hypervisor must return an error if unable to satisfy flags. E.g. the hypervisor driver will return failure if `LIVE` is specified but it only supports modifying the persisted device allocation.

This method is used for actions such changing CDROM/Floppy device media, altering the graphics configuration such as password, reconfiguring the NIC device backend connectivity, etc.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.218 `int virDrvSupportsFeature ( virConnectPtr conn, int feature )`

#### 4.7.2.219 `int virGetVersion ( unsigned long * libVer, const char *type ATTRIBUTE_UNUSED, unsigned long * typeVer )`

`virGetVersion`: : return value for the library version (OUT) : ignored; pass NULL : pass NULL; for historical purposes duplicates if non-NULL

Provides version information. is the version of the library and will always be set unless an error occurs, in which case an error code will be returned. exists for historical compatibility; if it is not NULL it will duplicate (it was originally intended to return hypervisor information based on , but due to the design of remote clients this is not reliable). To get the version of the running hypervisor use the `virConnectGetVersion` function instead. To get the libvirt library version used by a connection use the `virConnectGetLibVersion` instead.

Returns -1 in case of failure, 0 otherwise, and values for and have the format `major * 1,000,000 + minor * 1,000 + release`.

#### 4.7.2.220 `int virInitialize ( void )`

`virInitialize`:

Initialize the library. It's better to call this routine at startup in multithreaded applications to avoid potential race when initializing the library.

Calling `virInitialize` is mandatory, unless your first API call is one of `virConnectOpen*`.

Returns 0 in case of success, -1 in case of error

#### 4.7.2.221 `int virInterfaceChangeBegin ( virConnectPtr conn, unsigned int flags )`

`virInterfaceChangeBegin`: : pointer to hypervisor connection : extra flags; not used yet, so callers should always pass 0

This function creates a restore point to which one can return later by calling [virInterfaceChangeRollback\(\)](#). This function should be called before any transaction with interface configuration. Once it is known that a new configuration

works, it can be committed via [virInterfaceChangeCommit\(\)](#), which frees the restore point.

If [virInterfaceChangeBegin\(\)](#) is called when a transaction is already opened, this function will fail, and a `VIR_ERR_INVALID_OPERATION` will be logged.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.222 `int virInterfaceChangeCommit ( virConnectPtr conn, unsigned int flags )`

`virInterfaceChangeCommit`: : pointer to hypervisor connection : extra flags; not used yet, so callers should always pass 0

This commits the changes made to interfaces and frees the restore point created by [virInterfaceChangeBegin\(\)](#).

If [virInterfaceChangeCommit\(\)](#) is called when a transaction is not opened, this function will fail, and a `VIR_ERR_INVALID_OPERATION` will be logged.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.223 `int virInterfaceChangeRollback ( virConnectPtr conn, unsigned int flags )`

`virInterfaceChangeRollback`: : pointer to hypervisor connection : extra flags; not used yet, so callers should always pass 0

This cancels changes made to interfaces settings by restoring previous state created by [virInterfaceChangeBegin\(\)](#).

If [virInterfaceChangeRollback\(\)](#) is called when a transaction is not opened, this function will fail, and a `VIR_ERR_INVALID_OPERATION` will be logged.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.224 `int virInterfaceCreate ( virInterfacePtr iface, unsigned int flags )`

`virInterfaceCreate`: : pointer to a defined interface : extra flags; not used yet, so callers should always pass 0

Activate an interface (i.e. call "ifup").

If there was an open network config transaction at the time this interface was defined (that is, if [virInterfaceChangeBegin\(\)](#) had been called), the interface will be brought back down (and then undefined) if [virInterfaceChangeRollback\(\)](#) is called. p \* Returns 0 in case of success, -1 in case of error

#### 4.7.2.225 `virInterfacePtr virInterfaceDefineXML ( virConnectPtr conn, const char * xml, unsigned int flags )`

`virInterfaceDefineXML`: : pointer to the hypervisor connection : the XML description for the interface, preferably in UTF-8 : extra flags; not used yet, so callers should always pass 0

Define an interface (or modify existing interface configuration).

Normally this change in the interface configuration is immediately permanent/persistent, but if [virInterfaceChangeBegin\(\)](#) has been previously called (i.e. if an interface config transaction is open), the new interface definition will only become permanent if [virInterfaceChangeCommit\(\)](#) is called prior to the next reboot of the system running libvirtd. Prior to that time, it can be explicitly removed using [virInterfaceChangeRollback\(\)](#), or will be automatically removed during the next reboot of the system running libvirtd.

Returns NULL in case of error, a pointer to the interface otherwise

#### 4.7.2.226 `int virInterfaceDestroy ( virInterfacePtr iface, unsigned int flags )`

`virInterfaceDestroy`: : an interface object : extra flags; not used yet, so callers should always pass 0

deactivate an interface (ie call "ifdown") This does not remove the interface from the config, and does not free the associated `virInterfacePtr` object.

If there is an open network config transaction at the time this interface is destroyed (that is, if [virInterfaceChangeBegin\(\)](#) had been called), and if the interface is later undefined and then [virInterfaceChangeRollback\(\)](#) is called, the restoral of the interface definition will also bring the interface back up.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.227 int virInterfaceFree ( virInterfacePtr iface )

virInterfaceFree: : an interface object

Free the interface object. The interface itself is unaltered. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.228 virConnectPtr virInterfaceGetConnect ( virInterfacePtr iface )

virInterfaceGetConnect: : pointer to an interface

Provides the connection pointer associated with an interface. The reference counter on the connection is not increased by this call.

WARNING: When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the interface object together.

Returns the virConnectPtr or NULL in case of failure.

#### 4.7.2.229 const char\* virInterfaceGetMACString ( virInterfacePtr iface )

virInterfaceGetMACString: : an interface object

Get the MAC for an interface as string. For more information about MAC see RFC4122.

Returns a pointer to the MAC address (in null-terminated ASCII format) or NULL, the string need not be deallocated its lifetime will be the same as the interface object.

#### 4.7.2.230 const char\* virInterfaceGetName ( virInterfacePtr iface )

virInterfaceGetName: : an interface object

Get the public name for that interface

Returns a pointer to the name or NULL, the string need not be deallocated its lifetime will be the same as the interface object.

#### 4.7.2.231 char\* virInterfaceGetXMLDesc ( virInterfacePtr iface, unsigned int flags )

virInterfaceGetXMLDesc: : an interface object : bitwise-OR of extraction flags. Current valid bits:

```
VIR_INTERFACE_XML_INACTIVE - return the static configuration,
                             suitable for use redefining the
                             interface via virInterfaceDefineXML()
```

Provide an XML description of the interface. If VIR\_INTERFACE\_XML\_INACTIVE is set, the description may be reused later to redefine the interface with [virInterfaceDefineXML\(\)](#). If it is not set, the ip address and netmask will be the current live setting of the interface, not the settings from the config files.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

#### 4.7.2.232 `int virInterfaceIsActive ( virInterfacePtr iface )`

`virInterfaceIsActive`: : pointer to the interface object

Determine if the interface is currently running

Returns 1 if running, 0 if inactive, -1 on error

#### 4.7.2.233 `virInterfacePtr virInterfaceLookupByMACString ( virConnectPtr conn, const char * macstr )`

`virInterfaceLookupByMACString`: : pointer to the hypervisor connection : the MAC for the interface (null-terminated ASCII format)

Try to lookup an interface on the given hypervisor based on its MAC.

Returns a new interface object or NULL in case of failure. If the interface cannot be found, then `VIR_ERR_NO_INTERFACE` error is raised.

#### 4.7.2.234 `virInterfacePtr virInterfaceLookupByName ( virConnectPtr conn, const char * name )`

#### 4.7.2.235 `int virInterfaceRef ( virInterfacePtr iface )`

`virInterfaceRef`: : the interface to hold a reference on

Increment the reference count on the interface. For each additional call to this method, there shall be a corresponding call to `virInterfaceFree` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using an interface would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.236 `int virInterfaceUndefine ( virInterfacePtr iface )`

`virInterfaceUndefine`: : pointer to a defined interface

Undefine an interface, ie remove it from the config. This does not free the associated `virInterfacePtr` object.

Normally this change in the interface configuration is permanent/persistent, but if `virInterfaceChangeBegin()` has been previously called (i.e. if an interface config transaction is open), the removal of the interface definition will only become permanent if `virInterfaceChangeCommit()` is called prior to the next reboot of the system running libvirt. Prior to that time, the definition can be explicitly restored using `virInterfaceChangeRollback()`, or will be automatically restored during the next reboot of the system running libvirt.

Returns 0 in case of success, -1 in case of error

#### 4.7.2.237 `int virNetworkCreate ( virNetworkPtr network )`

`virNetworkCreate`: : pointer to a defined network

Create and start a defined network. If the call succeed the network moves from the defined to the running networks pools.

Returns 0 in case of success, -1 in case of error

#### 4.7.2.238 `virNetworkPtr virNetworkCreateXML ( virConnectPtr conn, const char * xmlDesc )`

`virNetworkCreateXML`: : pointer to the hypervisor connection : an XML description of the network

Create and start a new virtual network, based on an XML description similar to the one returned by [virNetworkGetXMLDesc\(\)](#)

Returns a new network object or NULL in case of failure

#### 4.7.2.239 **virNetworkPtr** virNetworkDefineXML ( **virConnectPtr** *conn*, **const char \*** *xml* )

virNetworkDefineXML: : pointer to the hypervisor connection : the XML description for the network, preferably in UTF-8

Define a network, but does not create it

Returns NULL in case of error, a pointer to the network otherwise

#### 4.7.2.240 **int** virNetworkDestroy ( **virNetworkPtr** *network* )

virNetworkDestroy: : a network object

Destroy the network object. The running instance is shutdown if not down already and all resources used by it are given back to the hypervisor. This does not free the associated virNetworkPtr object. This function may require privileged access

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.241 **int** virNetworkFree ( **virNetworkPtr** *network* )

virNetworkFree: : a network object

Free the network object. The running instance is kept alive. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.242 **int** virNetworkGetAutostart ( **virNetworkPtr** *network*, **int \*** *autostart* )

virNetworkGetAutostart: : a network object : the value returned

Provides a boolean value indicating whether the network configured to be automatically started when the host machine boots.

Returns -1 in case of error, 0 in case of success

#### 4.7.2.243 **char\*** virNetworkGetBridgeName ( **virNetworkPtr** *network* )

virNetworkGetBridgeName: : a network object

Provides a bridge interface name to which a domain may connect a network interface in order to join the network.

Returns a 0 terminated interface name, or NULL in case of error. the caller must free() the returned value.

#### 4.7.2.244 **POL New** **char\*** virNetworkGetBridgeType ( **virNetworkPtr** *network* )

virNetworkGetBridgeType: : a network object

Provides a bridge interface type to which a domain may connect a network interface in order to join the network.

Returns a 0 terminated interface name, or NULL in case of error. the caller must free() the returned value.

**4.7.2.245 virConnectPtr virNetworkGetConnect ( virNetworkPtr net )**

virNetworkGetConnect: : pointer to a network

Provides the connection pointer associated with a network. The reference counter on the connection is not increased by this call.

WARNING: When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the network object together.

Returns the virConnectPtr or NULL in case of failure.

**4.7.2.246 const char\* virNetworkGetName ( virNetworkPtr network )**

virNetworkGetName: : a network object

Get the public name for that network

Returns a pointer to the name or NULL, the string need not be deallocated its lifetime will be the same as the network object.

**4.7.2.247 int virNetworkGetUUID ( virNetworkPtr network, unsigned char \* uuid )**

virNetworkGetUUID: : a network object : pointer to a VIR\_UUID\_BUFLen bytes array

Get the UUID for a network

Returns -1 in case of error, 0 in case of success

**4.7.2.248 int virNetworkGetUUIDString ( virNetworkPtr network, char \* buf )**

virNetworkGetUUIDString: : a network object : pointer to a VIR\_UUID\_STRING\_BUFLen bytes array

Get the UUID for a network as string. For more information about UUID see RFC4122.

Returns -1 in case of error, 0 in case of success

**4.7.2.249 char\* virNetworkGetXMLDesc ( virNetworkPtr network, unsigned int flags )**

virNetworkGetXMLDesc: : a network object : bitwise-OR of virNetworkXMLFlags

Provide an XML description of the network. The description may be reused later to relaunch the network with [virNetworkCreateXML\(\)](#).

Normally, if a network included a physical function, the output includes all virtual functions tied to that physical interface. If includes VIR\_NETWORK\_XML\_INACTIVE, then the expansion of virtual interfaces is not performed.

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

**4.7.2.250 int virNetworkIsActive ( virNetworkPtr net )**

virNetworkIsActive: : pointer to the network object

Determine if the network is currently running

Returns 1 if running, 0 if inactive, -1 on error

**4.7.2.251 int virNetworkIsPersistent ( virNetworkPtr net )**

virNetworkIsPersistent: : pointer to the network object

Determine if the network has a persistent configuration which means it will still exist after shutting down

Returns 1 if persistent, 0 if transient, -1 on error

**4.7.2.252** `virNetworkPtr virNetworkLookupByName ( virConnectPtr conn, const char * name )`

**4.7.2.253** `virNetworkPtr virNetworkLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

`virNetworkLookupByUUID`: : pointer to the hypervisor connection : the raw UUID for the network

Try to lookup a network on the given hypervisor based on its UUID.

Returns a new network object or NULL in case of failure. If the network cannot be found, then VIR\_ERR\_NO\_NETWORK error is raised.

**4.7.2.254** `virNetworkPtr virNetworkLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

`virNetworkLookupByUUIDString`: : pointer to the hypervisor connection : the string UUID for the network

Try to lookup a network on the given hypervisor based on its UUID.

Returns a new network object or NULL in case of failure. If the network cannot be found, then VIR\_ERR\_NO\_NETWORK error is raised.

**4.7.2.255** `int virNetworkRef ( virNetworkPtr network )`

`virNetworkRef`: : the network to hold a reference on

Increment the reference count on the network. For each additional call to this method, there shall be a corresponding call to `virNetworkFree` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a network would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.256** `int virNetworkSetAutostart ( virNetworkPtr network, int autostart )`

`virNetworkSetAutostart`: : a network object : whether the network should be automatically started 0 or 1

Configure the network to be automatically started when the host machine boots.

Returns -1 in case of error, 0 in case of success

**4.7.2.257** `int virNetworkUndefine ( virNetworkPtr network )`

`virNetworkUndefine`: : pointer to a defined network

Undefine a network but does not stop it if it is running

Returns 0 in case of success, -1 in case of error

**4.7.2.258** `int virNetworkUpdate ( virNetworkPtr network, unsigned int command, unsigned int section, int parentIndex, const char * xml, unsigned int flags )`

`virNetworkUpdate`: : pointer to a defined network

#### 4.7.2.259 **virNodeDevicePtr virNodeDeviceCreateXML ( virConnectPtr conn, const char \* xmlDesc, unsigned int flags )**

virNodeDeviceCreateXML: : pointer to the hypervisor connection : string containing an XML description of the device to be created : extra flags; not used yet, so callers should always pass 0

Create a new device on the VM host machine, for example, virtual HBAs created using vport\_create.

Returns a node device object if successful, NULL in case of failure

#### 4.7.2.260 **int virNodeDeviceDestroy ( virNodeDevicePtr dev )**

virNodeDeviceDestroy: : a device object

Destroy the device object. The virtual device is removed from the host operating system. This function may require privileged access

Returns 0 in case of success and -1 in case of failure.

#### 4.7.2.261 **int virNodeDeviceDetach ( virNodeDevicePtr dev )**

virNodeDeviceDetach: : pointer to the node device

Detach the node device from the node itself so that it may be assigned to a guest domain.

Depending on the hypervisor, this may involve operations such as unbinding any device drivers from the device, binding the device to a dummy device driver and resetting the device.

If the device is currently in use by the node, this method may fail.

Once the device is not assigned to any guest, it may be re-attached to the node using the virNodeDeviceReattach() method.

Returns 0 in case of success, -1 in case of failure.

#### 4.7.2.262 **int virNodeDeviceFree ( virNodeDevicePtr dev )**

virNodeDeviceFree: : pointer to the node device

Drops a reference to the node device, freeing it if this was the last reference.

Returns the 0 for success, -1 for error.

#### 4.7.2.263 **const char\* virNodeDeviceGetName ( virNodeDevicePtr dev )**

virNodeDeviceGetName: : the device

Just return the device name

Returns the device name or NULL in case of error

#### 4.7.2.264 **const char\* virNodeDeviceGetParent ( virNodeDevicePtr dev )**

virNodeDeviceGetParent: : the device

Accessor for the parent of the device

Returns the name of the device's parent, or NULL if the device has no parent.

#### 4.7.2.265 **char\* virNodeDeviceGetXMLDesc ( virNodeDevicePtr dev, unsigned int flags )**

virNodeDeviceGetXMLDesc: : pointer to the node device : extra flags; not used yet, so callers should always pass 0



Fetch an XML document describing all aspects of the device.

Returns the XML document, or NULL on error

**4.7.2.266** `int virNodeDeviceListCaps ( virNodeDevicePtr dev, char **const names, int maxnames )`

virNodeDeviceListCaps: : the device : array to collect the list of capability names : size of

Lists the names of the capabilities supported by the device.

Returns the number of capability names listed in .

**4.7.2.267** `virNodeDevicePtr virNodeDeviceLookupByName ( virConnectPtr conn, const char * name )`

**4.7.2.268** `int virNodeDeviceNumOfCaps ( virNodeDevicePtr dev )`

virNodeDeviceNumOfCaps: : the device

Accessor for the number of capabilities supported by the device.

Returns the number of capabilities supported by the device.

**4.7.2.269** `int virNodeDeviceReAttach ( virNodeDevicePtr dev )`

virNodeDeviceReAttach: : pointer to the node device

Re-attach a previously detached node device to the node so that it may be used by the node again.

Depending on the hypervisor, this may involve operations such as resetting the device, unbinding it from a dummy device driver and binding it to its appropriate driver.

If the device is currently in use by a guest, this method may fail.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.270** `int virNodeDeviceRef ( virNodeDevicePtr dev )`

virNodeDeviceRef: : the dev to hold a reference on

Increment the reference count on the dev. For each additional call to this method, there shall be a corresponding call to virNodeDeviceFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a dev would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.271** `int virNodeDeviceReset ( virNodeDevicePtr dev )`

virNodeDeviceReset: : pointer to the node device

Reset a previously detached node device to the node before or after assigning it to a guest.

The exact reset semantics depends on the hypervisor and device type but, for example, KVM will attempt to reset PCI devices with a Function Level Reset, Secondary Bus Reset or a Power Management D-State reset.

If the reset will affect other devices which are currently in use, this function may fail.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.272** `int virNodeGetCellsFreeMemory ( virConnectPtr conn, unsigned long long * freeMems, int startCell, int maxCells )`

`virNodeGetCellsFreeMemory`: : pointer to the hypervisor connection : pointer to the array of unsigned long long : index of first cell to return freeMems info on. : Maximum number of cells for which freeMems information can be returned.

This call returns the amount of free memory in one or more NUMA cells. The array must be allocated by the caller and will be filled with the amount of free memory in bytes for each cell requested, starting with `startCell` (in `freeMems[0]`), up to either `(startCell + maxCells)`, or the number of additional cells in the node, whichever is smaller.

Returns the number of entries filled in `freeMems`, or -1 in case of error.

**4.7.2.273** `int virNodeGetCPUStats ( virConnectPtr conn, int cpuNum, virNodeCPUStatsPtr params, int * nparams, unsigned int flags )`

`virNodeGetCPUStats`: : pointer to the hypervisor connection. : number of node cpu. (`VIR_NODE_CPU_STATS_ALL_CPUS` means total cpu statistics) : pointer to node cpu time parameter objects : number of node cpu time parameter (this value should be same or less than the number of parameters supported) : extra flags; not used yet, so callers should always pass 0

This function provides individual cpu statistics of the node. If you want to get total cpu statistics of the node, you must specify `VIR_NODE_CPU_STATS_ALL_CPUS` to . The array will be filled with the values equal to the number of parameters suggested by

As the value of is dynamic, call the API setting to 0 and as NULL, the API returns the number of parameters supported by the HV by updating on SUCCESS. The caller should then allocate array, i.e. `(sizeof() * )` bytes and call the API again.

Here is a sample code snippet:

```
if ((virNodeGetCPUStats(conn, cpuNum, NULL, &nparams, 0) == 0) && (nparams != 0)) { if ((params = malloc(sizeof(virNodeCPUStats) * nparams)) == NULL) goto error; memset(params, 0, sizeof(virNodeCPUStats) * nparams); if (virNodeGetCPUStats(conn, cpuNum, params, &nparams, 0)) goto error; }
```

This function doesn't require privileged access to the hypervisor. This function expects the caller to allocate the .

CPU time Statistics:

`VIR_NODE_CPU_STATS_KERNEL`: The cumulative CPU time which spends by kernel, when the node booting up.(nanoseconds) `VIR_NODE_CPU_STATS_USER`: The cumulative CPU time which spends by user processes, when the node booting up.(nanoseconds) `VIR_NODE_CPU_STATS_IDLE`: The cumulative idle CPU time, when the node booting up.(nanoseconds) `VIR_NODE_CPU_STATS_IOWAIT`: The cumulative I/O wait CPU time, when the node booting up.(nanoseconds) `VIR_NODE_CPU_STATS_UTILIZATION`: The CPU utilization. The usage value is in percent and 100% represents all CPUs on the server.

Returns -1 in case of error, 0 in case of success.

**4.7.2.274** `unsigned long long virNodeGetFreeMemory ( virConnectPtr conn )`

`virNodeGetFreeMemory`: : pointer to the hypervisor connection

provides the free memory available on the Node Note: most libvirt APIs provide memory sizes in kibibytes, but in this function the returned value is in bytes. Divide by 1024 as necessary.

Returns the available free memory in bytes or 0 in case of error

**4.7.2.275** `int virNodeGetInfo ( virConnectPtr conn, virNodeInfoPtr info )`

`virNodeGetInfo`: : pointer to the hypervisor connection : pointer to a `virNodeInfo` structure allocated by the user

Extract hardware information about the node.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.276** `int virNodeGetMemoryParameters ( virConnectPtr conn, virTypedParameterPtr params, int * nparams, unsigned int flags )`

**4.7.2.277** `int virNodeGetMemoryStats ( virConnectPtr conn, int cellNum, virNodeMemoryStatsPtr params, int * nparams, unsigned int flags )`

`virNodeGetMemoryStats`: : pointer to the hypervisor connection. : number of node cell. (VIR\_NODE\_MEMORY\_STATS\_ALL\_CELLS means total cell statistics) : pointer to node memory stats objects : number of node memory stats (this value should be same or less than the number of stats supported) : extra flags; not used yet, so callers should always pass 0

This function provides memory stats of the node. If you want to get total cpu statistics of the node, you must specify VIR\_NODE\_MEMORY\_STATS\_ALL\_CELLS to . The array will be filled with the values equal to the number of stats suggested by

As the value of is dynamic, call the API setting to 0 and as NULL, the API returns the number of parameters supported by the HV by updating on SUCCESS. The caller should then allocate array, i.e. (sizeof() \* ) bytes and call the API again.

Here is the sample code snippet:

```
if ((virNodeGetMemoryStats(conn, cellNum, NULL, &nparams, 0) == 0) && (nparams != 0)) { if ((params = malloc(sizeof(virNodeMemoryStats) * nparams)) == NULL) goto error; memset(params, cellNum, 0, sizeof(virNodeMemoryStats) * nparams); if (virNodeGetMemoryStats(conn, params, &nparams, 0)) goto error; }
```

This function doesn't require privileged access to the hypervisor. This function expects the caller to allocate the .

Memory Stats:

VIR\_NODE\_MEMORY\_STATS\_TOTAL: The total memory usage.(KB) VIR\_NODE\_MEMORY\_STATS\_FREE: : The free memory usage.(KB) On linux, this usage includes buffers and cached. VIR\_NODE\_MEMORY\_STATS\_BUFFERS: The buffers memory usage.(KB) VIR\_NODE\_MEMORY\_STATS\_CACHED: The cached memory usage.(KB)

Returns -1 in case of error, 0 in case of success.

**4.7.2.278** `int virNodeGetSecurityModel ( virConnectPtr conn, virSecurityModelPtr secmodel )`

`virNodeGetSecurityModel`: : a connection object : pointer to a virSecurityModel structure

Extract the security model of a hypervisor. The 'model' field in the argument may be initialized to the empty string if the driver has not activated a security model.

Returns 0 in case of success, -1 in case of failure

**4.7.2.279** `int virNodeListDevices ( virConnectPtr conn, const char * cap, char **const names, int maxnames, unsigned int flags )`

`virNodeListDevices`: : pointer to the hypervisor connection : capability name : array to collect the list of node device names : size of : extra flags; not used yet, so callers should always pass 0

Collect the list of node devices, and store their names in

For more control over the results, see [virConnectListAllNodeDevices\(\)](#).

If the optional 'cap' argument is non-NULL, then the count will be restricted to devices with the specified capability

Returns the number of node devices found or -1 in case of error

**4.7.2.280** `int virNodeNumOfDevices ( virConnectPtr conn, const char * cap, unsigned int flags )`

`virNodeNumOfDevices`: : pointer to the hypervisor connection : capability name : extra flags; not used yet, so callers should always pass 0

Provides the number of node devices.

If the optional 'cap' argument is non-NULL, then the count will be restricted to devices with the specified capability

Returns the number of node devices or -1 in case of error

**4.7.2.281** `int virNodeSetMemoryParameters ( virConnectPtr conn, virTypedParameterPtr params, int nparams, unsigned int flags )`

**4.7.2.282** `int virNodeSuspendForDuration ( virConnectPtr conn, unsigned int target, unsigned long long duration, unsigned int flags )`

`virNodeSuspendForDuration`: : pointer to the hypervisor connection : the state to which the host must be suspended to, such as: `VIR_NODE_SUSPEND_TARGET_MEM` (Suspend-to-RAM) `VIR_NODE_SUSPEND_TARGET_DISK` (Suspend-to-Disk) `VIR_NODE_SUSPEND_TARGET_HYBRID` (Hybrid-Suspend, which is a combination of the former modes). : the time duration in seconds for which the host has to be suspended : extra flags; not used yet, so callers should always pass 0

Attempt to suspend the node (host machine) for the given duration of time in the specified state (Suspend-to-RAM, Suspend-to-Disk or Hybrid-Suspend). Schedule the node's Real-Time-Clock interrupt to resume the node after the duration is complete.

Returns 0 on success (i.e., the node will be suspended after a short delay), -1 on failure (the operation is not supported, or an attempted suspend is already underway).

**4.7.2.283** `virNWFilterPtr virNWFilterDefineXML ( virConnectPtr conn, const char * xmlDesc )`

`virNWFilterDefineXML`: : pointer to the hypervisor connection : an XML description of the nwfilter

Define a new network filter, based on an XML description similar to the one returned by [virNWFilterGetXMLDesc\(\)](#)

Returns a new nwfilter object or NULL in case of failure

**4.7.2.284** `int virNWFilterFree ( virNWFilterPtr nwfilter )`

`virNWFilterFree`: : a nwfilter object

Free the nwfilter object. The running instance is kept alive. The data structure is freed and should not be used thereafter.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.285** `const char* virNWFilterGetName ( virNWFilterPtr nwfilter )`

`virNWFilterGetName`: : a nwfilter object

Get the public name for the network filter

Returns a pointer to the name or NULL, the string need not be deallocated its lifetime will be the same as the nwfilter object.

**4.7.2.286** `int virNWFilterGetUUID ( virNWFilterPtr nwfilter, unsigned char * uuid )`

`virNWFilterGetUUID`: : a nwfilter object : pointer to a `VIR_UUID_BUFLen` bytes array

Get the UUID for a network filter

Returns -1 in case of error, 0 in case of success

**4.7.2.287** `int virNWFilterGetUUIDString ( virNWFilterPtr nwfilter, char * buf )`

virNWFilterGetUUIDString: : a nwfilter object : pointer to a VIR\_UUID\_STRING\_BUFLen bytes array

Get the UUID for a network filter as string. For more information about UUID see RFC4122.

Returns -1 in case of error, 0 in case of success

**4.7.2.288** `char* virNWFilterGetXMLDesc ( virNWFilterPtr nwfilter, unsigned int flags )`

virNWFilterGetXMLDesc: : a nwfilter object : extra flags; not used yet, so callers should always pass 0

Provide an XML description of the network filter. The description may be reused later to redefine the network filter with virNWFilterCreateXML().

Returns a 0 terminated UTF-8 encoded XML instance, or NULL in case of error. the caller must free() the returned value.

**4.7.2.289** `virNWFilterPtr virNWFilterLookupByName ( virConnectPtr conn, const char * name )`

**4.7.2.290** `virNWFilterPtr virNWFilterLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

virNWFilterLookupByUUID: : pointer to the hypervisor connection : the raw UUID for the network filter

Try to lookup a network filter on the given hypervisor based on its UUID.

Returns a new nwfilter object or NULL in case of failure. If the nwfilter cannot be found, then VIR\_ERR\_NO\_NWFILTER error is raised.

**4.7.2.291** `virNWFilterPtr virNWFilterLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

virNWFilterLookupByUUIDString: : pointer to the hypervisor connection : the string UUID for the nwfilter

Try to lookup an nwfilter on the given hypervisor based on its UUID.

Returns a new nwfilter object or NULL in case of failure. If the nwfilter cannot be found, then VIR\_ERR\_NO\_NWFILTER error is raised.

**4.7.2.292** `int virNWFilterRef ( virNWFilterPtr nwfilter )`

virNWFilterRef: : the nwfilter to hold a reference on

Increment the reference count on the nwfilter. For each additional call to this method, there shall be a corresponding call to virNWFilterFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using an nwfilter would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.293** `int virNWFilterUndefine ( virNWFilterPtr nwfilter )`

virNWFilterUndefine: : a nwfilter object

Undefine the nwfilter object. This call will not succeed if a running VM is referencing the filter. This does not free the associated virNWFilterPtr object.

Returns 0 in case of success and -1 in case of failure.

**4.7.2.294 int virRegisterDeviceMonitor ( virDeviceMonitorPtr driver )**

virRegisterDeviceMonitor: : pointer to a device monitor block

Register a device monitor

Returns the driver priority or -1 in case of error.

**4.7.2.295 int virRegisterDriver ( virDriverPtr driver )**

virRegisterDriver: : pointer to a driver block

Register a virtualization driver

Returns the driver priority or -1 in case of error.

**4.7.2.296 int virRegisterInterfaceDriver ( virInterfaceDriverPtr driver )**

virRegisterInterfaceDriver: : pointer to an interface driver block

Register an interface virtualization driver

Returns the driver priority or -1 in case of error.

**4.7.2.297 int virRegisterNetworkDriver ( virNetworkDriverPtr driver )**

virRegisterNetworkDriver: : pointer to a network driver block

Register a network virtualization driver

Returns the driver priority or -1 in case of error.

**4.7.2.298 int virRegisterNWFilterDriver ( virNWFilterDriverPtr driver )**

virRegisterNWFilterDriver: : pointer to a network filter driver block

Register a network filter virtualization driver

Returns the driver priority or -1 in case of error.

**4.7.2.299 int virRegisterSecretDriver ( virSecretDriverPtr driver )**

virRegisterSecretDriver: : pointer to a secret driver block

Register a secret driver

Returns the driver priority or -1 in case of error.

**4.7.2.300 int virRegisterStorageDriver ( virStorageDriverPtr driver )**

virRegisterStorageDriver: : pointer to a storage driver block

Register a storage virtualization driver

Returns the driver priority or -1 in case of error.

**4.7.2.301 virSecretPtr virSecretDefineXML ( virConnectPtr conn, const char \* xml, unsigned int flags )**

virSecretDefineXML: : virConnect connection : XML describing the secret. : extra flags; not used yet, so callers should always pass 0

If XML specifies a UUID, locates the specified secret and replaces all attributes of the secret specified by UUID by attributes specified in xml (any attributes not specified in xml are discarded).

Otherwise, creates a new secret with an automatically chosen UUID, and initializes its attributes from xml.

Returns a the secret on success, NULL on failure.

#### 4.7.2.302 `int virSecretFree ( virSecretPtr secret )`

`virSecretFree`: : pointer to a secret

Release the secret handle. The underlying secret continues to exist.

Returns 0 on success, or -1 on error

#### 4.7.2.303 `virConnectPtr virSecretGetConnect ( virSecretPtr secret )`

`virSecretGetConnect`: : A `virSecret` secret

Provides the connection pointer associated with a secret. The reference counter on the connection is not increased by this call.

**WARNING:** When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the secret object together.

Returns the `virConnectPtr` or NULL in case of failure.

#### 4.7.2.304 `const char* virSecretGetUsageID ( virSecretPtr secret )`

`virSecretGetUsageID`: : a secret object

Get the unique identifier of the object with which this secret is to be used. The format of the identifier is dependant on the usage type of the secret. For a secret with a usage type of `VIR_SECRET_USAGE_TYPE_VOLUME` the identifier will be a fully qualified path name. The identifiers are intended to be unique within the set of all secrets sharing the same usage type. ie, there shall only ever be one secret for each volume path.

Returns a string identifying the object using the secret, or NULL upon error

#### 4.7.2.305 `int virSecretGetUsageType ( virSecretPtr secret )`

`virSecretGetUsageType`: : a secret object

Get the type of object which uses this secret. The returned value is one of the constants defined in the `virSecretUsageType` enumeration. More values may be added to this enumeration in the future, so callers should expect to see usage types they do not explicitly know about.

Returns a positive integer identifying the type of object, or -1 upon error.

#### 4.7.2.306 `int virSecretGetUUID ( virSecretPtr secret, unsigned char * uuid )`

`virSecretGetUUID`: : A `virSecret` secret : buffer of `VIR_UUID_BUFLen` bytes in size

Fetches the UUID of the secret.

Returns 0 on success with the uuid buffer being filled, or -1 upon failure.

#### 4.7.2.307 `int virSecretGetUUIDString ( virSecretPtr secret, char * buf )`

`virSecretGetUUIDString`: : a secret object : pointer to a `VIR_UUID_STRING_BUFLen` bytes array

Get the UUID for a secret as string. For more information about UUID see RFC4122.

Returns -1 in case of error, 0 in case of success

#### 4.7.2.308 `unsigned char* virSecretGetValue ( virSecretPtr secret, size_t * value_size, unsigned int flags )`

`virSecretGetValue`: : A `virSecret` connection : Place for storing size of the secret value : extra flags; not used yet, so callers should always pass 0

Fetches the value of a secret.

Returns the secret value on success, NULL on failure. The caller must `free()` the secret value.

#### 4.7.2.309 `char* virSecretGetXMLDesc ( virSecretPtr secret, unsigned int flags )`

`virSecretGetXMLDesc`: : A `virSecret` secret : extra flags; not used yet, so callers should always pass 0

Fetches an XML document describing attributes of the secret.

Returns the XML document on success, NULL on failure. The caller must `free()` the XML.

#### 4.7.2.310 `virSecretPtr virSecretLookupByUsage ( virConnectPtr conn, int usageType, const char * usageID )`

`virSecretLookupByUsage`: : pointer to the hypervisor connection : the type of secret usage : identifier of the object using the secret

Try to lookup a secret on the given hypervisor based on its usage The `usageID` is unique within the set of secrets sharing the same `usageType` value.

Returns a new secret object or NULL in case of failure. If the secret cannot be found, then `VIR_ERR_NO_SECRET` error is raised.

#### 4.7.2.311 `virSecretPtr virSecretLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

`virSecretLookupByUUID`: : pointer to the hypervisor connection : the raw UUID for the secret

Try to lookup a secret on the given hypervisor based on its UUID. Uses the 16 bytes of raw data to describe the UUID

Returns a new secret object or NULL in case of failure. If the secret cannot be found, then `VIR_ERR_NO_SECRET` error is raised.

#### 4.7.2.312 `virSecretPtr virSecretLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

`virSecretLookupByUUIDString`: : pointer to the hypervisor connection : the string UUID for the secret

Try to lookup a secret on the given hypervisor based on its UUID. Uses the printable string value to describe the UUID

Returns a new secret object or NULL in case of failure. If the secret cannot be found, then `VIR_ERR_NO_SECRET` error is raised.

#### 4.7.2.313 `int virSecretRef ( virSecretPtr secret )`

`virSecretRef`: : the secret to hold a reference on

Increment the reference count on the secret. For each additional call to this method, there shall be a corresponding call to `virSecretFree` to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a secret would increment the reference count.



Returns 0 in case of success, -1 in case of failure.

**4.7.2.314** `int virSecretSetValue ( virSecretPtr secret, const unsigned char * value, size_t value_size, unsigned int flags )`

virSecretSetValue: : A virSecret secret : Value of the secret : Size of the value : extra flags; not used yet, so callers should always pass 0

Sets the value of a secret.

Returns 0 on success, -1 on failure.

**4.7.2.315** `int virSecretUndefine ( virSecretPtr secret )`

virSecretUndefine: : A virSecret secret

Deletes the specified secret. This does not free the associated virSecretPtr object.

Returns 0 on success, -1 on failure.

**4.7.2.316** `int virStoragePoolBuild ( virStoragePoolPtr pool, unsigned int flags )`

virStoragePoolBuild: : pointer to storage pool : bitwise-OR of virStoragePoolBuildFlags

Currently only filesystem pool accepts flags VIR\_STORAGE\_POOL\_BUILD\_OVERWRITE and VIR\_STORAGE\_POOL\_BUILD\_NO\_OVERWRITE.

Build the underlying storage pool

Returns 0 on success, or -1 upon failure

**4.7.2.317** `int virStoragePoolCreate ( virStoragePoolPtr pool, unsigned int flags )`

virStoragePoolCreate: : pointer to storage pool : extra flags; not used yet, so callers should always pass 0

Starts an inactive storage pool

Returns 0 on success, or -1 if it could not be started

**4.7.2.318** `virStoragePoolPtr virStoragePoolCreateXML ( virConnectPtr conn, const char * xmlDesc, unsigned int flags )`

virStoragePoolCreateXML: : pointer to hypervisor connection : XML description for new pool : extra flags; not used yet, so callers should always pass 0

Create a new storage based on its XML description. The pool is not persistent, so its definition will disappear when it is destroyed, or if the host is restarted

Returns a virStoragePoolPtr object, or NULL if creation failed

**4.7.2.319** `virStoragePoolPtr virStoragePoolDefineXML ( virConnectPtr conn, const char * xml, unsigned int flags )`

virStoragePoolDefineXML: : pointer to hypervisor connection : XML description for new pool : extra flags; not used yet, so callers should always pass 0

Define a new inactive storage pool based on its XML description. The pool is persistent, until explicitly undefined.

Returns a virStoragePoolPtr object, or NULL if creation failed

**4.7.2.320** `int virStoragePoolDelete ( virStoragePoolPtr pool, unsigned int flags )`

virStoragePoolDelete: : pointer to storage pool : bitwise-OR of virStoragePoolDeleteFlags

Delete the underlying pool resources. This is a non-recoverable operation. The `virStoragePoolPtr` object itself is not free'd.

Returns 0 on success, or -1 if it could not be obliterate

#### 4.7.2.321 `int virStoragePoolDestroy ( virStoragePoolPtr pool )`

`virStoragePoolDestroy`: : pointer to storage pool

Destroy an active storage pool. This will deactivate the pool on the host, but keep any persistent config associated with it. If it has a persistent config it can later be restarted with `virStoragePoolCreate()`. This does not free the associated `virStoragePoolPtr` object.

Returns 0 on success, or -1 if it could not be destroyed

#### 4.7.2.322 `int virStoragePoolFree ( virStoragePoolPtr pool )`

`virStoragePoolFree`: : pointer to storage pool

Free a storage pool object, releasing all memory associated with it. Does not change the state of the pool on the host.

Returns 0 on success, or -1 if it could not be free'd.

#### 4.7.2.323 `int virStoragePoolGetAutostart ( virStoragePoolPtr pool, int * autostart )`

`virStoragePoolGetAutostart`: : pointer to storage pool : location in which to store autostart flag

Fetches the value of the autostart flag, which determines whether the pool is automatically started at boot time

Returns 0 on success, -1 on failure

#### 4.7.2.324 `virConnectPtr virStoragePoolGetConnect ( virStoragePoolPtr pool )`

`virStoragePoolGetConnect`: : pointer to a pool

Provides the connection pointer associated with a storage pool. The reference counter on the connection is not increased by this call.

**WARNING:** When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the pool object together.

Returns the `virConnectPtr` or NULL in case of failure.

#### 4.7.2.325 `int virStoragePoolGetInfo ( virStoragePoolPtr pool, virStoragePoolInfoPtr info )`

`virStoragePoolGetInfo`: : pointer to storage pool : pointer at which to store info

Get volatile information about the storage pool such as free space / usage summary

Returns 0 on success, or -1 on failure.

#### 4.7.2.326 `const char* virStoragePoolGetName ( virStoragePoolPtr pool )`

`virStoragePoolGetName`: : pointer to storage pool

Fetch the locally unique name of the storage pool

Returns the name of the pool, or NULL on error

**4.7.2.327 int virStoragePoolGetUUID ( virStoragePoolPtr pool, unsigned char \* uuid )**

virStoragePoolGetUUID: : pointer to storage pool : buffer of VIR\_UUID\_BUFLen bytes in size

Fetch the globally unique ID of the storage pool

Returns 0 on success, or -1 on error;

**4.7.2.328 int virStoragePoolGetUUIDString ( virStoragePoolPtr pool, char \* buf )**

virStoragePoolGetUUIDString: : pointer to storage pool : buffer of VIR\_UUID\_STRING\_BUFLen bytes in size

Fetch the globally unique ID of the storage pool as a string

Returns 0 on success, or -1 on error;

**4.7.2.329 char\* virStoragePoolGetXMLDesc ( virStoragePoolPtr pool, unsigned int flags )**

virStoragePoolGetXMLDesc: : pointer to storage pool : bitwise-OR of virStorageXMLFlags

Fetch an XML document describing all aspects of the storage pool. This is suitable for later feeding back into the virStoragePoolCreateXML method.

Returns a XML document, or NULL on error

**4.7.2.330 int virStoragePoolsActive ( virStoragePoolPtr pool )**

virStoragePoolsActive: : pointer to the storage pool object

Determine if the storage pool is currently running

Returns 1 if running, 0 if inactive, -1 on error

**4.7.2.331 int virStoragePoolsPersistent ( virStoragePoolPtr pool )**

virStoragePoolsPersistent: : pointer to the storage pool object

Determine if the storage pool has a persistent configuration which means it will still exist after shutting down

Returns 1 if persistent, 0 if transient, -1 on error

**4.7.2.332 int virStoragePoolListAllVolumes ( virStoragePoolPtr pool, virStorageVolPtr \*\* vols, unsigned int flags )**

virStoragePoolListAllVolumes: : Pointer to storage pool : Pointer to a variable to store the array containing storage volume objects or NULL if the list is not required (just returns number of volumes). : extra flags; not used yet, so callers should always pass 0

Collect the list of storage volumes, and allocate an array to store those objects.

Returns the number of storage volumes found or -1 and sets to NULL in case of error. On success, the array stored into is guaranteed to have an extra allocated element set to NULL but not included in the return count, to make iteration easier. The caller is responsible for calling [virStorageVolFree\(\)](#) on each array element, then calling free() on .

**4.7.2.333 int virStoragePoolListVolumes ( virStoragePoolPtr pool, char \*\*const names, int maxnames )**

virStoragePoolListVolumes: : pointer to storage pool : array in which to store storage volume names : size of names array

Fetch list of storage volume names, limiting to at most maxnames.

To list the volume objects directly, see [virStoragePoolListAllVolumes\(\)](#).

Returns the number of names fetched, or -1 on error

**4.7.2.334** `virStoragePoolPtr virStoragePoolLookupByName ( virConnectPtr conn, const char * name )`

**4.7.2.335** `virStoragePoolPtr virStoragePoolLookupByUUID ( virConnectPtr conn, const unsigned char * uuid )`

virStoragePoolLookupByUUID: : pointer to hypervisor connection : globally unique id of pool to fetch

Fetch a storage pool based on its globally unique id

Returns a virStoragePoolPtr object, or NULL if no matching pool is found

**4.7.2.336** `virStoragePoolPtr virStoragePoolLookupByUUIDString ( virConnectPtr conn, const char * uuidstr )`

virStoragePoolLookupByUUIDString: : pointer to hypervisor connection : globally unique id of pool to fetch

Fetch a storage pool based on its globally unique id

Returns a virStoragePoolPtr object, or NULL if no matching pool is found

**4.7.2.337** `virStoragePoolPtr virStoragePoolLookupByVolume ( virStorageVolPtr vol )`

virStoragePoolLookupByVolume: : pointer to storage volume

Fetch a storage pool which contains a particular volume

Returns a virStoragePoolPtr object, or NULL if no matching pool is found

**4.7.2.338** `int virStoragePoolNumOfVolumes ( virStoragePoolPtr pool )`

virStoragePoolNumOfVolumes: : pointer to storage pool

Fetch the number of storage volumes within a pool

Returns the number of storage pools, or -1 on failure

**4.7.2.339** `int virStoragePoolRef ( virStoragePoolPtr pool )`

virStoragePoolRef: : the pool to hold a reference on

Increment the reference count on the pool. For each additional call to this method, there shall be a corresponding call to virStoragePoolFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a pool would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.340** `int virStoragePoolRefresh ( virStoragePoolPtr pool, unsigned int flags )`

virStoragePoolRefresh: : pointer to storage pool : extra flags; not used yet, so callers should always pass 0

Request that the pool refresh its list of volumes. This may involve communicating with a remote server, and/or initializing new devices at the OS layer

Returns 0 if the volume list was refreshed, -1 on failure

**4.7.2.341** int virStoragePoolSetAutostart ( virStoragePoolPtr *pool*, int *autostart* )

virStoragePoolSetAutostart: : pointer to storage pool : new flag setting

Sets the autostart flag

Returns 0 on success, -1 on failure

**4.7.2.342** int virStoragePoolUndefine ( virStoragePoolPtr *pool* )

virStoragePoolUndefine: : pointer to storage pool

Undefine an inactive storage pool

Returns 0 on success, -1 on failure

**4.7.2.343** virStorageVolPtr virStorageVolCreateXML ( virStoragePoolPtr *pool*, const char \* *xmldesc*, unsigned int *flags* )

virStorageVolCreateXML: : pointer to storage pool : description of volume to create : extra flags; not used yet, so callers should always pass 0

Create a storage volume within a pool based on an XML description. Not all pools support creation of volumes

Returns the storage volume, or NULL on error

**4.7.2.344** virStorageVolPtr virStorageVolCreateXMLFrom ( virStoragePoolPtr *pool*, const char \* *xmldesc*, virStorageVolPtr *clonevol*, unsigned int *flags* )

virStorageVolCreateXMLFrom: : pointer to parent pool for the new volume : description of volume to create : storage volume to use as input : extra flags; not used yet, so callers should always pass 0

Create a storage volume in the parent pool, using the 'clonevol' volume as input. Information for the new volume (name, perms) are passed via a typical volume XML description.

Returns the storage volume, or NULL on error

**4.7.2.345** int virStorageVolDelete ( virStorageVolPtr *vol*, unsigned int *flags* )

virStorageVolDelete: : pointer to storage volume : extra flags; not used yet, so callers should always pass 0

Delete the storage volume from the pool

Returns 0 on success, or -1 on error

**4.7.2.346** int virStorageVolDownload ( virStorageVolPtr *vol*, virStreamPtr *stream*, unsigned long long *offset*, unsigned long long *length*, unsigned int *flags* )

virStorageVolDownload: : pointer to volume to download from : stream to use as output : position in to start reading from : limit on amount of data to download : extra flags; not used yet, so callers should always pass 0

Download the content of the volume as a stream. If is zero, then the remaining contents of the volume after will be downloaded.

This call sets up an asynchronous stream; subsequent use of stream APIs is necessary to transfer the actual data, determine how much data is successfully transferred, and detect any errors. The results will be unpredictable if another active stream is writing to the storage volume.

Returns 0, or -1 upon error.

**4.7.2.347 int virStorageVolFree ( virStorageVolPtr vol )**

virStorageVolFree: : pointer to storage volume

Release the storage volume handle. The underlying storage volume continues to exist.

Returns 0 on success, or -1 on error

**4.7.2.348 virConnectPtr virStorageVolGetConnect ( virStorageVolPtr vol )**

virStorageVolGetConnect: : pointer to a pool

Provides the connection pointer associated with a storage volume. The reference counter on the connection is not increased by this call.

WARNING: When writing libvirt bindings in other languages, do not use this function. Instead, store the connection and the volume object together.

Returns the virConnectPtr or NULL in case of failure.

**4.7.2.349 int virStorageVolGetInfo ( virStorageVolPtr vol, virStorageVolInfoPtr info )**

virStorageVolGetInfo: : pointer to storage volume : pointer at which to store info

Fetches volatile information about the storage volume such as its current allocation

Returns 0 on success, or -1 on failure

**4.7.2.350 const char\* virStorageVolGetKey ( virStorageVolPtr vol )**

virStorageVolGetKey: : pointer to storage volume

Fetch the storage volume key. This is globally unique, so the same volume will have the same key no matter what host it is accessed from

Returns the volume key, or NULL on error

**4.7.2.351 const char\* virStorageVolGetName ( virStorageVolPtr vol )**

virStorageVolGetName: : pointer to storage volume

Fetch the storage volume name. This is unique within the scope of a pool

Returns the volume name, or NULL on error

**4.7.2.352 char\* virStorageVolGetPath ( virStorageVolPtr vol )**

virStorageVolGetPath: : pointer to storage volume

Fetch the storage volume path. Depending on the pool configuration this is either persistent across hosts, or dynamically assigned at pool startup. Consult pool documentation for information on getting the persistent naming

Returns the storage volume path, or NULL on error. The caller must free() the returned path after use.

**4.7.2.353 char\* virStorageVolGetXMLDesc ( virStorageVolPtr vol, unsigned int flags )**

virStorageVolGetXMLDesc: : pointer to storage volume : extra flags; not used yet, so callers should always pass 0

Fetch an XML document describing all aspects of the storage volume

Returns the XML document, or NULL on error

**4.7.2.354 virStorageVolPtr virStorageVolLookupByKey ( virConnectPtr conn, const char \* key )**

virStorageVolLookupByKey: : pointer to hypervisor connection : globally unique key

Fetch a pointer to a storage volume based on its globally unique key

Returns a storage volume, or NULL if not found / error

**4.7.2.355 virStorageVolPtr virStorageVolLookupByName ( virStoragePoolPtr pool, const char \* name )****4.7.2.356 virStorageVolPtr virStorageVolLookupByPath ( virConnectPtr conn, const char \* path )**

virStorageVolLookupByPath: : pointer to hypervisor connection : locally unique path

Fetch a pointer to a storage volume based on its locally (host) unique path

Returns a storage volume, or NULL if not found / error

**4.7.2.357 int virStorageVolRef ( virStorageVolPtr vol )**

virStorageVolRef: : the vol to hold a reference on

Increment the reference count on the vol. For each additional call to this method, there shall be a corresponding call to virStorageVolFree to release the reference count, once the caller no longer needs the reference to this object.

This method is typically useful for applications where multiple threads are using a connection, and it is required that the connection remain open until all threads have finished using it. ie, each new thread using a vol would increment the reference count.

Returns 0 in case of success, -1 in case of failure.

**4.7.2.358 int virStorageVolResize ( virStorageVolPtr vol, unsigned long long capacity, unsigned int flags )**

virStorageVolResize: : pointer to storage volume : new capacity, in bytes : bitwise-OR of virStorageVolResizeFlags

Changes the capacity of the storage volume to . The operation will fail if the new capacity requires allocation that would exceed the remaining free space in the parent pool. The contents of the new capacity will appear as all zero bytes.

Normally, the operation will attempt to affect capacity with a minimum impact on allocation (that is, the default operation favors a sparse resize). If contains VIR\_STORAGE\_VOL\_RESIZE\_ALLOCATE, then the operation will ensure that allocation is sufficient for the new capacity; this may make the operation take noticeably longer.

Normally, the operation treats as the new size in bytes; but if contains VIR\_STORAGE\_VOL\_RESIZE\_DELTA, then represents the size difference to add to the current size. It is up to the storage pool implementation whether unaligned requests are rounded up to the next valid boundary, or rejected.

Normally, this operation should only be used to enlarge capacity; but if contains VIR\_STORAGE\_VOL\_RESIZE\_SHRINK, it is possible to attempt a reduction in capacity even though it might cause data loss. If VIR\_STORAGE\_VOL\_RESIZE\_DELTA is also present, then is subtracted from the current size; without it, represents the absolute new size regardless of whether it is larger or smaller than the current size.

Returns 0 on success, or -1 on error.

**4.7.2.359 int virStorageVolUpload ( virStorageVolPtr vol, virStreamPtr stream, unsigned long long offset, unsigned long long length, unsigned int flags )**

virStorageVolUpload: : pointer to volume to upload : stream to use as input : position to start writing to : limit on amount of data to upload : extra flags; not used yet, so callers should always pass 0

Upload new content to the volume from a stream. This call will fail if + exceeds the size of the volume. Otherwise, if is non-zero, an error will be raised if an attempt is made to upload greater than bytes of data.

This call sets up an asynchronous stream; subsequent use of stream APIs is necessary to transfer the actual data, determine how much data is successfully transferred, and detect any errors. The results will be unpredictable if another active stream is writing to the storage volume.

Returns 0, or -1 upon error.

#### 4.7.2.360 `int virStorageVolWipe ( virStorageVolPtr vol, unsigned int flags )`

`virStorageVolWipe`: : pointer to storage volume : extra flags; not used yet, so callers should always pass 0

Ensure data previously on a volume is not accessible to future reads

Returns 0 on success, or -1 on error

#### 4.7.2.361 `int virStorageVolWipePattern ( virStorageVolPtr vol, unsigned int algorithm, unsigned int flags )`

`virStorageVolWipePattern`: : pointer to storage volume : one of `virStorageVolWipeAlgorithm` : future flags, use 0 for now

Similar to `virStorageVolWipe`, but one can choose between different wiping algorithms.

Returns 0 on success, or -1 on error.

#### 4.7.2.362 `int virStreamAbort ( virStreamPtr stream )`

`virStreamAbort`: : pointer to the stream object

Request that the in progress data transfer be cancelled abnormally before the end of the stream has been reached. For output streams this can be used to inform the driver that the stream is being terminated early. For input streams this can be used to inform the driver that it should stop sending data.

Returns 0 on success, -1 upon error

#### 4.7.2.363 `int virStreamEventAddCallback ( virStreamPtr stream, int events, virStreamEventCallback cb, void * opaque, virFreeCallback ff )`

`virStreamEventAddCallback`: : pointer to the stream object : set of events to monitor : callback to invoke when an event occurs : application defined data : callback to free data

Register a callback to be notified when a stream becomes writable, or readable. This is most commonly used in conjunction with non-blocking data streams to integrate into an event loop

Returns 0 on success, -1 upon error

#### 4.7.2.364 `int virStreamEventRemoveCallback ( virStreamPtr stream )`

`virStreamEventRemoveCallback`: : pointer to the stream object

Remove an event callback from the stream

Returns 0 on success, -1 on error

#### 4.7.2.365 `int virStreamEventUpdateCallback ( virStreamPtr stream, int events )`

`virStreamEventUpdateCallback`: : pointer to the stream object : set of events to monitor

Changes the set of events to monitor for a stream. This allows for event notification to be changed without having to unregister & register the callback completely. This method is guaranteed to succeed if a callback is already registered

Returns 0 on success, -1 if no callback is registered



**4.7.2.366 int virStreamFinish ( virStreamPtr stream )**

virStreamFinish: : pointer to the stream object

Indicate that there is no further data is to be transmitted on the stream. For output streams this should be called once all data has been written. For input streams this should be called once virStreamRecv returns end-of-file.

This method is a synchronization point for all asynchronous errors, so if this returns a success code the application can be sure that all data has been successfully processed.

Returns 0 on success, -1 upon error

**4.7.2.367 int virStreamFree ( virStreamPtr stream )**

virStreamFree: : pointer to the stream object

Decrement the reference count on a stream, releasing the stream object if the reference count has hit zero.

There must not be an active data transfer in progress when releasing the stream. If a stream needs to be disposed of prior to end of stream being reached, then the virStreamAbort function should be called first.

Returns 0 upon success, or -1 on error

**4.7.2.368 virStreamPtr virStreamNew ( virConnectPtr conn, unsigned int flags )**

virStreamNew: : pointer to the connection : bitwise-OR of virStreamFlags

Creates a new stream object which can be used to perform streamed I/O with other public API function.

When no longer needed, a stream object must be released with virStreamFree. If a data stream has been used, then the application must call virStreamFinish or virStreamAbort before free'ing to, in order to notify the driver of termination.

If a non-blocking data stream is required passed VIR\_STREAM\_NONBLOCK for flags, otherwise pass 0.

Returns the new stream, or NULL upon error

**4.7.2.369 int virStreamRecv ( virStreamPtr stream, char \* data, size\_t nbytes )**

virStreamRecv: : pointer to the stream object : buffer to read into from stream : size of buffer

Reads a series of bytes from the stream. This method may block the calling application for an arbitrary amount of time.

Errors are not guaranteed to be reported synchronously with the call, but may instead be delayed until a subsequent call.

An example using this with a hypothetical file download API looks like

```
virStreamPtr st = virStreamNew(conn, 0); int fd = open("demo.iso", O_WRONLY, 0600)
```

```
virConnectDownloadFile(conn, "demo.iso", st);
```

```
while (1) { char buf[1024]; int got = virStreamRecv(st, buf, 1024); if (got < 0) break; if (got == 0) { virStreamFinish(st); break; } int offset = 0; while (offset < got) { int sent = write(fd, buf+offset, got-offset) if (sent < 0) { virStreamAbort(st); goto done; } offset += sent; } if (virStreamFinish(st) < 0) ... report an error .... done: virStreamFree(st); close(fd);
```

Returns the number of bytes read, which may be less than requested.

Returns 0 when the end of the stream is reached, at which time the caller should invoke [virStreamFinish\(\)](#) to get confirmation of stream completion.

Returns -1 upon error, at which time the stream will be marked as aborted, and the caller should now release the stream with virStreamFree.

Returns -2 if there is no data pending to be read & the stream is marked as non-blocking.

#### 4.7.2.370 `int virStreamRecvAll ( virStreamPtr stream, virStreamSinkFunc handler, void * opaque )`

`virStreamRecvAll`: : pointer to the stream object : sink callback for writing data to application : application defined data

Receive the entire data stream, sending the data to the requested data sink. This is simply a convenient alternative to `virStreamRecv`, for apps that do blocking-I/O.

An example using this with a hypothetical file download API looks like

```
int mysink(virStreamPtr st, const char *buf, int nbytes, void *opaque) { int *fd = opaque;
return write(*fd, buf, nbytes); }
```

```
virStreamPtr st = virStreamNew(conn, 0); int fd = open("demo.iso", O_WRONLY)
```

```
virConnectUploadFile(conn, st); if (virStreamRecvAll(st, mysink, &fd) < 0) { ...report an error ... goto done; } if
(virStreamFinish(st) < 0) ...report an error... virStreamFree(st); close(fd);
```

Returns 0 if all the data was successfully received. The caller should invoke `virStreamFinish(st)` to flush the stream upon success and then `virStreamFree`

Returns -1 upon any error, with `virStreamAbort()` already having been called, so the caller need only call `virStreamFree()`

#### 4.7.2.371 `int virStreamRef ( virStreamPtr stream )`

`virStreamRef`: : pointer to the stream

Increment the reference count on the stream. For each additional call to this method, there shall be a corresponding call to `virStreamFree` to release the reference count, once the caller no longer needs the reference to this object.

Returns 0 in case of success, -1 in case of failure

#### 4.7.2.372 `int virStreamSend ( virStreamPtr stream, const char * data, size_t nbytes )`

`virStreamSend`: : pointer to the stream object : buffer to write to stream : size of buffer

Write a series of bytes to the stream. This method may block the calling application for an arbitrary amount of time. Once an application has finished sending data it should call `virStreamFinish` to wait for successful confirmation from the driver, or detect any error.

This method may not be used if a stream source has been registered.

Errors are not guaranteed to be reported synchronously with the call, but may instead be delayed until a subsequent call.

An example using this with a hypothetical file upload API looks like

```
virStreamPtr st = virStreamNew(conn, 0); int fd = open("demo.iso", O_RDONLY)
```

```
virConnectUploadFile(conn, "demo.iso", st);
```

```
while (1) { char buf[1024]; int got = read(fd, buf, 1024); if (got < 0) { virStreamAbort(st); break; } if (got == 0) {
virStreamFinish(st); break; } int offset = 0; while (offset < got) { int sent = virStreamSend(st, buf+offset, got-offset)
if (sent < 0) { virStreamAbort(st); goto done; } offset += sent; } } if (virStreamFinish(st) < 0) ... report an error ....
done: virStreamFree(st); close(fd);
```

Returns the number of bytes written, which may be less than requested.

Returns -1 upon error, at which time the stream will be marked as aborted, and the caller should now release the stream with `virStreamFree`.

Returns -2 if the outgoing transmit buffers are full & the stream is marked as non-blocking.

**4.7.2.373** `int virStreamSendAll ( virStreamPtr stream, virStreamSourceFunc handler, void * opaque )`

`virStreamSendAll`: : pointer to the stream object : source callback for reading data from application : application defined data

Send the entire data stream, reading the data from the requested data source. This is simply a convenient alternative to `virStreamSend`, for apps that do blocking-I/O.

An example using this with a hypothetical file upload API looks like

```
int mysource(virStreamPtr st, char *buf, int nbytes, void *opaque) { int *fd = opaque;
```

```
return read(*fd, buf, nbytes); }
```

```
virStreamPtr st = virStreamNew(conn, 0); int fd = open("demo.iso", O_RDONLY)
```

```
virConnectUploadFile(conn, st); if (virStreamSendAll(st, mysource, &fd) < 0) { ...report an error ... goto done; } if (virStreamFinish(st) < 0) ...report an error... virStreamFree(st); close(fd);
```

Returns 0 if all the data was successfully sent. The caller should invoke `virStreamFinish(st)` to flush the stream upon success and then `virStreamFree`

Returns -1 upon any error, with `virStreamAbort()` already having been called, so the caller need only call `virStreamFree()`

**4.7.2.374** `static int virTLSMutexDestroy ( void ** priv ) [static]`

**4.7.2.375** `static int virTLSMutexInit ( void ** priv ) [static]`

**4.7.2.376** `static int virTLSMutexLock ( void ** priv ) [static]`

**4.7.2.377** `static int virTLSMutexUnlock ( void ** priv ) [static]`

**4.7.2.378** `static int virTypedParameterValidateSet ( virConnectPtr conn, virTypedParameterPtr params, int nparams ) [static]`

### 4.7.3 Variable Documentation

**4.7.3.1** `int initialized = 0 [static]`

**4.7.3.2** `virConnectAuth virConnectAuthDefault [static]`

**Initial value:**

```
= {
    virConnectCredTypeDefault,
    sizeof(virConnectCredTypeDefault)/sizeof(int),
    virConnectAuthCallbackDefault,
    NULL,
}
```

**4.7.3.3** `virConnectAuthPtr virConnectAuthPtrDefault = &virConnectAuthDefault`

**4.7.3.4** `int virConnectCredTypeDefault[] [static]`

**Initial value:**

```
= {
    VIR_CRED_AUTHNAME,
    VIR_CRED_ECHOPROMPT,
    VIR_CRED_REALM,
    VIR_CRED_PASSPHRASE,
    VIR_CRED_NOECHOPROMPT,
}
```

```

    VIR_CRED_EXTERNAL,
}

```

**4.7.3.5** `virDeviceMonitorPtr virDeviceMonitorTab[MAX_DRIVERS]` [static]

**4.7.3.6** `int virDeviceMonitorTabCount = 0` [static]

**4.7.3.7** `virDriverPtr virDriverTab[MAX_DRIVERS]` [static]

**4.7.3.8** `int virDriverTabCount = 0` [static]

**4.7.3.9** `virInterfaceDriverPtr virInterfaceDriverTab[MAX_DRIVERS]` [static]

**4.7.3.10** `int virInterfaceDriverTabCount = 0` [static]

**4.7.3.11** `virNetworkDriverPtr virNetworkDriverTab[MAX_DRIVERS]` [static]

**4.7.3.12** `int virNetworkDriverTabCount = 0` [static]

**4.7.3.13** `virNWFilterDriverPtr virNWFilterDriverTab[MAX_DRIVERS]` [static]

**4.7.3.14** `int virNWFilterDriverTabCount = 0` [static]

**4.7.3.15** `virSecretDriverPtr virSecretDriverTab[MAX_DRIVERS]` [static]

**4.7.3.16** `int virSecretDriverTabCount = 0` [static]

**4.7.3.17** `virStorageDriverPtr virStorageDriverTab[MAX_DRIVERS]` [static]

**4.7.3.18** `int virStorageDriverTabCount = 0` [static]

**4.7.3.19** `struct gcry_thread_cbs virTLSThreadImpl` [static]

**Initial value:**

```

= {

    GCRY_THREAD_OPTION_PTHREAD,

    NULL,
    virTLSMutexInit,
    virTLSMutexDestroy,
    virTLSMutexLock,
    virTLSMutexUnlock,
    NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL
}

```

## 4.8 src/network/bridge\_driver.c File Reference

```
#include <config.h>
```

```
#include <sys/types.h>
#include <sys/poll.h>
#include <limits.h>
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/utsname.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>
#include <paths.h>
#include <pwd.h>
#include <sys/wait.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include "virterror_internal.h"
#include "datatypes.h"
#include "bridge_driver.h"
#include "network_conf.h"
#include "device_conf.h"
#include "driver.h"
#include "buf.h"
#include "virpidfile.h"
#include "util.h"
#include "memory.h"
#include "uuid.h"
#include "iptables.h"
#include "logging.h"
#include "configmake.h"
#include "virnetdev.h"
#include "pci.h"
#include "virnetdevbridge.h"
#include "virnetdevopenvswitch.h"
#include "virnetdevtap.h"
#include "virnetdevvportprofile.h"
#include "virdbus.h"
#include "virfile.h"
```

## Data Structures

- struct [network\\_driver](#)

## Macros

- #define [NETWORK\\_PID\\_DIR](#) LOCALSTATEDIR "/run/libvirt/network"
- #define [NETWORK\\_STATE\\_DIR](#) LOCALSTATEDIR "/lib/libvirt/network"
- #define [DNSMASQ\\_STATE\\_DIR](#) LOCALSTATEDIR "/lib/libvirt/dnsmasq"
- #define [RADVD\\_STATE\\_DIR](#) LOCALSTATEDIR "/lib/libvirt/radvd"
- #define [VIR\\_FROM\\_THIS](#) VIR\_FROM\_NETWORK
- #define [SYSCTL\\_PATH](#) "/proc/sys"
- #define [PROC\\_NET\\_ROUTE](#) "/proc/net/route"

## Functions

- static void [networkDriverLock](#) (struct [network\\_driver](#) \*driver)
- static void [networkDriverUnlock](#) (struct [network\\_driver](#) \*driver)
- static int [networkShutdown](#) (void)
- static int [networkStartNetwork](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkShutdownNetwork](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- [POL Mod](#) static int [networkStartNetworkVirtual](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- [POL Mod](#) static int [networkShutdownNetworkVirtual](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkStartNetworkExternal](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkShutdownNetworkExternal](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static void [networkReloadIptablesRules](#) (struct [network\\_driver](#) \*driver)
- static void [networkRefreshDaemons](#) (struct [network\\_driver](#) \*driver)
- static char \* [networkDnsmasqLeaseFileNameDefault](#) (const char \*netname)
- static char \* [networkRadvdPidfileBasename](#) (const char \*netname)
- static char \* [networkRadvdConfigFileName](#) (const char \*netname)
- static int [networkRemoveInactive](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) net)
- static char \* [networkBridgeDummyNicName](#) (const char \*brname)
- static void [networkFindActiveConfigs](#) (struct [network\\_driver](#) \*driver)
- static void [networkAutostartConfigs](#) (struct [network\\_driver](#) \*driver)
- static int [networkStartup](#) (int privileged)
- static int [networkReload](#) (void)
- static int [networkActive](#) (void)
- static int [networkKillDaemon](#) (pid\_t pid, const char \*daemonName, const char \*networkName)
- static int [networkBuildDnsmasqHostsfile](#) (dnsmasqContext \*dctx, [virNetworkIpDefPtr](#) ipdef, [virNetworkDNSDefPtr](#) dnsdef)
- static int [networkBuildDnsmasqArgv](#) ([virNetworkObjPtr](#) network, [virNetworkIpDefPtr](#) ipdef, const char \*pidfile, [virCommandPtr](#) cmd, dnsmasqContext \*dctx, dnsmasqCapsPtr caps ATTRIBUTE\_UNUSED)
- int [networkBuildDhcpDaemonCommandLine](#) ([virNetworkObjPtr](#) network, [virCommandPtr](#) \*cmdout, char \*pidfile, dnsmasqContext \*dctx, dnsmasqCapsPtr caps)
- static int [networkStartDhcpDaemon](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkRefreshDhcpDaemon](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkRestartDhcpDaemon](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkRadvdConfContents](#) ([virNetworkObjPtr](#) network, char \*\*configstr)
- static int [networkRadvdConfWrite](#) ([virNetworkObjPtr](#) network, char \*\*configFile)
- static int [networkStartRadvd](#) ([virNetworkObjPtr](#) network)
- static int [networkRefreshRadvd](#) (struct [network\\_driver](#) \*driver ATTRIBUTE\_UNUSED, [virNetworkObjPtr](#) network)
- static int [networkAddMasqueradingIptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network, [virNetworkIpDefPtr](#) ipdef)
- static void [networkRemoveMasqueradingIptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network, [virNetworkIpDefPtr](#) ipdef)
- static int [networkAddRoutingIptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network, [virNetworkIpDefPtr](#) ipdef)
- static void [networkRemoveRoutingIptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network, [virNetworkIpDefPtr](#) ipdef)
- static int [networkAddGenerallp6tablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static void [networkRemoveGenerallp6tablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkAddGenerallptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static void [networkRemoveGenerallptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkAddIpSpecificIptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network, [virNetworkIpDefPtr](#) ipdef)
- static void [networkRemoveIpSpecificIptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network, [virNetworkIpDefPtr](#) ipdef)

- static int [networkAddIptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static void [networkRemoveIptablesRules](#) (struct [network\\_driver](#) \*driver, [virNetworkObjPtr](#) network)
- static int [networkEnableIpForwarding](#) (bool enableIPv4, bool enableIPv6)
- static int [networkSetIPv6Sysctls](#) ([virNetworkObjPtr](#) network)
- static int [networkCheckRouteCollision](#) ([virNetworkObjPtr](#) network)
- static int [networkAddAddrToBridge](#) ([virNetworkObjPtr](#) network, [virNetworkIpDefPtr](#) ipdef)
- static int [networkStartNetworkExternal](#) (struct [network\\_driver](#) \*driver ATTRIBUTE\_UNUSED, [virNetworkObjPtr](#) network ATTRIBUTE\_UNUSED)
- static int [networkShutdownNetworkExternal](#) (struct [network\\_driver](#) \*driver ATTRIBUTE\_UNUSED, [virNetworkObjPtr](#) network ATTRIBUTE\_UNUSED)
- static [virNetworkPtr](#) [networkLookupByUUID](#) ([virConnectPtr](#) conn, const unsigned char \*uuid)
- static [virNetworkPtr](#) [networkLookupByName](#) ([virConnectPtr](#) conn, const char \*name)
- static [virDrvOpenStatus](#) [networkOpenNetwork](#) ([virConnectPtr](#) conn, [virConnectAuthPtr](#) auth ATTRIBUTE\_UNUSED, unsigned int flags)
- static int [networkCloseNetwork](#) ([virConnectPtr](#) conn)
- static int [networkNumNetworks](#) ([virConnectPtr](#) conn)
- static int [networkListNetworks](#) ([virConnectPtr](#) conn, char \*\*const names, int nnames)
- static int [networkNumDefinedNetworks](#) ([virConnectPtr](#) conn)
- static int [networkListDefinedNetworks](#) ([virConnectPtr](#) conn, char \*\*const names, int nnames)
- static int [networkListAllNetworks](#) ([virConnectPtr](#) conn, [virNetworkPtr](#) \*\*nets, unsigned int flags)
- static int [networkIsActive](#) ([virNetworkPtr](#) net)
- static int [networkIsPersistent](#) ([virNetworkPtr](#) net)
- **POL Mod** static int [networkValidate](#) ([virNetworkDefPtr](#) def)
- static [virNetworkPtr](#) [networkCreate](#) ([virConnectPtr](#) conn, const char \*xml)
- static [virNetworkPtr](#) [networkDefine](#) ([virConnectPtr](#) conn, const char \*xml)
- static int [networkUndefine](#) ([virNetworkPtr](#) net)
- static int [networkUpdate](#) ([virNetworkPtr](#) net, unsigned int command, unsigned int section, int parentIndex, const char \*xml, unsigned int flags)
- static int [networkStart](#) ([virNetworkPtr](#) net)
- static int [networkDestroy](#) ([virNetworkPtr](#) net)
- static char \* [networkGetXMLDesc](#) ([virNetworkPtr](#) net, unsigned int flags)
- static char \* [networkGetBridgeName](#) ([virNetworkPtr](#) net)
- **POL New** static char \* [networkGetBridgeType](#) ([virNetworkPtr](#) net)
- static int [networkGetAutostart](#) ([virNetworkPtr](#) net, int \*autostart)
- static int [networkSetAutostart](#) ([virNetworkPtr](#) net, int autostart)
- int [networkRegister](#) (void)
- static int [networkCreateInterfacePool](#) ([virNetworkDefPtr](#) netdef)
- **POL Mod** int [networkAllocateActualDevice](#) ([virDomainNetDefPtr](#) iface)
- int [networkNotifyActualDevice](#) ([virDomainNetDefPtr](#) iface)
- int [networkReleaseActualDevice](#) ([virDomainNetDefPtr](#) iface)
- int [networkGetNetworkAddress](#) (const char \*netname, char \*\*netaddr)

## Variables

- static struct [network\\_driver](#) \* [driverState](#) = NULL
- [networkDnsmasqLeaseFileNameFunc](#) [networkDnsmasqLeaseFileName](#)
- static [virNetworkDriver](#) [networkDriver](#)
- static [virStateDriver](#) [networkStateDriver](#)

## 4.8.1 Macro Definition Documentation

4.8.1.1 `#define DNSMASQ_STATE_DIR LOCALSTATEDIR "/lib/libvirt/dnsmasq"`

4.8.1.2 `#define NETWORK_PID_DIR LOCALSTATEDIR "/run/libvirt/network"`

4.8.1.3 `#define NETWORK_STATE_DIR LOCALSTATEDIR "/lib/libvirt/network"`

4.8.1.4 `#define PROC_NET_ROUTE "/proc/net/route"`

4.8.1.5 `#define RADVD_STATE_DIR LOCALSTATEDIR "/lib/libvirt/radvd"`

4.8.1.6 `#define SYSCTL_PATH "/proc/sys"`

4.8.1.7 `#define VIR_FROM_THIS VIR_FROM_NETWORK`

## 4.8.2 Function Documentation

4.8.2.1 `static int networkActive ( void ) [static]`

networkActive:

Checks if the QEmu daemon is active, i.e. has an active domain or an active network

Returns 1 if active, 0 otherwise

4.8.2.2 `static int networkAddAddrToBridge ( virNetworkObjPtr network, virNetworkIpfDefPtr ipdef ) [static]`

4.8.2.3 `static int networkAddGenerallptablesRules ( struct network_driver * driver, virNetworkObjPtr network ) [static]`

4.8.2.4 `static int networkAddGenerallptablesRules ( struct network_driver * driver, virNetworkObjPtr network ) [static]`

4.8.2.5 `static int networkAddIpfSpecificlptablesRules ( struct network_driver * driver, virNetworkObjPtr network, virNetworkIpfDefPtr ipdef ) [static]`

4.8.2.6 `static int networkAddIptablesRules ( struct network_driver * driver, virNetworkObjPtr network ) [static]`

4.8.2.7 `static int networkAddMasqueradingIptablesRules ( struct network_driver * driver, virNetworkObjPtr network, virNetworkIpfDefPtr ipdef ) [static]`

4.8.2.8 `static int networkAddRoutingIptablesRules ( struct network_driver * driver, virNetworkObjPtr network, virNetworkIpfDefPtr ipdef ) [static]`

4.8.2.9 **POL Mod** `int networkAllocateActualDevice ( virDomainNetDefPtr iface )`

4.8.2.10 `static void networkAutostartConfigs ( struct network_driver * driver ) [static]`

4.8.2.11 `static char* networkBridgeDummyNicName ( const char * brname ) [static]`

4.8.2.12 `int networkBuildDhcpDaemonCommandLine ( virNetworkObjPtr network, virCommandPtr * cmdout, char * pidfile, dnsmasqContext * dctx, dnsmasqCapsPtr caps )`

4.8.2.13 `static int networkBuildDnsmasqArgv ( virNetworkObjPtr network, virNetworkIpfDefPtr ipdef, const char * pidfile, virCommandPtr cmd, dnsmasqContext * dctx, dnsmasqCapsPtr caps ATTRIBUTE_UNUSED ) [static]`



- 4.8.2.14 `static int networkBuildDnsmasqHostsfile ( dnsmasqContext * dctx, virNetworkIpDefPtr ipdef, virNetworkDNSDefPtr dnsdef ) [static]`
- 4.8.2.15 `static int networkCheckRouteCollision ( virNetworkObjPtr network ) [static]`
- 4.8.2.16 `static int networkCloseNetwork ( virConnectPtr conn ) [static]`
- 4.8.2.17 `static virNetworkPtr networkCreate ( virConnectPtr conn, const char * xml ) [static]`
- 4.8.2.18 `static int networkCreateInterfacePool ( virNetworkDefPtr netdef ) [static]`
- 4.8.2.19 `static virNetworkPtr networkDefine ( virConnectPtr conn, const char * xml ) [static]`
- 4.8.2.20 `static int networkDestroy ( virNetworkPtr net ) [static]`
- 4.8.2.21 `static char* networkDnsmasqLeaseFileNameDefault ( const char * netname ) [static]`
- 4.8.2.22 `static void networkDriverLock ( struct network_driver * driver ) [static]`
- 4.8.2.23 `static void networkDriverUnlock ( struct network_driver * driver ) [static]`
- 4.8.2.24 `static int networkEnableIpForwarding ( bool enableIPv4, bool enableIPv6 ) [static]`
- 4.8.2.25 `static void networkFindActiveConfigs ( struct network_driver * driver ) [static]`
- 4.8.2.26 `static int networkGetAutostart ( virNetworkPtr net, int * autostart ) [static]`
- 4.8.2.27 `static char* networkGetBridgeName ( virNetworkPtr net ) [static]`
- 4.8.2.28 **POL New** `static char* networkGetBridgeType ( virNetworkPtr net ) [static]`

networkGetBridgeType

#### Parameters

<i>net</i>	pointer to virNetwork structure
------------	---------------------------------

Return the bridge type of a virtual network

- 4.8.2.29 `int networkGetNetworkAddress ( const char * netname, char ** netaddr )`
- 4.8.2.30 `static char* networkGetXMLDesc ( virNetworkPtr net, unsigned int flags ) [static]`
- 4.8.2.31 `static int networkIsActive ( virNetworkPtr net ) [static]`
- 4.8.2.32 `static int networkIsPersistent ( virNetworkPtr net ) [static]`
- 4.8.2.33 `static int networkKillDaemon ( pid_t pid, const char * daemonName, const char * networkName ) [static]`
- 4.8.2.34 `static int networkListAllNetworks ( virConnectPtr conn, virNetworkPtr ** nets, unsigned int flags ) [static]`
- 4.8.2.35 `static int networkListDefinedNetworks ( virConnectPtr conn, char **const names, int nnames ) [static]`
- 4.8.2.36 `static int networkListNetworks ( virConnectPtr conn, char **const names, int nnames ) [static]`

- 4.8.2.37 `static virNetworkPtr networkLookupByName ( virConnectPtr conn, const char * name ) [static]`
- 4.8.2.38 `static virNetworkPtr networkLookupByUUID ( virConnectPtr conn, const unsigned char * uuid ) [static]`
- 4.8.2.39 `int networkNotifyActualDevice ( virDomainNetDefPtr iface )`
- 4.8.2.40 `static int networkNumDefinedNetworks ( virConnectPtr conn ) [static]`
- 4.8.2.41 `static int networkNumNetworks ( virConnectPtr conn ) [static]`
- 4.8.2.42 `static virDrvOpenStatus networkOpenNetwork ( virConnectPtr conn, virConnectAuthPtr auth  
ATTRIBUTE_UNUSED, unsigned int flags ) [static]`
- 4.8.2.43 `static int networkRadvdConfContents ( virNetworkObjPtr network, char ** configstr ) [static]`
- 4.8.2.44 `static char* networkRadvdConfigFileName ( const char * netname ) [static]`
- 4.8.2.45 `static int networkRadvdConfWrite ( virNetworkObjPtr network, char ** configFile ) [static]`
- 4.8.2.46 `static char* networkRadvdPidfileBasename ( const char * netname ) [static]`
- 4.8.2.47 `static void networkRefreshDaemons ( struct network_driver * driver ) [static]`
- 4.8.2.48 `static int networkRefreshDhcpDaemon ( struct network_driver * driver, virNetworkObjPtr network )  
[static]`
- 4.8.2.49 `static int networkRefreshRadvd ( struct network_driver * driver ATTRIBUTE_UNUSED, virNetworkObjPtr  
network ) [static]`
- 4.8.2.50 `int networkRegister ( void )`
- 4.8.2.51 `int networkReleaseActualDevice ( virDomainNetDefPtr iface )`
- 4.8.2.52 `static int networkReload ( void ) [static]`

networkReload:

Function to restart the QEmu daemon, it will recheck the configuration files and update its state and the networking

- 4.8.2.53 `static void networkReloadIptablesRules ( struct network_driver * driver ) [static]`
- 4.8.2.54 `static void networkRemoveGenerallp6tablesRules ( struct network_driver * driver, virNetworkObjPtr network )  
[static]`
- 4.8.2.55 `static void networkRemoveGenerallptablesRules ( struct network_driver * driver, virNetworkObjPtr network )  
[static]`
- 4.8.2.56 `static int networkRemovelInactive ( struct network_driver * driver, virNetworkObjPtr net ) [static]`
- 4.8.2.57 `static void networkRemoveIptablesRules ( struct network_driver * driver, virNetworkObjPtr network,  
virNetworkIptDefPtr iptdef ) [static]`
- 4.8.2.58 `static void networkRemoveIptablesRules ( struct network_driver * driver, virNetworkObjPtr network )  
[static]`

- 4.8.2.59 `static void networkRemoveMasqueradingIptablesRules ( struct network_driver * driver, virNetworkObjPtr network, virNetworkIpfDefPtr ipdef ) [static]`
- 4.8.2.60 `static void networkRemoveRoutingIptablesRules ( struct network_driver * driver, virNetworkObjPtr network, virNetworkIpfDefPtr ipdef ) [static]`
- 4.8.2.61 `static int networkRestartDhcpDaemon ( struct network_driver * driver, virNetworkObjPtr network ) [static]`
- 4.8.2.62 `static int networkSetAutostart ( virNetworkPtr net, int autostart ) [static]`
- 4.8.2.63 `static int networkSetIPv6Sysctls ( virNetworkObjPtr network ) [static]`
- 4.8.2.64 `static int networkShutdown ( void ) [static]`

networkShutdown:

Shutdown the QEmu daemon, it will stop all active domains and networks

- 4.8.2.65 `static int networkShutdownNetwork ( struct network_driver * driver, virNetworkObjPtr network ) [static]`
- 4.8.2.66 `static int networkShutdownNetworkExternal ( struct network_driver * driver, virNetworkObjPtr network ) [static]`
- 4.8.2.67 `static int networkShutdownNetworkExternal ( struct network_driver * driver ATTRIBUTE_UNUSED, virNetworkObjPtr network ATTRIBUTE_UNUSED ) [static]`
- 4.8.2.68 **POL Mod** `static int networkShutdownNetworkVirtual ( struct network_driver * driver, virNetworkObjPtr network ) [static]`

networkShutdownNetworkVirtual

Parameters

<i>driver</i>	pointer to <a href="#">network_driver</a> structure
<i>network</i>	pointer to virNetworkObj structure

Shutdown a virtual network.

POL modification:

- Added support for Open vSwitch switches

- 4.8.2.69 `static int networkStart ( virNetworkPtr net ) [static]`
- 4.8.2.70 `static int networkStartDhcpDaemon ( struct network_driver * driver, virNetworkObjPtr network ) [static]`
- 4.8.2.71 `static int networkStartNetwork ( struct network_driver * driver, virNetworkObjPtr network ) [static]`
- 4.8.2.72 `static int networkStartNetworkExternal ( struct network_driver * driver, virNetworkObjPtr network ) [static]`
- 4.8.2.73 `static int networkStartNetworkExternal ( struct network_driver * driver ATTRIBUTE_UNUSED, virNetworkObjPtr network ATTRIBUTE_UNUSED ) [static]`

4.8.2.74 **POL Mod** static int networkStartNetworkVirtual ( struct network\_driver \* *driver*, virNetworkObjPtr *network* )  
[static]

networkStartNetworkVirtual

Parameters

<i>driver</i>	pointer to <a href="#">network_driver</a> structure
<i>network</i>	pointer to virNetworkObj structure

Start a virtual network

POL modification:

- Added support for Open vSwitch switches
- Added support for GRE tunnels

4.8.2.75 static int networkStartRadvd ( virNetworkObjPtr *network* ) [static]

4.8.2.76 static int networkStartup ( int *privileged* ) [static]

networkStartup:

Initialization function for the QEmu daemon

4.8.2.77 static int networkUndefine ( virNetworkPtr *net* ) [static]

4.8.2.78 static int networkUpdate ( virNetworkPtr *net*, unsigned int *command*, unsigned int *section*, int *parentIndex*, const char \* *xml*, unsigned int *flags* ) [static]

4.8.2.79 **POL Mod** static int networkValidate ( virNetworkDefPtr *def* ) [static]

networkValidate

Parameters

<i>def</i>	pointer to virNetworkDef structure
------------	------------------------------------

Validate a virtual network configuration.

POL modification:

- Allow use of VLANs if the network forward mode is also: nat, route, none

### 4.8.3 Variable Documentation

4.8.3.1 struct network\_driver\* driverState = NULL [static]

4.8.3.2 networkDnsmasqLeaseFileNameFunc networkDnsmasqLeaseFileName

Initial value:

```
=
networkDnsmasqLeaseFileNameDefault
```

## 4.8.3.3 virNetworkDriver networkDriver [static]

**Initial value:**

```
= {
    "Network",
    .open = networkOpenNetwork,
    .close = networkCloseNetwork,
    .numOfNetworks = networkNumNetworks,
    .listNetworks = networkListNetworks,
    .numOfDefinedNetworks = networkNumDefinedNetworks,
    .listDefinedNetworks = networkListDefinedNetworks,
    .listAllNetworks = networkListAllNetworks,
    .networkLookupByUUID = networkLookupByUUID,
    .networkLookupByName = networkLookupByName,
    .networkCreateXML = networkCreate,
    .networkDefineXML = networkDefine,
    .networkUndefine = networkUndefine,
    .networkUpdate = networkUpdate,
    .networkCreate = networkStart,
    .networkDestroy = networkDestroy,
    .networkGetXMLDesc = networkGetXMLDesc,
    .networkGetBridgeName = networkGetBridgeName,
    .networkGetBridgeType = networkGetBridgeType,
    .networkGetAutostart = networkGetAutostart,
    .networkSetAutostart = networkSetAutostart,
    .networkIsActive = networkIsActive,
    .networkIsPersistent = networkIsPersistent,
}
```

## 4.8.3.4 virStateDriver networkStateDriver [static]

**Initial value:**

```
= {
    "Network",
    networkStartup,
    networkShutdown,
    networkReload,
    networkActive,
}
```

## 4.9 src/qemu/qemu\_command.c File Reference

```
#include <config.h>
#include "qemu_command.h"
#include "qemu_hostdev.h"
#include "qemu_capabilities.h"
#include "qemu_bridge_filter.h"
#include "cpu/cpu.h"
#include "memory.h"
#include "logging.h"
#include "virterror_internal.h"
#include "util.h"
#include "virfile.h"
#include "uuid.h"
#include "c-ctype.h"
#include "domain_nwfilter.h"
#include "domain_audit.h"
#include "domain_conf.h"
#include "network/bridge_driver.h"
#include "virnetdevtap.h"
#include "base64.h"
#include <sys/utsname.h>
#include <sys/stat.h>
#include <fcntl.h>
```

## Data Structures

- [struct \\_qemuDomainPCIAddressSet](#)

## Macros

- [#define VIR\\_FROM\\_THIS](#) [VIR\\_FROM\\_QEMU](#)
- [#define QEMU\\_PCI\\_ADDRESS\\_LAST\\_SLOT](#) 31
- [#define QEMU\\_PCI\\_ADDRESS\\_LAST\\_FUNCTION](#) 8
- [#define IS\\_USB2\\_CONTROLLER](#)(ctrl)
- [#define QEMU\\_SERIAL\\_PARAM\\_ACCEPTED\\_CHARS](#) "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-\_"
- [#define WANT\\_VALUE](#)()

## Functions

- [VIR\\_ENUM\\_IMPL](#) (virDomainDiskQEMUBus, [VIR\\_DOMAIN\\_DISK\\_BUS\\_LAST](#), "ide", "floppy", "scsi", "virtio", "xen", "usb", "uml", "s")
- int [qemuPhysIfaceConnect](#) (virDomainDefPtr def, struct qemud\_driver \*driver, [virDomainNetDefPtr](#) net, qemuCapsPtr caps, enum virNetDevVPortProfileOp vmop)
- [POL Mod](#) int [qemuNetworkIfaceConnect](#) (virDomainDefPtr def, [virConnectPtr](#) conn, struct qemud\_driver \*driver, [virDomainNetDefPtr](#) net, qemuCapsPtr caps)
- int [qemuOpenVhostNet](#) (virDomainDefPtr def, [virDomainNetDefPtr](#) net, qemuCapsPtr caps, int \*vhostfd)
- static int [qemuDomainDeviceAliasIndex](#) (virDomainDeviceInfoPtr info, const char \*prefix)
- int [qemuDomainNetVLAN](#) (virDomainNetDefPtr def)
- static int [qemuAssignDeviceDiskAliasLegacy](#) (virDomainDiskDefPtr disk)
- char \* [qemuDeviceDriveHostAlias](#) (virDomainDiskDefPtr disk, qemuCapsPtr caps)
- static int [qemuAssignDeviceDiskAliasFixed](#) (virDomainDiskDefPtr disk)
- static int [qemuSetScsiControllerModel](#) (virDomainDefPtr def, qemuCapsPtr caps, int \*model)
- static int [qemuAssignDeviceDiskAliasCustom](#) (virDomainDefPtr def, [virDomainDiskDefPtr](#) disk, qemuCapsPtr caps)
- int [qemuAssignDeviceDiskAlias](#) (virDomainDefPtr vmdef, [virDomainDiskDefPtr](#) def, qemuCapsPtr caps)
- int [qemuAssignDeviceNetAlias](#) (virDomainDefPtr def, [virDomainNetDefPtr](#) net, int idx)
- int [qemuAssignDeviceHostdevAlias](#) (virDomainDefPtr def, [virDomainHostdevDefPtr](#) hostdev, int idx)
- int [qemuAssignDeviceRedirdevAlias](#) (virDomainDefPtr def, [virDomainRedirdevDefPtr](#) redirdev, int idx)
- int [qemuAssignDeviceControllerAlias](#) (virDomainControllerDefPtr controller)
- int [qemuAssignDeviceAliases](#) (virDomainDefPtr def, qemuCapsPtr caps)
- static void [qemuDomainPrimeS390VirtioDevices](#) (virDomainDefPtr def, enum [virDomainDeviceAddressType](#) type)
- static int [qemuDomainAssignS390Addresses](#) (virDomainDefPtr def, qemuCapsPtr caps)
- static int [qemuSpaprVIOFindByReg](#) (virDomainDefPtr def, ATTRIBUTE\_UNUSED, [virDomainDeviceDefPtr](#) device, ATTRIBUTE\_UNUSED, [virDomainDeviceInfoPtr](#) info, void \*opaque)
- static int [qemuAssignSpaprVIOAddress](#) (virDomainDefPtr def, [virDomainDeviceInfoPtr](#) info, unsigned long long default\_reg)
- int [qemuDomainAssignSpaprVIOAddresses](#) (virDomainDefPtr def, qemuCapsPtr caps)
- static char \* [qemuPCIAddressAsString](#) (virDomainDeviceInfoPtr dev)
- static int [qemuCollectPCIAddress](#) (virDomainDefPtr def, ATTRIBUTE\_UNUSED, [virDomainDeviceDefPtr](#) device, [virDomainDeviceInfoPtr](#) info, void \*opaque)
- int [qemuDomainAssignPCIAddresses](#) (virDomainDefPtr def, qemuCapsPtr caps, [virDomainObjPtr](#) obj)
- int [qemuDomainAssignAddresses](#) (virDomainDefPtr def, qemuCapsPtr caps, [virDomainObjPtr](#) obj)
- static void [qemuDomainPCIAddressSetFreeEntry](#) (void \*payload, const void \*name, ATTRIBUTE\_UNUSED)
- [qemuDomainPCIAddressSetPtr](#) [qemuDomainPCIAddressSetCreate](#) (virDomainDefPtr def)

- static int [qemuDomainPCIAddressCheckSlot](#) (qemuDomainPCIAddressSetPtr addr, [virDomainDeviceInfoPtr](#) dev)
- int [qemuDomainPCIAddressReserveAddr](#) (qemuDomainPCIAddressSetPtr addr, [virDomainDeviceInfoPtr](#) dev)
- int [qemuDomainPCIAddressReserveFunction](#) (qemuDomainPCIAddressSetPtr addr, int slot, int function)
- int [qemuDomainPCIAddressReserveSlot](#) (qemuDomainPCIAddressSetPtr addr, int slot)
- int [qemuDomainPCIAddressEnsureAddr](#) (qemuDomainPCIAddressSetPtr addr, [virDomainDeviceInfoPtr](#) dev)
- int [qemuDomainPCIAddressReleaseAddr](#) (qemuDomainPCIAddressSetPtr addr, [virDomainDeviceInfoPtr](#) dev)
- int [qemuDomainPCIAddressReleaseFunction](#) (qemuDomainPCIAddressSetPtr addr, int slot, int function)
- int [qemuDomainPCIAddressReleaseSlot](#) (qemuDomainPCIAddressSetPtr addr, int slot)
- void [qemuDomainPCIAddressSetFree](#) (qemuDomainPCIAddressSetPtr addr)
- static int [qemuDomainPCIAddressGetNextSlot](#) (qemuDomainPCIAddressSetPtr addr)
- int [qemuDomainPCIAddressSetNextAddr](#) (qemuDomainPCIAddressSetPtr addr, [virDomainDeviceInfoPtr](#) dev)
- int [qemuAssignDevicePCISlots](#) ([virDomainDefPtr](#) def, qemuDomainPCIAddressSetPtr addr)
- static void [qemuUsbId](#) ([virBufferPtr](#) buf, int idx)
- static int [qemuBuildDeviceAddressStr](#) ([virBufferPtr](#) buf, [virDomainDeviceInfoPtr](#) info, qemuCapsPtr caps)
- static int [qemuBuildRomStr](#) ([virBufferPtr](#) buf, [virDomainDeviceInfoPtr](#) info, qemuCapsPtr caps)
- static int [qemuBuildIoEventFdStr](#) ([virBufferPtr](#) buf, enum [virDomainIoEventFd](#) use, qemuCapsPtr caps)
- static int [qemuSafeSerialParamValue](#) (const char \*value)
- static int [qemuBuildRBDString](#) ([virConnectPtr](#) conn, [virDomainDiskDefPtr](#) disk, [virBufferPtr](#) opt)
- static int [qemuAddRBDHost](#) ([virDomainDiskDefPtr](#) disk, char \*hostport)
- static int [qemuParseRBDString](#) ([virDomainDiskDefPtr](#) disk)
- char \* [qemuBuildDriveStr](#) ([virConnectPtr](#) conn ATTRIBUTE\_UNUSED, [virDomainDiskDefPtr](#) disk, bool bootable, qemuCapsPtr caps)
- char \* [qemuBuildDriveDevStr](#) ([virDomainDefPtr](#) def, [virDomainDiskDefPtr](#) disk, int bootindex, qemuCapsPtr caps)
- char \* [qemuBuildFSStr](#) ([virDomainFSDefPtr](#) fs, qemuCapsPtr caps ATTRIBUTE\_UNUSED)
- char \* [qemuBuildFSDevStr](#) ([virDomainFSDefPtr](#) fs, qemuCapsPtr caps)
- static int [qemuControllerModelUSBToCaps](#) (int model)
- static int [qemuBuildUSBControllerDevStr](#) ([virDomainDefPtr](#) domainDef, [virDomainControllerDefPtr](#) def, qemuCapsPtr caps, [virBuffer](#) \*buf)
- char \* [qemuBuildControllerDevStr](#) ([virDomainDefPtr](#) domainDef, [virDomainControllerDefPtr](#) def, qemuCapsPtr caps, int \*nusbcontroller)
- char \* [qemuBuildNicStr](#) ([virDomainNetDefPtr](#) net, const char \*prefix, int vlan)
- char \* [qemuBuildNicDevStr](#) ([virDomainNetDefPtr](#) net, int vlan, int bootindex, qemuCapsPtr caps)
- char \* [qemuBuildHostNetStr](#) ([virDomainNetDefPtr](#) net, struct [qemu\\_driver](#) \*driver, qemuCapsPtr caps, char type\_sep, int vlan, const char \*tapfd, const char \*vhostfd)
- char \* [qemuBuildWatchdogDevStr](#) ([virDomainWatchdogDefPtr](#) dev, qemuCapsPtr caps)
- char \* [qemuBuildMemballoonDevStr](#) ([virDomainMemballoonDefPtr](#) dev, qemuCapsPtr caps)
- char \* [qemuBuildUSBInputDevStr](#) ([virDomainInputDefPtr](#) dev, qemuCapsPtr caps)
- char \* [qemuBuildSoundDevStr](#) ([virDomainSoundDefPtr](#) sound, qemuCapsPtr caps)
- static int [qemuSoundCodecTypeToCaps](#) (int type)
- static char \* [qemuBuildSoundCodecStr](#) ([virDomainSoundDefPtr](#) sound, [virDomainSoundCodecDefPtr](#) codec, qemuCapsPtr caps)
- static char \* [qemuBuildVideoDevStr](#) ([virDomainVideoDefPtr](#) video, qemuCapsPtr caps)
- int [qemuOpenPCIConfig](#) ([virDomainHostdevDefPtr](#) dev)
- char \* [qemuBuildPCIHostdevDevStr](#) ([virDomainHostdevDefPtr](#) dev, const char \*configfd, qemuCapsPtr caps)
- char \* [qemuBuildPCIHostdevPCIDevStr](#) ([virDomainHostdevDefPtr](#) dev)
- char \* [qemuBuildRedirdevDevStr](#) ([virDomainDefPtr](#) def, [virDomainRedirdevDefPtr](#) dev, qemuCapsPtr caps)
- char \* [qemuBuildUSBHostdevDevStr](#) ([virDomainHostdevDefPtr](#) dev, qemuCapsPtr caps)
- char \* [qemuBuildHubDevStr](#) ([virDomainHubDefPtr](#) dev, qemuCapsPtr caps)
- char \* [qemuBuildUSBHostdevUsbDevStr](#) ([virDomainHostdevDefPtr](#) dev)

- static char \* [qemuBuildChrChardevStr](#) (virDomainChrSourceDefPtr dev, const char \*alias, qemuCapsPtr caps)
- static char \* [qemuBuildChrArgStr](#) (virDomainChrSourceDefPtr dev, const char \*prefix)
- static char \* [qemuBuildVirtioSerialPortDevStr](#) (virDomainChrDefPtr dev, qemuCapsPtr caps)
- static char \* [qemuBuildSmbiosBiosStr](#) (virSysinfoDefPtr def)
- static char \* [qemuBuildSmbiosSystemStr](#) (virSysinfoDefPtr def, bool skip\_uuid)
- static char \* [qemuBuildClockArgStr](#) (virDomainClockDefPtr def)
- static int [qemuBuildCpuArgStr](#) (const struct qemu\_driver \*driver, const virDomainDefPtr def, const char \*emulator, qemuCapsPtr caps, const struct utsname \*ut, char \*\*opt, bool \*hasHwVirt, bool migrating)
- static int [qemuBuildMachineArgStr](#) (virCommandPtr cmd, const virDomainDefPtr def, qemuCapsPtr caps)
- static char \* [qemuBuildSmpArgStr](#) (const virDomainDefPtr def, qemuCapsPtr caps)
- static int [qemuBuildNumaArgStr](#) (const virDomainDefPtr def, virCommandPtr cmd)
- virCommandPtr [qemuBuildCommandLine](#) (virConnectPtr conn, struct qemu\_driver \*driver, virDomainDefPtr def, virDomainChrSourceDefPtr monitor\_chr, bool monitor\_json, qemuCapsPtr caps, const char \*migrateFrom, int migrateFd, virDomainSnapshotObjPtr snapshot, enum virNetDevVPortProfileOp vmop)
- char \* [qemuBuildChrDeviceStr](#) (virDomainChrDefPtr serial, qemuCapsPtr caps, char \*os\_arch, char \*machine)
- static int [qemuStringToArgvEnv](#) (const char \*args, const char \*\*\*retenv, const char \*\*\*retargv)
- static const char \* [qemuFindEnv](#) (const char \*\*progenv, const char \*name)
- int [qemuParseKeywords](#) (const char \*str, char \*\*\*retkeywords, char \*\*\*retvalues, int allowEmptyValue)
- static virDomainDiskDefPtr [qemuParseCommandLineDisk](#) (virCapsPtr caps, const char \*val, int nvirtiodisk, bool old\_style\_ceph\_args)
- static const char \* [qemuFindNICForVLAN](#) (int nnics, const char \*\*nics, int wantvlan)
- static virDomainNetDefPtr [qemuParseCommandLineNet](#) (virCapsPtr caps, const char \*val, int nnics, const char \*\*nics)
- static virDomainHostdevDefPtr [qemuParseCommandLinePCI](#) (const char \*val)
- static virDomainHostdevDefPtr [qemuParseCommandLineUSB](#) (const char \*val)
- static int [qemuParseCommandLineChr](#) (virDomainChrSourceDefPtr source, const char \*val)
- static virCPUDefPtr [qemuInitGuestCPU](#) (virDomainDefPtr dom)
- static int [qemuParseCommandLineCPU](#) (virDomainDefPtr dom, const char \*val)
- static int [qemuParseCommandLineSmp](#) (virDomainDefPtr dom, const char \*val)
- static void [qemuParseCommandLineBootDevs](#) (virDomainDefPtr def, const char \*str)
- virDomainDefPtr [qemuParseCommandLine](#) (virCapsPtr caps, const char \*\*progenv, const char \*\*progargv, char \*\*pidfile, virDomainChrSourceDefPtr \*monConfig, bool \*monJSON)
- virDomainDefPtr [qemuParseCommandLineString](#) (virCapsPtr caps, const char \*args, char \*\*pidfile, virDomainChrSourceDefPtr \*monConfig, bool \*monJSON)
- static int [qemuParseProcFileStrings](#) (int pid\_value, const char \*name, const char \*\*\*list)
- virDomainDefPtr [qemuParseCommandLinePid](#) (virCapsPtr caps, pid\_t pid, char \*\*pidfile, virDomainChrSourceDefPtr \*monConfig, bool \*monJSON)

## 4.9.1 Macro Definition Documentation

### 4.9.1.1 #define IS\_USB2\_CONTROLLER( ctrl )

Value:

```
((ctrl)->type == VIR_DOMAIN_CONTROLLER_TYPE_USB) && \
((ctrl)->model == VIR_DOMAIN_CONTROLLER_MODEL_USB_ICH9_EHCI1
|| \
(ctrl)->model == VIR_DOMAIN_CONTROLLER_MODEL_USB_ICH9_UHCI1
|| \
(ctrl)->model == VIR_DOMAIN_CONTROLLER_MODEL_USB_ICH9_UHCI2
|| \
(ctrl)->model == VIR_DOMAIN_CONTROLLER_MODEL_USB_ICH9_UHCI3
))
```



4.9.1.2 `#define QEMU_PCI_ADDRESS_LAST_FUNCTION 8`

4.9.1.3 `#define QEMU_PCI_ADDRESS_LAST_SLOT 31`

4.9.1.4 `#define QEMU_SERIAL_PARAM_ACCEPTED_CHARS "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_"`

4.9.1.5 `#define VIR_FROM_THIS VIR_FROM_QEMU`

4.9.1.6 `#define WANT_VALUE( )`

**Value:**

```
const char *val = progargv[++i];
if (!val) {
    virReportError(VIR_ERR_INTERNAL_ERROR,
                  _("missing value for %s argument"), arg);
    goto error;
}
```

## 4.9.2 Function Documentation

4.9.2.1 `static int qemuAddRBDHost ( virDomainDiskDefPtr disk, char * hostport ) [static]`

4.9.2.2 `int qemuAssignDeviceAliases ( virDomainDefPtr def, qemuCapsPtr caps )`

4.9.2.3 `int qemuAssignDeviceControllerAlias ( virDomainControllerDefPtr controller )`

4.9.2.4 `int qemuAssignDeviceDiskAlias ( virDomainDefPtr vmdef, virDomainDiskDefPtr def, qemuCapsPtr caps )`

4.9.2.5 `static int qemuAssignDeviceDiskAliasCustom ( virDomainDefPtr def, virDomainDiskDefPtr disk, qemuCapsPtr caps ) [static]`

4.9.2.6 `static int qemuAssignDeviceDiskAliasFixed ( virDomainDiskDefPtr disk ) [static]`

4.9.2.7 `static int qemuAssignDeviceDiskAliasLegacy ( virDomainDiskDefPtr disk ) [static]`

4.9.2.8 `int qemuAssignDeviceHostdevAlias ( virDomainDefPtr def, virDomainHostdevDefPtr hostdev, int idx )`

4.9.2.9 `int qemuAssignDeviceNetAlias ( virDomainDefPtr def, virDomainNetDefPtr net, int idx )`

4.9.2.10 `int qemuAssignDevicePCISlots ( virDomainDefPtr def, qemuDomainPCIAddressSetPtr addrs )`

4.9.2.11 `int qemuAssignDeviceRedirdevAlias ( virDomainDefPtr def, virDomainRedirdevDefPtr redirdev, int idx )`

4.9.2.12 `static int qemuAssignSpaprVIOAddress ( virDomainDefPtr def, virDomainDeviceInfoPtr info, unsigned long long default_reg ) [static]`

4.9.2.13 `static char* qemuBuildChrArgStr ( virDomainChrSourceDefPtr dev, const char * prefix ) [static]`

4.9.2.14 `static char* qemuBuildChrChardevStr ( virDomainChrSourceDefPtr dev, const char * alias, qemuCapsPtr caps ) [static]`

4.9.2.15 `char* qemuBuildChrDeviceStr ( virDomainChrDefPtr serial, qemuCapsPtr caps, char * os_arch, char * machine )`

4.9.2.16 `static char* qemuBuildClockArgStr ( virDomainClockDefPtr def ) [static]`

- 4.9.2.17 `virCommandPtr qemuBuildCommandLine ( virConnectPtr conn, struct qemud_driver * driver, virDomainDefPtr def, virDomainChrSourceDefPtr monitor_chr, bool monitor_json, qemuCapsPtr caps, const char * migrateFrom, int migrateFd, virDomainSnapshotObjPtr snapshot, enum virNetDevVPortProfileOp vmop )`
- 4.9.2.18 `char* qemuBuildControllerDevStr ( virDomainDefPtr domainDef, virDomainControllerDefPtr def, qemuCapsPtr caps, int * nusbcontroller )`
- 4.9.2.19 `static int qemuBuildCpuArgStr ( const struct qemud_driver * driver, const virDomainDefPtr def, const char * emulator, qemuCapsPtr caps, const struct utsname * ut, char ** opt, bool * hasHwVirt, bool migrating ) [static]`
- 4.9.2.20 `static int qemuBuildDeviceAddressStr ( virBufferPtr buf, virDomainDeviceInfoPtr info, qemuCapsPtr caps ) [static]`
- 4.9.2.21 `char* qemuBuildDriveDevStr ( virDomainDefPtr def, virDomainDiskDefPtr disk, int bootindex, qemuCapsPtr caps )`
- 4.9.2.22 `char* qemuBuildDriveStr ( virConnectPtr conn ATTRIBUTE_UNUSED, virDomainDiskDefPtr disk, bool bootable, qemuCapsPtr caps )`
- 4.9.2.23 `char* qemuBuildFSDevStr ( virDomainFSDefPtr fs, qemuCapsPtr caps )`
- 4.9.2.24 `char* qemuBuildFSStr ( virDomainFSDefPtr fs, qemuCapsPtr caps ATTRIBUTE_UNUSED )`
- 4.9.2.25 `char* qemuBuildHostNetStr ( virDomainNetDefPtr net, struct qemud_driver * driver, qemuCapsPtr caps, char type_sep, int vlan, const char * tapfd, const char * vhostfd )`
- 4.9.2.26 `char* qemuBuildHubDevStr ( virDomainHubDefPtr dev, qemuCapsPtr caps )`
- 4.9.2.27 `static int qemuBuildIoEventFdStr ( virBufferPtr buf, enum virDomainIoEventFd use, qemuCapsPtr caps ) [static]`
- 4.9.2.28 `static int qemuBuildMachineArgStr ( virCommandPtr cmd, const virDomainDefPtr def, qemuCapsPtr caps ) [static]`
- 4.9.2.29 `char* qemuBuildMemballoonDevStr ( virDomainMemballoonDefPtr dev, qemuCapsPtr caps )`
- 4.9.2.30 `char* qemuBuildNicDevStr ( virDomainNetDefPtr net, int vlan, int bootindex, qemuCapsPtr caps )`
- 4.9.2.31 `char* qemuBuildNicStr ( virDomainNetDefPtr net, const char * prefix, int vlan )`
- 4.9.2.32 `static int qemuBuildNumaArgStr ( const virDomainDefPtr def, virCommandPtr cmd ) [static]`
- 4.9.2.33 `char* qemuBuildPCIHostdevDevStr ( virDomainHostdevDefPtr dev, const char * configfd, qemuCapsPtr caps )`
- 4.9.2.34 `char* qemuBuildPCIHostdevPCIDevStr ( virDomainHostdevDefPtr dev )`
- 4.9.2.35 `static int qemuBuildRBDString ( virConnectPtr conn, virDomainDiskDefPtr disk, virBufferPtr opt ) [static]`
- 4.9.2.36 `char* qemuBuildRedirdevDevStr ( virDomainDefPtr def, virDomainRedirdevDefPtr dev, qemuCapsPtr caps )`
- 4.9.2.37 `static int qemuBuildRomStr ( virBufferPtr buf, virDomainDeviceInfoPtr info, qemuCapsPtr caps ) [static]`
- 4.9.2.38 `static char* qemuBuildSmbiosBiosStr ( virSysinfoDefPtr def ) [static]`

- 4.9.2.39 static char\* qemuBuildSmbiosSystemStr ( virSysinfoDefPtr *def*, bool *skip\_uuid* ) [static]
- 4.9.2.40 static char\* qemuBuildSmpArgStr ( const virDomainDefPtr *def*, qemuCapsPtr *caps* ) [static]
- 4.9.2.41 static char\* qemuBuildSoundCodecStr ( virDomainSoundDefPtr *sound*, virDomainSoundCodecDefPtr *codec*, qemuCapsPtr *caps* ) [static]
- 4.9.2.42 char\* qemuBuildSoundDevStr ( virDomainSoundDefPtr *sound*, qemuCapsPtr *caps* )
- 4.9.2.43 static int qemuBuildUSBControllerDevStr ( virDomainDefPtr *domainDef*, virDomainControllerDefPtr *def*, qemuCapsPtr *caps*, virBuffer \* *buf* ) [static]
- 4.9.2.44 char\* qemuBuildUSBHostdevDevStr ( virDomainHostdevDefPtr *dev*, qemuCapsPtr *caps* )
- 4.9.2.45 char\* qemuBuildUSBHostdevUsbDevStr ( virDomainHostdevDefPtr *dev* )
- 4.9.2.46 char\* qemuBuildUSBInputDevStr ( virDomainInputDefPtr *dev*, qemuCapsPtr *caps* )
- 4.9.2.47 static char\* qemuBuildVideoDevStr ( virDomainVideoDefPtr *video*, qemuCapsPtr *caps* ) [static]
- 4.9.2.48 static char\* qemuBuildVirtioSerialPortDevStr ( virDomainChrDefPtr *dev*, qemuCapsPtr *caps* ) [static]
- 4.9.2.49 char\* qemuBuildWatchdogDevStr ( virDomainWatchdogDefPtr *dev*, qemuCapsPtr *caps* )
- 4.9.2.50 static int qemuCollectPCIAddress ( virDomainDefPtr *def* ATTRIBUTE\_UNUSED, virDomainDeviceDefPtr *device*, virDomainDeviceInfoPtr *info*, void \* *opaque* ) [static]
- 4.9.2.51 static int qemuControllerModelUSBToCaps ( int *model* ) [static]
- 4.9.2.52 char\* qemuDeviceDriveHostAlias ( virDomainDiskDefPtr *disk*, qemuCapsPtr *caps* )
- 4.9.2.53 int qemuDomainAssignAddresses ( virDomainDefPtr *def*, qemuCapsPtr *caps*, virDomainObjPtr *obj* )
- 4.9.2.54 int qemuDomainAssignPCIAddresses ( virDomainDefPtr *def*, qemuCapsPtr *caps*, virDomainObjPtr *obj* )
- 4.9.2.55 static int qemuDomainAssignS390Addresses ( virDomainDefPtr *def*, qemuCapsPtr *caps* ) [static]
- 4.9.2.56 int qemuDomainAssignSpaprVIOAddresses ( virDomainDefPtr *def*, qemuCapsPtr *caps* )
- 4.9.2.57 static int qemuDomainDeviceAliasIndex ( virDomainDeviceInfoPtr *info*, const char \* *prefix* ) [static]
- 4.9.2.58 int qemuDomainNetVLAN ( virDomainNetDefPtr *def* )
- 4.9.2.59 static int qemuDomainPCIAddressCheckSlot ( qemuDomainPCIAddressSetPtr *addrs*, virDomainDeviceInfoPtr *dev* ) [static]
- 4.9.2.60 int qemuDomainPCIAddressEnsureAddr ( qemuDomainPCIAddressSetPtr *addrs*, virDomainDeviceInfoPtr *dev* )
- 4.9.2.61 static int qemuDomainPCIAddressGetNextSlot ( qemuDomainPCIAddressSetPtr *addrs* ) [static]
- 4.9.2.62 int qemuDomainPCIAddressReleaseAddr ( qemuDomainPCIAddressSetPtr *addrs*, virDomainDeviceInfoPtr *dev* )
- 4.9.2.63 int qemuDomainPCIAddressReleaseFunction ( qemuDomainPCIAddressSetPtr *addrs*, int *slot*, int *function* )
- 4.9.2.64 int qemuDomainPCIAddressReleaseSlot ( qemuDomainPCIAddressSetPtr *addrs*, int *slot* )

- 4.9.2.65 `int qemuDomainPCIAddressReserveAddr ( qemuDomainPCIAddressSetPtr addrs, virDomainDeviceInfoPtr dev )`
- 4.9.2.66 `int qemuDomainPCIAddressReserveFunction ( qemuDomainPCIAddressSetPtr addrs, int slot, int function )`
- 4.9.2.67 `int qemuDomainPCIAddressReserveSlot ( qemuDomainPCIAddressSetPtr addrs, int slot )`
- 4.9.2.68 `qemuDomainPCIAddressSetPtr qemuDomainPCIAddressSetCreate ( virDomainDefPtr def )`
- 4.9.2.69 `void qemuDomainPCIAddressSetFree ( qemuDomainPCIAddressSetPtr addrs )`
- 4.9.2.70 `static void qemuDomainPCIAddressSetFreeEntry ( void * payload, const void *name ATTRIBUTE_UNUSED )`  
[static]
- 4.9.2.71 `int qemuDomainPCIAddressSetNextAddr ( qemuDomainPCIAddressSetPtr addrs, virDomainDeviceInfoPtr dev )`
- 4.9.2.72 `static void qemuDomainPrimeS390VirtioDevices ( virDomainDefPtr def, enum virDomainDeviceAddressType type )` [static]
- 4.9.2.73 `static const char* qemuFindEnv ( const char ** progenv, const char * name )` [static]
- 4.9.2.74 `static const char* qemuFindNICForVLAN ( int nnics, const char ** nics, int wantvlan )` [static]
- 4.9.2.75 `static virCPUDefPtr qemuInitGuestCPU ( virDomainDefPtr dom )` [static]
- 4.9.2.76 **POL Mod** `int qemuNetworklfaceConnect ( virDomainDefPtr def, virConnectPtr conn, struct qemud_driver * driver, virDomainNetDefPtr net, qemuCapsPtr caps )`

qemuNetworklfaceConnect

#### Parameters

<i>def</i>	pointer to virDomainDef structure
<i>conn</i>	pointer to virConnect structure
<i>driver</i>	pointer to qemud_driver structure
<i>net</i>	pointer to virDomainNetDef structure
<i>caps</i>	pointer to qemuCaps structure

Connect a domain network interface by using the QEMU driver

POL modification:

- Get the bridge type before calling virNetDevTapCreateInBridgePort

- 4.9.2.77 `int qemuOpenPCIConfig ( virDomainHostdevDefPtr dev )`
- 4.9.2.78 `int qemuOpenVhostNet ( virDomainDefPtr def, virDomainNetDefPtr net, qemuCapsPtr caps, int * vhostfd )`
- 4.9.2.79 `virDomainDefPtr qemuParseCommandLine ( virCapsPtr caps, const char ** progenv, const char ** progargv, char ** pidfile, virDomainChrSourceDefPtr * monConfig, bool * monJSON )`
- 4.9.2.80 `static void qemuParseCommandLineBootDevs ( virDomainDefPtr def, const char * str )` [static]
- 4.9.2.81 `static int qemuParseCommandLineChr ( virDomainChrSourceDefPtr source, const char * val )` [static]
- 4.9.2.82 `static int qemuParseCommandLineCPU ( virDomainDefPtr dom, const char * val )` [static]

- 4.9.2.83 `static virDomainDiskDefPtr qemuParseCommandLineDisk ( virCapsPtr caps, const char * val, int nvirtiodisk, bool old_style_ceph_args ) [static]`
- 4.9.2.84 `static virDomainNetDefPtr qemuParseCommandLineNet ( virCapsPtr caps, const char * val, int nnics, const char ** nics ) [static]`
- 4.9.2.85 `static virDomainHostdevDefPtr qemuParseCommandLinePCI ( const char * val ) [static]`
- 4.9.2.86 `virDomainDefPtr qemuParseCommandLinePid ( virCapsPtr caps, pid_t pid, char ** pidfile, virDomainChrSourceDefPtr * monConfig, bool * monJSON )`
- 4.9.2.87 `static int qemuParseCommandLineSmp ( virDomainDefPtr dom, const char * val ) [static]`
- 4.9.2.88 `virDomainDefPtr qemuParseCommandLineString ( virCapsPtr caps, const char * args, char ** pidfile, virDomainChrSourceDefPtr * monConfig, bool * monJSON )`
- 4.9.2.89 `static virDomainHostdevDefPtr qemuParseCommandLineUSB ( const char * val ) [static]`
- 4.9.2.90 `int qemuParseKeywords ( const char * str, char *** retkeywords, char *** retvalues, int allowEmptyValue )`
- 4.9.2.91 `static int qemuParseProcFileStrings ( int pid_value, const char * name, const char *** list ) [static]`
- 4.9.2.92 `static int qemuParseRBDString ( virDomainDiskDefPtr disk ) [static]`
- 4.9.2.93 `static char* qemuPCIAddressAsString ( virDomainDeviceInfoPtr dev ) [static]`
- 4.9.2.94 `int qemuPhysIfConnect ( virDomainDefPtr def, struct qemu_driver * driver, virDomainNetDefPtr net, qemuCapsPtr caps, enum virNetDevVPortProfileOp vmop )`
- 4.9.2.95 `static int qemuSafeSerialParamValue ( const char * value ) [static]`
- 4.9.2.96 `static int qemuSetScsiControllerModel ( virDomainDefPtr def, qemuCapsPtr caps, int * model ) [static]`
- 4.9.2.97 `static int qemuSoundCodecTypeToCaps ( int type ) [static]`
- 4.9.2.98 `static int qemuSpaprVIOFindByReg ( virDomainDefPtr def ATTRIBUTE_UNUSED, virDomainDeviceDefPtr device ATTRIBUTE_UNUSED, virDomainDeviceInfoPtr info, void * opaque ) [static]`
- 4.9.2.99 `static int qemuStringToArgvEnv ( const char * args, const char *** retenv, const char *** retargv ) [static]`
- 4.9.2.100 `static void qemuUsbld ( virBufferPtr buf, int idx ) [static]`
- 4.9.2.101 `VIR_ENUM_IMPL ( virDomainDiskQEMUBus, VIR_DOMAIN_DISK_BUS_LAST, "ide", "floppy", "scsi", "virtio", "xen", "usb", "uml", "sata" )`

## 4.10 src/uml/uml\_conf.c File Reference

```
#include <config.h>
#include <string.h>
#include <limits.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <arpa/inet.h>
#include <sys/utsname.h>
#include "uml_conf.h"
#include "uuid.h"
#include "buf.h"
#include "conf.h"
#include "util.h"
#include "memory.h"
#include "nodeinfo.h"
#include "logging.h"
#include "domain_nwfilter.h"
#include "virfile.h"
#include "virnetdevtap.h"
#include "virnodesuspend.h"
```

### Macros

- `#define VIR_FROM_THIS VIR_FROM_UML`
- `#define umlLog(level, msg,...) virLogMessage(__FILE__, level, 0, msg, __VA_ARGS__)`

### Functions

- static int `umlDefaultConsoleType` (const char \*ostype ATTRIBUTE\_UNUSED)
- virCapsPtr `umlCapsInit` (void)
- `POL Mod` static int `umlConnectTapDevice` (virConnectPtr conn, virDomainDefPtr vm, virDomainNetDefPtr net, const char \*bridge, const char \*bridge\_type)
- `POL Mod` static char \* `umlBuildCommandLineNet` (virConnectPtr conn, virDomainDefPtr vm, virDomainNetDefPtr def, int idx)
- static char \* `umlBuildCommandLineChr` (virDomainChrDefPtr def, const char \*dev, virCommandPtr cmd)
- static char \* `umlNextArg` (char \*args)
- virCommandPtr `umlBuildCommandLine` (virConnectPtr conn, struct uml\_driver \*driver, virDomainObjPtr vm)

### 4.10.1 Macro Definition Documentation

4.10.1.1 `#define umlLog( level, msg, ... ) virLogMessage(__FILE__, level, 0, msg, __VA_ARGS__)`

4.10.1.2 `#define VIR_FROM_THIS VIR_FROM_UML`

### 4.10.2 Function Documentation

4.10.2.1 `virCommandPtr umlBuildCommandLine ( virConnectPtr conn, struct uml_driver * driver, virDomainObjPtr vm )`

4.10.2.2 `static char* umlBuildCommandLineChr ( virDomainChrDefPtr def, const char * dev, virCommandPtr cmd )`  
`[static]`

4.10.2.3 **POL Mod** `static char* umlBuildCommandLineNet ( virConnectPtr conn, virDomainDefPtr vm,  
 virDomainNetDefPtr def, int idx )` `[static]`

umlBuildCommandLineNet

#### Parameters

<i>conn</i>	pointer to virConnect structure
<i>vm</i>	pointer to virDomainDef structure
<i>def</i>	pointer to virDomainNetDef structure
<i>idx</i>	index

Build the system command to configure domain network interfaces

Return a buffer with the system command to execute

POL modification:

- Get bridge type before calling umlConnectTapDevice

4.10.2.4 `virCapsPtr umlCapsInit ( void )`

4.10.2.5 **POL Mod** `static int umlConnectTapDevice ( virConnectPtr conn, virDomainDefPtr vm, virDomainNetDefPtr  
net, const char * bridge, const char * bridge_type )` `[static]`

umlConnectTapDevice

#### Parameters

<i>conn</i>	pointer to virConnect structure
<i>vm</i>	pointer to virDomainDef structure
<i>net</i>	pointer to virDomainNetDef structure
<i>bridge</i>	bridge name
<i>bridge_type</i>	bridge type

Connect a TAP device by using the UML driver

Return 0 if success, -1 otherwise

POL modification:

- Pass bridge\_type to virNetDevTapCreateInBridgePort

4.10.2.6 `static int umlDefaultConsoleType ( const char *ostype ATTRIBUTE_UNUSED )` `[static]`

4.10.2.7 `static char* umlNextArg ( char * args )` `[static]`

## 4.11 src/util/virnetdevbridge.c File Reference

```
#include <config.h>
#include "virnetdevbridge.h"
#include "virterror_internal.h"
#include "util.h"
#include "virfile.h"
#include "memory.h"
#include "intprops.h"
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/in.h>
```

### Macros

- `#define VIR_FROM_THIS VIR_FROM_NONE`

### Functions

- `int virNetDevBridgeCreate (const char *brname)`
- `int virNetDevBridgeDelete (const char *brname ATTRIBUTE_UNUSED)`
- `int virNetDevBridgeAddPort (const char *brname, const char *ifname)`
- `int virNetDevBridgeRemovePort (const char *brname, const char *ifname)`
- `int virNetDevBridgeSetSTPDelay (const char *brname, int delay ATTRIBUTE_UNUSED)`
- `int virNetDevBridgeGetSTPDelay (const char *brname, int *delay ATTRIBUTE_UNUSED)`
- `int virNetDevBridgeSetSTP (const char *brname, bool enable ATTRIBUTE_UNUSED)`
- `int virNetDevBridgeGetSTP (const char *brname, bool *enable ATTRIBUTE_UNUSED)`

### 4.11.1 Macro Definition Documentation

4.11.1.1 `#define VIR_FROM_THIS VIR_FROM_NONE`

### 4.11.2 Function Documentation

4.11.2.1 `int virNetDevBridgeAddPort ( const char * brname, const char * ifname )`

`virNetDevBridgeAddPort`: : the bridge name : the network interface name

Adds an interface to a bridge

Returns 0 in case of success or an errno code in case of failure.

4.11.2.2 `int virNetDevBridgeCreate ( const char * brname )`

`virNetDevBridgeCreate`: : the bridge name

This function register a new bridge

Returns 0 in case of success or -1 on failure



4.11.2.3 `int virNetDevBridgeDelete ( const char *brname ATTRIBUTE_UNUSED )`

virNetDevBridgeDelete: : the bridge name

Remove a bridge from the layer.

Returns 0 in case of success or an errno code in case of failure.

4.11.2.4 `int virNetDevBridgeGetSTP ( const char * brname, bool *enable ATTRIBUTE_UNUSED )`

4.11.2.5 `int virNetDevBridgeGetSTPDelay ( const char * brname, int *delay ATTRIBUTE_UNUSED )`

4.11.2.6 `int virNetDevBridgeRemovePort ( const char * brname, const char * ifname )`

virNetDevBridgeRemovePort: : the bridge name : the network interface name

Removes an interface from a bridge

Returns 0 in case of success or an errno code in case of failure.

4.11.2.7 `int virNetDevBridgeSetSTP ( const char * brname, bool enable ATTRIBUTE_UNUSED )`

4.11.2.8 `int virNetDevBridgeSetSTPDelay ( const char * brname, int delay ATTRIBUTE_UNUSED )`

## 4.12 src/util/virnetdevopenvswitch.c File Reference

```
#include <config.h>
#include <stdio.h>
#include "virnetdevopenvswitch.h"
#include "command.h"
#include "memory.h"
#include "virterror_internal.h"
#include "virmacaddr.h"
```

### Macros

- `#define VIR_FROM_THIS VIR_FROM_NONE`

### Functions

- **POL New** `int virNetDevOpenvswitchBridgeCreate (const char *brname)`
- **POL New** `int virNetDevOpenvswitchFakeBridgeCreate (const char *fakebrname, const char *brname, const char *tag)`
- **POL New** `int virNetDevOpenvswitchBridgeDelete (const char *brname)`
- **POL Mod** `int virNetDevOpenvswitchAddPort (const char *brname, const char *ifname, const virMacAddrPtr macaddr, const unsigned char *vmuuid, virNetDevVPortProfilePtr ovsport, virNetDevVlanPtr virtVlan)`
- `int virNetDevOpenvswitchRemovePort (const char *brname ATTRIBUTE_UNUSED, const char *ifname)`
- **POL New** `int virNetDevOpenvswitchBridgeSetSTPDelay (const char *brname ATTRIBUTE_UNUSED, int delay ATTRIBUTE_UNUSED)`
- **POL New** `int virNetDevOpenvswitchBridgeGetSTPDelay (const char *brname ATTRIBUTE_UNUSED, int *delay ATTRIBUTE_UNUSED)`
- **POL New** `int virNetDevOpenvswitchBridgeSetSTP (const char *brname ATTRIBUTE_UNUSED, bool enable ATTRIBUTE_UNUSED)`

- **POL New** int `virNetDevOpenvswitchBridgeGetSTP` (const char \*brname ATTRIBUTE\_UNUSED, bool \*enable ATTRIBUTE\_UNUSED)
- **POL New** int `virNetDevOpenvswitchTunnelCreate` (const char \*brname, const char \*device, const char \*remote\_ip, const char \*trunks)
- **POL New** int `virNetDevOpenvswitchTunnelDestroy` (const char \*brname, const char \*device)

#### 4.12.1 Macro Definition Documentation

##### 4.12.1.1 #define VIR\_FROM\_THIS VIR\_FROM\_NONE

#### 4.12.2 Function Documentation

##### 4.12.2.1 **POL Mod** int `virNetDevOpenvswitchAddPort` ( const char \* *brname*, const char \* *ifname*, const virMacAddrPtr *macaddr*, const unsigned char \* *vmuuid*, virNetDevVPortProfilePtr *ovsport*, virNetDevVlanPtr *virtVlan* )

`virNetDevOpenvswitchAddPort`: : the bridge name : the network interface name : the mac address of the virtual interface : the Domain UUID that has this interface : the ovs specific fields

Add an interface to the OVS bridge

Returns 0 in case of success or -1 in case of failure.

POL modification:

- Added support for `ovsport=null` and `vmuuid=null`

##### 4.12.2.2 **POL New** int `virNetDevOpenvswitchBridgeCreate` ( const char \* *brname* )

`virNetDevOpenvswitchBridgeCreate`: : the bridge name

This function register a new Open vSwitch bridge

Returns 0 in case of success or -1 on failure

##### 4.12.2.3 **POL New** int `virNetDevOpenvswitchBridgeDelete` ( const char \* *brname* )

`virNetDevBridgeDelete`: : the bridge name

Remove an Open vSwitch bridge.

Returns 0 in case of success or an errno code in case of failure.

##### 4.12.2.4 **POL New** int `virNetDevOpenvswitchBridgeGetSTP` ( const char \*brname ATTRIBUTE\_UNUSED, bool \*enable ATTRIBUTE\_UNUSED )

`virNetDevBridgeGetSTP`: : the bridge device name : returns the STP state

Determine the state of the spanning tree protocol on the device , returning the state in

Returns 0 on success, -1 on error

##### 4.12.2.5 **POL New** int `virNetDevOpenvswitchBridgeGetSTPDelay` ( const char \*brname ATTRIBUTE\_UNUSED, int \*delay ATTRIBUTE\_UNUSED )

`virNetDevOpenvswitchBridgeGetSTPDelay`: : the bridge device name : the forward delay in milliseconds

Retrives the forward delay for the bridge device storing it in . The forward delay is only meaningful if STP is enabled

Returns 0 on success, -1 on error

4.12.2.6 **POL New** int virNetDevOpenvswitchBridgeSetSTP ( const char \*brname *ATTRIBUTE\_UNUSED*, bool enable *ATTRIBUTE\_UNUSED* )

virNetDevBridgeSetSTP: : the bridge name : 1 to enable, 0 to disable

Control whether the bridge participates in the spanning tree protocol, in general don't disable it without good reasons.

Returns 0 in case of success or -1 on failure

4.12.2.7 **POL New** int virNetDevOpenvswitchBridgeSetSTPDelay ( const char \*brname *ATTRIBUTE\_UNUSED*, int delay *ATTRIBUTE\_UNUSED* )

virNetDevOpenvswitchBridgeSetSTPDelay: : the bridge name : delay in seconds

Set the bridge forward delay

Returns 0 in case of success or -1 on failure

4.12.2.8 **POL New** int virNetDevOpenvswitchFakeBridgeCreate ( const char \*fakebrname, const char \*brname, const char \*tag )

virNetDevOpenvswitchFakeBridgeCreate: : the fake bridge name : the bridge name : the fakebridge tag This function register a new Open vSwitch bridge

Returns 0 in case of success or -1 on failure

4.12.2.9 int virNetDevOpenvswitchRemovePort ( const char \*brname *ATTRIBUTE\_UNUSED*, const char \*ifname )

virNetDevOpenvswitchRemovePort: : the network interface name

Deletes an interface from a OVS bridge

Returns 0 in case of success or -1 in case of failure.

4.12.2.10 **POL New** int virNetDevOpenvswitchTunnelCreate ( const char \*brname, const char \*device, const char \*remote\_ip, const char \*trunks )

virNetDevOpenvswitchTunnelCreate: : bridge name : tunnel device name : tunnel remote ip : white list

Create an ipsec tunnel in bridge name, connected with remote\_ip

Returns 0 on success, -1 on error

4.12.2.11 **POL New** int virNetDevOpenvswitchTunnelDestroy ( const char \*brname, const char \*device )

virNetDevOpenvswitchTunnelDestroy: : bridge name : tunnel device name

Destroy an ipsec tunnel in bridge name

Returns 0 on success, -1 on error

## 4.13 src/util/virnetdevopenvswitch.h File Reference

```
#include "internal.h"
#include "util.h"
#include "virnetdevvportprofile.h"
#include "virnetdevvlan.h"
```

## Functions

- **POL New** int [virNetDevOpenvswitchBridgeCreate](#) (const char \*brname) [ATTRIBUTE\\_NONNULL\(1\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- **POL New** int [virNetDevOpenvswitchBridgeDelete](#) (const char \*brname) [ATTRIBUTE\\_NONNULL\(1\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- **POL New** int [virNetDevOpenvswitchFakeBridgeCreate](#) (const char \*fakebrname, const char \*brname, const char \*tag)
- [ATTRIBUTE\\_NONNULL](#) (1) [ATTRIBUTE\\_NONNULL\(2\)](#) [ATTRIBUTE\\_NONNULL\(3\)](#) int [virNetDevOpenvswitchAddPort](#)(const char \*brname
- int [virNetDevOpenvswitchRemovePort](#) (const char \*brname, const char \*ifname) [ATTRIBUTE\\_NONNULL\(2\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- **POL New** int [virNetDevOpenvswitchBridgeSetSTPDelay](#) (const char \*brname, int delay) [ATTRIBUTE\\_NONNULL\(1\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- **POL New** int [virNetDevOpenvswitchBridgeGetSTPDelay](#) (const char \*brname, int \*delay) [ATTRIBUTE\\_NONNULL\(1\)](#) [ATTRIBUTE\\_NONNULL\(2\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- **POL New** int [virNetDevOpenvswitchBridgeSetSTP](#) (const char \*brname, bool enable) [ATTRIBUTE\\_NONNULL\(1\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- **POL New** int [virNetDevOpenvswitchBridgeGetSTP](#) (const char \*brname, bool \*enable) [ATTRIBUTE\\_NONNULL\(1\)](#) [ATTRIBUTE\\_NONNULL\(2\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- **POL New** int [virNetDevOpenvswitchTunnelCreate](#) (const char \*brname, const char \*device, const char \*remote\_ip, const char \*trunks) [ATTRIBUTE\\_NONNULL\(1\)](#) [ATTRIBUTE\\_NONNULL\(2\)](#) [ATTRIBUTE\\_NONNULL\(3\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)
- int [virNetDevOpenvswitchTunnelDestroy](#) (const char \*brname, const char \*device) [ATTRIBUTE\\_NONNULL\(1\)](#) [ATTRIBUTE\\_NONNULL\(2\)](#) [ATTRIBUTE\\_RETURN\\_CHECK](#)

## Variables

- const char \* ifname
- const char const virMacAddrPtr [macaddr](#)
- const char const virMacAddrPtr  
const unsigned char \* [vmuuid](#)
- const char const virMacAddrPtr  
const unsigned char  
virNetDevVPortProfilePtr [ovsport](#)

### 4.13.1 Function Documentation

4.13.1.1 const char const virMacAddrPtr const unsigned char virNetDevVPortProfilePtr virNetDevVlanPtr virtVlan  
[ATTRIBUTE\\_NONNULL](#) ( 1 ) const

4.13.1.2 **POL New** int [virNetDevOpenvswitchBridgeCreate](#) ( const char \* *brname* )

[virNetDevOpenvswitchBridgeCreate](#): : the bridge name

This function register a new Open vSwitch bridge

Returns 0 in case of success or -1 on failure

4.13.1.3 **POL New** int [virNetDevOpenvswitchBridgeDelete](#) ( const char \* *brname* )

[virNetDevBridgeDelete](#): : the bridge name

Remove an Open vSwitch bridge.

Returns 0 in case of success or an errno code in case of failure.

4.13.1.4 **POL New** int virNetDevOpenvswitchBridgeGetSTP ( const char \* *brname*, bool \* *enable* )

4.13.1.5 **POL New** int virNetDevOpenvswitchBridgeGetSTPDelay ( const char \* *brname*, int \* *delay* )

4.13.1.6 **POL New** int virNetDevOpenvswitchBridgeSetSTP ( const char \* *brname*, bool *enable* )

4.13.1.7 **POL New** int virNetDevOpenvswitchBridgeSetSTPDelay ( const char \* *brname*, int *delay* )

4.13.1.8 **POL New** int virNetDevOpenvswitchFakeBridgeCreate ( const char \* *fakebrname*, const char \* *brname*, const char \* *tag* )

virNetDevOpenvswitchFakeBridgeCreate: : the fake bridge name : the bridge name : the fakebridge tag This function register a new Open vSwitch bridge

Returns 0 in case of success or -1 on failure

4.13.1.9 int virNetDevOpenvswitchRemovePort ( const char \* *brname*, const char \* *ifname* )

4.13.1.10 **POL New** int virNetDevOpenvswitchTunnelCreate ( const char \* *brname*, const char \* *device*, const char \* *remote\_ip*, const char \* *trunks* )

virNetDevOpenvswitchTunnelCreate: : bridge name : tunnel device name : tunnel remote ip : white list

Create an ipsec tunnel in bridge name, connected with remote\_ip

Returns 0 on success, -1 on error

4.13.1.11 **POL New** int virNetDevOpenvswitchTunnelDestroy ( const char \* *brname*, const char \* *device* )

virNetDevOpenvswitchTunnelDestroy: : bridge name : tunnel device name

Destroy an ipsec tunnel in bridge name

Returns 0 on success, -1 on error

## 4.13.2 Variable Documentation

4.13.2.1 const char\* ifname

4.13.2.2 const char const virMacAddrPtr macaddr

4.13.2.3 const char const virMacAddrPtr const unsigned char virNetDevVPortProfilePtr ovspport

4.13.2.4 const char const virMacAddrPtr const unsigned char\* vmuuid

## 4.14 src/util/virnetdevtap.c File Reference

```
#include <config.h>
```

```
#include "virmacaddr.h"
#include "virnetdevtap.h"
#include "virnetdev.h"
#include "virnetdevbridge.h"
#include "virnetdevopenvswitch.h"
#include "virterror_internal.h"
#include "virfile.h"
#include "memory.h"
#include "logging.h"
#include "util.h"
#include <sys/ioctl.h>
#include <net/if.h>
#include <fcntl.h>
```

## Macros

- `#define VIR_FROM_THIS VIR_FROM_NONE`

## Functions

- int `virNetDevTapCreate` (char **\*\*ifname** ATTRIBUTE\_UNUSED, int **\*tapfd** ATTRIBUTE\_UNUSED, unsigned int **flags** ATTRIBUTE\_UNUSED)
- int `virNetDevTapDelete` (const char **\*ifname** ATTRIBUTE\_UNUSED)
- **POL Mod** int `virNetDevTapCreateInBridgePort` (const char **\*brname**, const char **\*brtype**, char **\*\*ifname**, const virMacAddrPtr **macaddr**, const unsigned char **\*vmuuid**, int **\*tapfd**, virNetDevVPortProfilePtr **virtPortProfile**, virNetDevVlanPtr **virtVlan**, unsigned int **flags**)

### 4.14.1 Macro Definition Documentation

#### 4.14.1.1 `#define VIR_FROM_THIS VIR_FROM_NONE`

### 4.14.2 Function Documentation

#### 4.14.2.1 int `virNetDevTapCreate` ( char **\*\*ifname** *ATTRIBUTE\_UNUSED*, int **\*tapfd** *ATTRIBUTE\_UNUSED*, unsigned int **flags** *ATTRIBUTE\_UNUSED* )

`virNetDevProbeVnetHdr`: : a tun/tap file descriptor

Check whether it is safe to enable the IFF\_VNET\_HDR flag on the tap interface.

Setting IFF\_VNET\_HDR enables QEMU's virtio\_net driver to allow guests to pass larger (GSO) packets, with partial checksums, to the host. This greatly increases the achievable throughput.

It is only useful to enable this when we're setting up a virtio interface. And it is only *safe* to enable it when we know for sure that a) qemu has support for IFF\_VNET\_HDR and b) the running kernel implements the TUNGETIFF ioctl(), which qemu needs to query the supplied tapfd.

Returns 1 if VnetHdr is supported, 0 if not supported

#### 4.14.2.2 **POL Mod** int `virNetDevTapCreateInBridgePort` ( const char **\*brname**, const char **\*brtype**, char **\*\*ifname**, const virMacAddrPtr **macaddr**, const unsigned char **\*vmuuid**, int **\*tapfd**, virNetDevVPortProfilePtr **virtPortProfile**, virNetDevVlanPtr **virtVlan**, unsigned int **flags** )

`virNetDevTapCreateInBridgePort`: : the bridge name : the interface name (or name template) : desired MAC address : file descriptor return value for the new tap device : bridge/port specific configuration : OR of `virNetDevTapCreateFlags`:

## VIR\_NETDEV\_TAP\_CREATE\_IFUP

- Bring the interface up VIR\_NETDEV\_TAP\_CREATE\_VNET\_HDR
- Enable IFF\_VNET\_HDR on the tap device VIR\_NETDEV\_TAP\_CREATE\_USE\_MAC\_FOR\_BRIDGE
- Set this interface's MAC as the bridge's MAC address VIR\_NETDEV\_TAP\_CREATE\_PERSIST
- The device will persist after the file descriptor is closed

This function creates a new tap device on a bridge. can be either a fixed name or a name template with 'd' for dynamic name allocation. in either case the final name for the bridge will be stored in . If the parameter is supplied, the open tap device file descriptor will be returned, otherwise the TAP device will be closed. The caller must use virNetDevTapDelete to remove a persistent TAP device when it is no longer needed.

Returns 0 in case of success or -1 on failure

POL modification:

- Use bridge type to determine if virNetDevOpenvswitchAddPort must be executed

4.14.2.3 int virNetDevTapDelete ( const char \*ifname *ATTRIBUTE\_UNUSED* )

## 4.15 src/util/virnetdevtap.h File Reference

```
#include "internal.h"
#include "virnetdevvportprofile.h"
#include "virnetdevvlan.h"
```

## Enumerations

- enum virNetDevTapCreateFlags {  
VIR\_NETDEV\_TAP\_CREATE\_NONE = 0, VIR\_NETDEV\_TAP\_CREATE\_IFUP = 1 << 0, VIR\_NETDEV\_TAP\_CREATE\_VNET\_HDR = 1 << 1, VIR\_NETDEV\_TAP\_CREATE\_USE\_MAC\_FOR\_BRIDGE = 1 << 2,  
VIR\_NETDEV\_TAP\_CREATE\_PERSIST = 1 << 3 }

## Functions

- int virNetDevTapCreate (char \*\*ifname, int \*tapfd, unsigned int flags) *ATTRIBUTE\_NONNULL*(1) *ATTRIBUTE\_RETURN\_CHECK*
- int virNetDevTapDelete (const char \*ifname) *ATTRIBUTE\_NONNULL*(1) *ATTRIBUTE\_RETURN\_CHECK*
- *POL Mod* int virNetDevTapCreateInBridgePort (const char \*brname, const char \*brtype, char \*\*ifname, const virMacAddrPtr macaddr, const unsigned char \*vmuuid, int \*tapfd, virNetDevVPortProfilePtr virtPortProfile, virNetDevVlanPtr virtVlan, unsigned int flags) *ATTRIBUTE\_NONNULL*(1) *ATTRIBUTE\_NONNULL*(3) *ATTRIBUTE\_NONNULL*(4) *ATTRIBUTE\_RETURN\_CHECK*

## 4.15.1 Enumeration Type Documentation

## 4.15.1.1 enum virNetDevTapCreateFlags

## Enumerator

**VIR\_NETDEV\_TAP\_CREATE\_NONE**  
**VIR\_NETDEV\_TAP\_CREATE\_IFUP**

***VIR\_NETDEV\_TAP\_CREATE\_VNET\_HDR***  
***VIR\_NETDEV\_TAP\_CREATE\_USE\_MAC\_FOR\_BRIDGE***  
***VIR\_NETDEV\_TAP\_CREATE\_PERSIST***

## 4.15.2 Function Documentation

4.15.2.1 `int virNetDevTapCreate ( char ** ifname, int * tapfd, unsigned int flags )`

4.15.2.2 **POL Mod** `int virNetDevTapCreateInBridgePort ( const char * brname, const char * brtype, char ** ifname, const virMacAddrPtr macaddr, const unsigned char * vmuuid, int * tapfd, virNetDevVPortProfilePtr virtPortProfile, virNetDevVlanPtr virtVlan, unsigned int flags )`

`virNetDevTapCreateInBridgePort`: : the bridge name : the interface name (or name template) : desired MAC address : file descriptor return value for the new tap device : bridge/port specific configuration : OR of `virNetDevTapCreate`-Flags:

***VIR\_NETDEV\_TAP\_CREATE\_IFUP***

- Bring the interface up ***VIR\_NETDEV\_TAP\_CREATE\_VNET\_HDR***
- Enable IFF\_VNET\_HDR on the tap device ***VIR\_NETDEV\_TAP\_CREATE\_USE\_MAC\_FOR\_BRIDGE***
- Set this interface's MAC as the bridge's MAC address ***VIR\_NETDEV\_TAP\_CREATE\_PERSIST***
- The device will persist after the file descriptor is closed

This function creates a new tap device on a bridge. can be either a fixed name or a name template with 'd' for dynamic name allocation. in either case the final name for the bridge will be stored in . If the parameter is supplied, the open tap device file descriptor will be returned, otherwise the TAP device will be closed. The caller must use `virNetDevTapDelete` to remove a persistent TAP device when it is no longer needed.

Returns 0 in case of success or -1 on failure

POL modification:

- Use bridge type to determine if `virNetDevOpenvswitchAddPort` must be executed

4.15.2.3 `int virNetDevTapDelete ( const char * ifname )`

## 4.16 tools/virsh-network.c File Reference

```
#include <config.h>
#include "virsh-network.h"
#include <libxml/parser.h>
#include <libxml/tree.h>
#include <libxml/xpath.h>
#include <libxml/xmlsave.h>
#include "buf.h"
#include "memory.h"
#include "util.h"
#include "xml.h"
#include "conf/network_conf.h"
#include "virsh-edit.c"
```

## Data Structures

- struct [vshNetworkList](#)



## Macros

- `#define EDIT_GET_XML vshNetworkGetXMLDesc(network)`
- `#define EDIT_NOT_CHANGED`
- `#define EDIT_DEFINE (network_edited = virNetworkDefineXML(ctl->conn, doc_edited))`
- `#define EDIT_FREE`

## Typedefs

- `typedef struct vshNetworkList * vshNetworkListPtr`

## Functions

- `virNetworkPtr vshCommandOptNetworkBy` (vshControl \*ctl, const vshCmd \*cmd, const char \*\*name, unsigned int flags)
- `static bool cmdNetworkAutostart` (vshControl \*ctl, const vshCmd \*cmd)
- `static bool cmdNetworkCreate` (vshControl \*ctl, const vshCmd \*cmd)
- `static bool cmdNetworkDefine` (vshControl \*ctl, const vshCmd \*cmd)
- `static bool cmdNetworkDestroy` (vshControl \*ctl, const vshCmd \*cmd)
- `static bool cmdNetworkDumpXML` (vshControl \*ctl, const vshCmd \*cmd)
- `static bool cmdNetworkInfo` (vshControl \*ctl, const vshCmd \*cmd)
- `static int vshNetworkSorter` (const void \*a, const void \*b)
- `static void vshNetworkListFree` (vshNetworkListPtr list)
- `static vshNetworkListPtr vshNetworkListCollect` (vshControl \*ctl, unsigned int flags)
- `static bool cmdNetworkList` (vshControl \*ctl, const vshCmd \*cmd ATTRIBUTE\_UNUSED)
- `static bool cmdNetworkName` (vshControl \*ctl, const vshCmd \*cmd)
- `static bool cmdNetworkStart` (vshControl \*ctl, const vshCmd \*cmd)
- `static bool cmdNetworkUndefine` (vshControl \*ctl, const vshCmd \*cmd)
- `VIR_ENUM_IMPL` (virNetworkUpdateCommand, VIR\_NETWORK\_UPDATE\_COMMAND\_LAST, "none", "modify", "delete", "add-last", "add-first")
- `VIR_ENUM_IMPL` (virNetworkSection, VIR\_NETWORK\_SECTION\_LAST, "none", "bridge", "domain", "ip", "ip-dhcp-host", "ip-dhcp-range", "forward", "forward-interface", "forward-pf", "portgroup", "dns-host", "dns-txt", "dns-srv", "tunnel")
- `static bool cmdNetworkUpdate` (vshControl \*ctl, const vshCmd \*cmd)
- `static bool cmdNetworkUuid` (vshControl \*ctl, const vshCmd \*cmd)
- `static char * vshNetworkGetXMLDesc` (virNetworkPtr network)
- `static bool cmdNetworkEdit` (vshControl \*ctl, const vshCmd \*cmd)

## Variables

- `static const vshCmdInfo info_network_autostart []`
- `static const vshCmdOptDef opts_network_autostart []`
- `static const vshCmdInfo info_network_create []`
- `static const vshCmdOptDef opts_network_create []`
- `static const vshCmdInfo info_network_define []`
- `static const vshCmdOptDef opts_network_define []`
- `static const vshCmdInfo info_network_destroy []`
- `static const vshCmdOptDef opts_network_destroy []`
- `static const vshCmdInfo info_network_dumpxml []`
- `static const vshCmdOptDef opts_network_dumpxml []`
- `static const vshCmdInfo info_network_info []`
- `static const vshCmdOptDef opts_network_info []`
- `static const vshCmdInfo info_network_list []`

- static const vshCmdOptDef [opts\\_network\\_list](#) []
- static const vshCmdInfo [info\\_network\\_name](#) []
- static const vshCmdOptDef [opts\\_network\\_name](#) []
- static const vshCmdInfo [info\\_network\\_start](#) []
- static const vshCmdOptDef [opts\\_network\\_start](#) []
- static const vshCmdInfo [info\\_network\\_undefine](#) []
- static const vshCmdOptDef [opts\\_network\\_undefine](#) []
- static const vshCmdInfo [info\\_network\\_update](#) []
- static const vshCmdOptDef [opts\\_network\\_update](#) []
- static const vshCmdInfo [info\\_network\\_uuid](#) []
- static const vshCmdOptDef [opts\\_network\\_uuid](#) []
- static const vshCmdInfo [info\\_network\\_edit](#) []
- static const vshCmdOptDef [opts\\_network\\_edit](#) []
- const vshCmdDef [networkCmds](#) []

#### 4.16.1 Macro Definition Documentation

4.16.1.1 **#define EDIT\_DEFINE** (network\_edited = virNetworkDefineXML(ctl->conn, doc\_edited))

4.16.1.2 **#define EDIT\_FREE**

**Value:**

```
if (network_edited) \
    virNetworkFree(network_edited);
```

4.16.1.3 **#define EDIT\_GET\_XML** vshNetworkGetXMLDesc(network)

4.16.1.4 **#define EDIT\_NOT\_CHANGED**

**Value:**

```
vshPrint(ctl, _("Network %s XML configuration not changed.\n"), \
    virNetworkGetName(network)); \
ret = true; goto edit_cleanup;
```

#### 4.16.2 Typedef Documentation

4.16.2.1 **typedef struct vshNetworkList\*** vshNetworkListPtr

#### 4.16.3 Function Documentation

4.16.3.1 **static bool** cmdNetworkAutostart ( vshControl \* *ctl*, const vshCmd \* *cmd* ) [static]

4.16.3.2 **static bool** cmdNetworkCreate ( vshControl \* *ctl*, const vshCmd \* *cmd* ) [static]

4.16.3.3 **static bool** cmdNetworkDefine ( vshControl \* *ctl*, const vshCmd \* *cmd* ) [static]

4.16.3.4 **static bool** cmdNetworkDestroy ( vshControl \* *ctl*, const vshCmd \* *cmd* ) [static]

4.16.3.5 **static bool** cmdNetworkDumpXML ( vshControl \* *ctl*, const vshCmd \* *cmd* ) [static]

4.16.3.6 **static bool** cmdNetworkEdit ( vshControl \* *ctl*, const vshCmd \* *cmd* ) [static]

- 4.16.3.7 `static bool cmdNetworkInfo ( vshControl * ctl, const vshCmd * cmd ) [static]`
- 4.16.3.8 `static bool cmdNetworkList ( vshControl * ctl, const vshCmd * cmd ATTRIBUTE_UNUSED ) [static]`
- 4.16.3.9 `static bool cmdNetworkName ( vshControl * ctl, const vshCmd * cmd ) [static]`
- 4.16.3.10 `static bool cmdNetworkStart ( vshControl * ctl, const vshCmd * cmd ) [static]`
- 4.16.3.11 `static bool cmdNetworkUndefine ( vshControl * ctl, const vshCmd * cmd ) [static]`
- 4.16.3.12 `static bool cmdNetworkUpdate ( vshControl * ctl, const vshCmd * cmd ) [static]`
- 4.16.3.13 `static bool cmdNetworkUuid ( vshControl * ctl, const vshCmd * cmd ) [static]`
- 4.16.3.14 `VIR_ENUM_IMPL ( virNetworkUpdateCommand , VIR_NETWORK_UPDATE_COMMAND_LAST , "none", "modify", "delete", "add-last", "add-first" )`
- 4.16.3.15 `VIR_ENUM_IMPL ( virNetworkSection , VIR_NETWORK_SECTION_LAST , "none", "bridge", "domain", "ip", "ip-dhcp-host", "ip-dhcp-range", "forward", "forward-interface", "forward-pf", "portgroup", "dns-host", "dns-txt", "dns-srv", "tunnel" )`
- 4.16.3.16 `virNetworkPtr vshCommandOptNetworkBy ( vshControl * ctl, const vshCmd * cmd, const char ** name, unsigned int flags )`
- 4.16.3.17 `static char* vshNetworkGetXMLDesc ( virNetworkPtr network ) [static]`
- 4.16.3.18 `static vshNetworkListPtr vshNetworkListCollect ( vshControl * ctl, unsigned int flags ) [static]`
- 4.16.3.19 `static void vshNetworkListFree ( vshNetworkListPtr list ) [static]`
- 4.16.3.20 `static int vshNetworkSorter ( const void * a, const void * b ) [static]`

## 4.16.4 Variable Documentation

- 4.16.4.1 `const vshCmdInfo info_network_autostart[] [static]`

### Initial value:

```
= {
    {"help", N_("autostart a network")},
    {"desc",
     N_("Configure a network to be automatically started at boot.")},
    {NULL, NULL}
}
```

- 4.16.4.2 `const vshCmdInfo info_network_create[] [static]`

### Initial value:

```
= {
    {"help", N_("create a network from an XML file")},
    {"desc", N_("Create a network.")},
    {NULL, NULL}
}
```

- 4.16.4.3 `const vshCmdInfo info_network_define[] [static]`

### Initial value:

```
= {
    {"help", N_("define (but don't start) a network from an XML file")},
    {"desc", N_("Define a network.")},
    {NULL, NULL}
}
```

#### 4.16.4.4 const vshCmdInfo info\_network\_destroy[] [static]

**Initial value:**

```
= {
    {"help", N_("destroy (stop) a network")},
    {"desc", N_("Forcefully stop a given network.")},
    {NULL, NULL}
}
```

#### 4.16.4.5 const vshCmdInfo info\_network\_dumpxml[] [static]

**Initial value:**

```
= {
    {"help", N_("network information in XML")},
    {"desc", N_("Output the network information as an XML dump to stdout.")},
    {NULL, NULL}
}
```

#### 4.16.4.6 const vshCmdInfo info\_network\_edit[] [static]

**Initial value:**

```
= {
    {"help", N_("edit XML configuration for a network")},
    {"desc", N_("Edit the XML configuration for a network.")},
    {NULL, NULL}
}
```

#### 4.16.4.7 const vshCmdInfo info\_network\_info[] [static]

**Initial value:**

```
= {
    {"help", N_("network information")},
    {"desc", N_("Returns basic information about the network")},
    {NULL, NULL}
}
```

#### 4.16.4.8 const vshCmdInfo info\_network\_list[] [static]

**Initial value:**

```
= {
    {"help", N_("list networks")},
    {"desc", N_("Returns list of networks.")},
    {NULL, NULL}
}
```

4.16.4.9 `const vshCmdInfo info_network_name[]` `[static]`**Initial value:**

```
= {
    {"help", N_("convert a network UUID to network name")},
    {"desc", ""},
    {NULL, NULL}
}
```

4.16.4.10 `const vshCmdInfo info_network_start[]` `[static]`**Initial value:**

```
= {
    {"help", N_("start a (previously defined) inactive network")},
    {"desc", N_("Start a network.")},
    {NULL, NULL}
}
```

4.16.4.11 `const vshCmdInfo info_network_undefine[]` `[static]`**Initial value:**

```
= {
    {"help", N_("undefine an inactive network")},
    {"desc", N_("Undefine the configuration for an inactive network.")},
    {NULL, NULL}
}
```

4.16.4.12 `const vshCmdInfo info_network_update[]` `[static]`**Initial value:**

```
= {
    {"help", N_("update parts of an existing network's configuration")},
    {"desc", ""},
    {NULL, NULL}
}
```

4.16.4.13 `const vshCmdInfo info_network_uuid[]` `[static]`**Initial value:**

```
= {
    {"help", N_("convert a network name to network UUID")},
    {"desc", ""},
    {NULL, NULL}
}
```

4.16.4.14 `const vshCmdDef networkCmds[]`**Initial value:**

```
= {
    {"net-autostart", cmdNetworkAutostart,
     opts_network_autostart,
     info_network_autostart, 0},
    {"net-create", cmdNetworkCreate, opts_network_create,
```

```

    info_network_create, 0},
{"net-define", cmdNetworkDefine, opts_network_define,
 info_network_define, 0},
{"net-destroy", cmdNetworkDestroy, opts_network_destroy,
 info_network_destroy, 0},
{"net-dumpxml", cmdNetworkDumpXML, opts_network_dumpxml,
 info_network_dumpxml, 0},
{"net-edit", cmdNetworkEdit, opts_network_edit,
 info_network_edit, 0},
{"net-info", cmdNetworkInfo, opts_network_info,
 info_network_info, 0},
{"net-list", cmdNetworkList, opts_network_list,
 info_network_list, 0},
{"net-name", cmdNetworkName, opts_network_name,
 info_network_name, 0},
{"net-start", cmdNetworkStart, opts_network_start,
 info_network_start, 0},
{"net-undefine", cmdNetworkUndefine, opts_network_undefine,
 info_network_undefine, 0},
{"net-update", cmdNetworkUpdate, opts_network_update,
 info_network_update, 0},
{"net-uuid", cmdNetworkUuid, opts_network_uuid,
 info_network_uuid, 0},
{NULL, NULL, NULL, NULL, 0}
}

```

#### 4.16.4.15 const vshCmdOptDef opts\_network\_autostart[] [static]

##### Initial value:

```

= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name or uuid")},
    {"disable", VSH_OT_BOOL, 0, N_("disable autostarting")},
    {NULL, 0, 0, NULL}
}

```

#### 4.16.4.16 const vshCmdOptDef opts\_network\_create[] [static]

##### Initial value:

```

= {
    {"file", VSH_OT_DATA, VSH_OFLAG_REQ, N_("file containing an XML network description")},
    {NULL, 0, 0, NULL}
}

```

#### 4.16.4.17 const vshCmdOptDef opts\_network\_define[] [static]

##### Initial value:

```

= {
    {"file", VSH_OT_DATA, VSH_OFLAG_REQ, N_("file containing an XML network description")},
    {NULL, 0, 0, NULL}
}

```

#### 4.16.4.18 const vshCmdOptDef opts\_network\_destroy[] [static]

##### Initial value:

```

= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name or uuid")},
    {NULL, 0, 0, NULL}
}

```

**4.16.4.19** `const vshCmdOptDef opts_network_dumpxml[]` `[static]`**Initial value:**

```
= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name or uuid")},
    {"inactive", VSH_OT_BOOL, VSH_OFLAG_NONE, N_("network information of an inactive domain")},
    {NULL, 0, 0, NULL}
}
```

**4.16.4.20** `const vshCmdOptDef opts_network_edit[]` `[static]`**Initial value:**

```
= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name or uuid")},
    {NULL, 0, 0, NULL}
}
```

**4.16.4.21** `const vshCmdOptDef opts_network_info[]` `[static]`**Initial value:**

```
= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name or uuid")},
    {NULL, 0, 0, NULL}
}
```

**4.16.4.22** `const vshCmdOptDef opts_network_list[]` `[static]`**Initial value:**

```
= {
    {"inactive", VSH_OT_BOOL, 0, N_("list inactive networks")},
    {"all", VSH_OT_BOOL, 0, N_("list inactive & active networks")},
    {"persistent", VSH_OT_BOOL, 0, N_("list persistent networks")},
    {"transient", VSH_OT_BOOL, 0, N_("list transient networks")},
    {"autostart", VSH_OT_BOOL, 0, N_("list networks with autostart enabled")},
    {"no-autostart", VSH_OT_BOOL, 0, N_("list networks with autostart disabled")},
    {NULL, 0, 0, NULL}
}
```

**4.16.4.23** `const vshCmdOptDef opts_network_name[]` `[static]`**Initial value:**

```
= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network uuid")},
    {NULL, 0, 0, NULL}
}
```

**4.16.4.24** `const vshCmdOptDef opts_network_start[]` `[static]`**Initial value:**

```
= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name or uuid")},
    {NULL, 0, 0, NULL}
}
```

#### 4.16.4.25 `const vshCmdOptDef opts_network_undefine[]` [static]

##### Initial value:

```
= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name or uuid")},
    {NULL, 0, 0, NULL}
}
```

#### 4.16.4.26 `const vshCmdOptDef opts_network_update[]` [static]

##### Initial value:

```
= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name or uuid")},
    {"command", VSH_OT_DATA, VSH_OFLAG_REQ,
     N_("type of update (add-first, add-last (add), delete, or modify)")},
    {"section", VSH_OT_DATA, VSH_OFLAG_REQ,
     N_("which section of network configuration to update")},
    {"xml", VSH_OT_DATA, VSH_OFLAG_REQ,
     N_("name of file containing xml (or, if it starts with '<', the complete "
        "xml element itself) to add/modify, or to be matched for search")},
    {"parent-index", VSH_OT_INT, 0, N_("which parent object to search through")},
    {"config", VSH_OT_BOOL, 0, N_("affect next network startup")},
    {"live", VSH_OT_BOOL, 0, N_("affect running network")},
    {"current", VSH_OT_BOOL, 0, N_("affect current state of network")},
    {NULL, 0, 0, NULL}
}
```

#### 4.16.4.27 `const vshCmdOptDef opts_network_uuid[]` [static]

##### Initial value:

```
= {
    {"network", VSH_OT_DATA, VSH_OFLAG_REQ, N_("network name")},
    {NULL, 0, 0, NULL}
}
```



# Index

- [\\_qemuDomainPCIAddressSet](#), [7](#)
  - [nextslot](#), [7](#)
  - [used](#), [7](#)
- [\\_virBlkioDeviceWeight](#), [7](#)
  - [path](#), [7](#)
  - [weight](#), [7](#)
- [\\_virBlkioParameter](#)
  - [libvirt.h](#), [111](#)
- [\\_virConnectAuth](#), [8](#)
  - [cb](#), [8](#)
  - [cbdata](#), [8](#)
  - [credtype](#), [8](#)
  - [ncredtype](#), [8](#)
- [\\_virConnectCredential](#), [8](#)
  - [challenge](#), [8](#)
  - [defresult](#), [8](#)
  - [prompt](#), [8](#)
  - [result](#), [8](#)
  - [resultlen](#), [8](#)
  - [type](#), [8](#)
- [\\_virDeviceMonitor](#), [9](#)
  - [close](#), [9](#)
  - [deviceCreateXML](#), [9](#)
  - [deviceDestroy](#), [9](#)
  - [deviceGetParent](#), [9](#)
  - [deviceGetXMLDesc](#), [9](#)
  - [deviceListCaps](#), [9](#)
  - [deviceLookupByName](#), [9](#)
  - [deviceNumOfCaps](#), [9](#)
  - [listAllNodeDevices](#), [9](#)
  - [listDevices](#), [9](#)
  - [name](#), [10](#)
  - [numOfDevices](#), [10](#)
  - [open](#), [10](#)
- [\\_virDomainActualNetDef](#), [10](#)
  - [bandwidth](#), [10](#)
  - [bridge](#), [10](#)
  - [brname](#), [10](#)
  - [brtype](#), [10](#)
  - [data](#), [10](#)
  - [def](#), [10](#)
  - [direct](#), [10](#)
  - [hostdev](#), [10](#)
  - [linkdev](#), [11](#)
  - [mode](#), [11](#)
  - [type](#), [11](#)
  - [virtPortProfile](#), [11](#)
  - [vlan](#), [11](#)
- [\\_virDomainBIOSDef](#), [11](#)
  - [rt\\_delay](#), [11](#)
  - [rt\\_set](#), [11](#)
  - [useserial](#), [11](#)
- [\\_virDomainBlockInfo](#), [11](#)
  - [allocation](#), [12](#)
  - [capacity](#), [12](#)
  - [physical](#), [12](#)
- [\\_virDomainBlockIoTunelInfo](#), [12](#)
  - [read\\_bytes\\_sec](#), [12](#)
  - [read\\_iops\\_sec](#), [12](#)
  - [total\\_bytes\\_sec](#), [12](#)
  - [total\\_iops\\_sec](#), [12](#)
  - [write\\_bytes\\_sec](#), [12](#)
  - [write\\_iops\\_sec](#), [12](#)
- [\\_virDomainBlockJobInfo](#), [12](#)
  - [bandwidth](#), [13](#)
  - [cur](#), [13](#)
  - [end](#), [13](#)
  - [type](#), [13](#)
- [\\_virDomainBlockStats](#), [13](#)
  - [errs](#), [13](#)
  - [rd\\_bytes](#), [13](#)
  - [rd\\_req](#), [13](#)
  - [wr\\_bytes](#), [13](#)
  - [wr\\_req](#), [13](#)
- [\\_virDomainChrDef](#), [13](#)
  - [addr](#), [14](#)
  - [deviceType](#), [14](#)
  - [info](#), [14](#)
  - [name](#), [14](#)
  - [nseclabels](#), [14](#)
  - [port](#), [14](#)
  - [seclabels](#), [14](#)
  - [source](#), [14](#)
  - [target](#), [14](#)
  - [targetType](#), [14](#)
- [\\_virDomainChrSourceDef](#), [14](#)
  - [bindHost](#), [15](#)
  - [bindService](#), [15](#)
  - [connectHost](#), [15](#)
  - [connectService](#), [15](#)
  - [data](#), [15](#)
  - [file](#), [15](#)
  - [host](#), [15](#)
  - [listen](#), [15](#)
  - [nix](#), [15](#)
  - [path](#), [15](#)
  - [protocol](#), [15](#)
  - [service](#), [15](#)

- spicevmc, [15](#)
- tcp, [15](#)
- type, [15](#)
- udp, [15](#)
- \_virDomainClockDef, [16](#)
  - adjustment, [16](#)
  - basis, [16](#)
  - data, [16](#)
  - ntimers, [16](#)
  - offset, [16](#)
  - timers, [16](#)
  - timezone, [16](#)
  - utc\_reset, [16](#)
  - variable, [16](#)
- \_virDomainControllInfo, [16](#)
  - details, [17](#)
  - state, [17](#)
  - stateTime, [17](#)
- \_virDomainControllerDef, [17](#)
  - idx, [17](#)
  - info, [17](#)
  - model, [17](#)
  - opts, [17](#)
  - type, [17](#)
  - vioserial, [17](#)
- \_virDomainDef, [17](#)
  - apic\_eoi, [19](#)
  - blkio, [19](#)
  - channels, [19](#)
  - clock, [19](#)
  - consoles, [19](#)
  - controllers, [19](#)
  - cpu, [20](#)
  - cpumask, [20](#)
  - cputune, [20](#)
  - cur\_balloon, [20](#)
  - description, [20](#)
  - devices, [20](#)
  - disks, [20](#)
  - dump\_core, [20](#)
  - emulator, [20](#)
  - emulator\_period, [20](#)
  - emulator\_quota, [20](#)
  - emulatorpin, [20](#)
  - features, [20](#)
  - fss, [20](#)
  - graphics, [20](#)
  - hard\_limit, [20](#)
  - hostdevs, [20](#)
  - hubs, [20](#)
  - hugepage\_backed, [20](#)
  - id, [20](#)
  - inputs, [20](#)
  - leases, [20](#)
  - max\_balloon, [20](#)
  - maxvcpus, [20](#)
  - mem, [20](#)
  - memballoon, [20](#)
  - metadata, [20](#)
  - min\_guarantee, [20](#)
  - name, [21](#)
  - namespaceData, [21](#)
  - nchannels, [21](#)
  - nconsoles, [21](#)
  - ncontrollers, [21](#)
  - ndevices, [21](#)
  - ndisks, [21](#)
  - nets, [21](#)
  - nfss, [21](#)
  - ngraphics, [21](#)
  - nhostdevs, [21](#)
  - nhubs, [21](#)
  - ninputs, [21](#)
  - nleases, [21](#)
  - nnets, [21](#)
  - nparallels, [21](#)
  - nredirdevs, [21](#)
  - ns, [21](#)
  - nseclabels, [21](#)
  - nserials, [21](#)
  - nsmartcards, [21](#)
  - nsounds, [21](#)
  - numatune, [21](#)
  - nvcupin, [21](#)
  - nvideos, [21](#)
  - onCrash, [21](#)
  - onPoweroff, [21](#)
  - onReboot, [21](#)
  - os, [22](#)
  - parallels, [22](#)
  - period, [22](#)
  - placement\_mode, [22](#)
  - pm, [22](#)
  - quota, [22](#)
  - redirdevs, [22](#)
  - redirfilter, [22](#)
  - s3, [22](#)
  - s4, [22](#)
  - seclabels, [22](#)
  - serials, [22](#)
  - shares, [22](#)
  - smartcards, [22](#)
  - soft\_limit, [22](#)
  - sounds, [22](#)
  - swap\_hard\_limit, [22](#)
  - sysinfo, [22](#)
  - title, [22](#)
  - uuid, [22](#)
  - vcpupin, [22](#)
  - vcpus, [22](#)
  - videos, [22](#)
  - virtType, [22](#)
  - watchdog, [22](#)
  - weight, [22](#)
- \_virDomainDeviceCcidAddress, [23](#)
  - controller, [23](#)

- slot, [23](#)
- [\\_virDomainDeviceDef](#), [23](#)
  - chr, [23](#)
  - controller, [23](#)
  - data, [23](#)
  - disk, [24](#)
  - fs, [24](#)
  - graphics, [24](#)
  - hostdev, [24](#)
  - hub, [24](#)
  - input, [24](#)
  - lease, [24](#)
  - memballoon, [24](#)
  - net, [24](#)
  - redirdev, [24](#)
  - smartcard, [24](#)
  - sound, [24](#)
  - type, [24](#)
  - video, [24](#)
  - watchdog, [24](#)
- [\\_virDomainDeviceDriveAddress](#), [24](#)
  - bus, [24](#)
  - controller, [24](#)
  - target, [24](#)
  - unit, [25](#)
- [\\_virDomainDeviceInfo](#), [25](#)
  - addr, [25](#)
  - alias, [25](#)
  - bootIndex, [25](#)
  - ccid, [25](#)
  - drive, [25](#)
  - master, [25](#)
  - mastertype, [25](#)
  - pci, [25](#)
  - rombar, [25](#)
  - romfile, [25](#)
  - spaprpio, [26](#)
  - type, [26](#)
  - usb, [26](#)
  - vioserial, [26](#)
- [\\_virDomainDeviceSpaprVioAddress](#), [26](#)
  - has\_reg, [26](#)
  - reg, [26](#)
- [\\_virDomainDeviceUSBAddress](#), [26](#)
  - bus, [26](#)
  - port, [26](#)
- [\\_virDomainDeviceUSBMaster](#), [27](#)
  - startport, [27](#)
- [\\_virDomainDeviceVirtioSerialAddress](#), [27](#)
  - bus, [27](#)
  - controller, [27](#)
  - port, [27](#)
- [\\_virDomainDiskDef](#), [27](#)
  - auth, [29](#)
  - blkdeviotune, [29](#)
  - blockio, [29](#)
  - bus, [29](#)
  - cachemode, [29](#)
  - copy\_on\_read, [29](#)
  - cylinders, [29](#)
  - device, [29](#)
  - driverName, [29](#)
  - driverType, [29](#)
  - dst, [29](#)
  - encryption, [29](#)
  - error\_policy, [29](#)
  - event\_idx, [29](#)
  - geometry, [29](#)
  - heads, [29](#)
  - hosts, [29](#)
  - info, [29](#)
  - ioeventfd, [29](#)
  - iomode, [29](#)
  - logical\_block\_size, [29](#)
  - mirror, [29](#)
  - mirrorFormat, [29](#)
  - mirroring, [29](#)
  - nhosts, [29](#)
  - nseclabels, [29](#)
  - physical\_block\_size, [29](#)
  - protocol, [29](#)
  - rawio, [30](#)
  - rawio\_specified, [30](#)
  - readonly, [30](#)
  - rerror\_policy, [30](#)
  - seclabels, [30](#)
  - secret, [30](#)
  - secretType, [30](#)
  - sectors, [30](#)
  - serial, [30](#)
  - shared, [30](#)
  - snapshot, [30](#)
  - src, [30](#)
  - startupPolicy, [30](#)
  - trans, [30](#)
  - transient, [30](#)
  - tray\_status, [30](#)
  - type, [30](#)
  - usage, [30](#)
  - username, [30](#)
  - uuid, [30](#)
  - wwn, [30](#)
- [\\_virDomainDiskError](#), [30](#)
  - disk, [31](#)
  - error, [31](#)
- [\\_virDomainDiskHostDef](#), [31](#)
  - name, [31](#)
  - port, [31](#)
- [\\_virDomainEventGraphicsAddress](#), [31](#)
  - family, [32](#)
  - node, [32](#)
  - service, [32](#)
- [\\_virDomainEventGraphicsSubject](#), [32](#)
  - identities, [32](#)
  - nidentity, [32](#)
- [\\_virDomainEventGraphicsSubjectIdentity](#), [32](#)

- name, 33
- type, 33
- \_virDomainFSDef, 33
  - accessmode, 33
  - dst, 33
  - fsdriver, 33
  - info, 33
  - readonly, 33
  - space\_hard\_limit, 33
  - space\_soft\_limit, 33
  - src, 33
  - type, 34
  - usage, 34
  - wrpolicy, 34
- \_virDomainGraphicsAuthDef, 34
  - connected, 34
  - expires, 34
  - passwd, 34
  - validTo, 34
- \_virDomainGraphicsDef, 34
  - auth, 35
  - autoport, 35
  - channels, 35
  - copypaste, 35
  - data, 35
  - defaultMode, 35
  - desktop, 35
  - display, 35
  - fullscreen, 35
  - image, 36
  - jpeg, 36
  - keymap, 36
  - listens, 36
  - mousemode, 36
  - multiUser, 36
  - nListens, 36
  - playback, 36
  - port, 36
  - rdp, 36
  - replaceUser, 36
  - sdl, 36
  - socket, 36
  - spice, 36
  - streaming, 36
  - tlsPort, 36
  - type, 36
  - vnc, 36
  - xauth, 36
  - zlib, 36
- \_virDomainGraphicsListenDef, 36
  - address, 37
  - network, 37
  - type, 37
- \_virDomainHostdevDef, 37
  - caps, 37
  - dummy, 37
  - info, 37
  - managed, 37
  - mode, 37
  - origstates, 37
  - parent, 37
  - source, 37
  - subsys, 37
- \_virDomainHostdevOrigStates, 38
  - pci, 38
  - remove\_slot, 38
  - reprobe, 38
  - states, 38
  - unbind\_from\_stub, 38
- \_virDomainHostdevSubsys, 38
  - bus, 38
  - device, 38
  - pci, 39
  - product, 39
  - type, 39
  - u, 39
  - usb, 39
  - vendor, 39
- \_virDomainHubDef, 39
  - info, 39
  - type, 39
- \_virDomainInfo, 39
  - cpuTime, 40
  - maxMem, 40
  - memory, 40
  - nrVirtCpu, 40
  - state, 40
- \_virDomainInputDef, 40
  - bus, 40
  - info, 40
  - type, 40
- \_virDomainInterfaceStats, 40
  - rx\_bytes, 41
  - rx\_drop, 41
  - rx\_errs, 41
  - rx\_packets, 41
  - tx\_bytes, 41
  - tx\_drop, 41
  - tx\_errs, 41
  - tx\_packets, 41
- \_virDomainJobInfo, 41
  - dataProcessed, 41
  - dataRemaining, 41
  - dataTotal, 41
  - fileProcessed, 41
  - fileRemaining, 41
  - fileTotal, 41
  - memProcessed, 42
  - memRemaining, 42
  - memTotal, 42
  - timeElapsed, 42
  - timeRemaining, 42
  - type, 42
- \_virDomainLeaseDef, 42
  - key, 42
  - lockspace, 42

- offset, [42](#)
- path, [42](#)
- [\\_virDomainMemballoonDef](#), [42](#)
  - info, [43](#)
  - model, [43](#)
- [\\_virDomainMemoryStat](#), [43](#)
  - tag, [43](#)
  - val, [43](#)
- [\\_virDomainNetDef](#), [43](#)
  - actual, [44](#)
  - address, [44](#)
  - bandwidth, [44](#)
  - bridge, [44](#)
  - brname, [44](#)
  - brtype, [44](#)
  - data, [44](#)
  - def, [45](#)
  - dev, [45](#)
  - direct, [45](#)
  - driver, [45](#)
  - ethernet, [45](#)
  - event\_idx, [45](#)
  - filter, [45](#)
  - filterparams, [45](#)
  - hostdev, [45](#)
  - ifname, [45](#)
  - info, [45](#)
  - internal, [45](#)
  - ioeventfd, [45](#)
  - ipaddr, [45](#)
  - linkdev, [45](#)
  - linkstate, [45](#)
  - mac, [45](#)
  - mode, [45](#)
  - model, [45](#)
  - name, [45](#)
  - network, [45](#)
  - port, [45](#)
  - portgroup, [45](#)
  - script, [45](#)
  - sndbuf, [45](#)
  - sndbuf\_specified, [45](#)
  - socket, [45](#)
  - tune, [46](#)
  - txmode, [46](#)
  - type, [46](#)
  - virtPortProfile, [46](#)
  - virtio, [46](#)
  - vlan, [46](#)
- [\\_virDomainNumatuneDef](#), [46](#)
  - memory, [46](#)
  - mode, [46](#)
  - nodemask, [46](#)
  - placement\_mode, [46](#)
- [\\_virDomainOSDef](#), [48](#)
  - arch, [48](#)
  - bios, [48](#)
  - bootDevs, [48](#)
  - bootloader, [48](#)
  - bootloaderArgs, [48](#)
  - bootmenu, [48](#)
  - cmdline, [48](#)
  - init, [48](#)
  - initargv, [49](#)
  - initrd, [49](#)
  - kernel, [49](#)
  - loader, [49](#)
  - machine, [49](#)
  - nBootDevs, [49](#)
  - root, [49](#)
  - smbios\_mode, [49](#)
  - type, [49](#)
- [\\_virDomainObj](#), [46](#)
  - autostart, [47](#)
  - current\_snapshot, [47](#)
  - def, [47](#)
  - hasManagedSave, [47](#)
  - lock, [47](#)
  - newDef, [47](#)
  - object, [47](#)
  - persistent, [47](#)
  - pid, [47](#)
  - privateData, [47](#)
  - privateDataFreeFunc, [47](#)
  - snapshots, [47](#)
  - state, [47](#)
  - taint, [47](#)
  - updated, [47](#)
- [\\_virDomainObjList](#), [47](#)
  - objs, [48](#)
- [\\_virDomainRedirFilterDef](#), [50](#)
  - nusbdevs, [50](#)
  - usbdevs, [50](#)
- [\\_virDomainRedirFilterUsbDevDef](#), [50](#)
  - allow, [50](#)
  - product, [50](#)
  - usbClass, [50](#)
  - vendor, [50](#)
  - version, [50](#)
- [\\_virDomainRedirdevDef](#), [49](#)
  - bus, [49](#)
  - chr, [49](#)
  - info, [49](#)
  - source, [49](#)
- [\\_virDomainSmartcardDef](#), [50](#)
  - cert, [51](#)
  - data, [51](#)
  - database, [51](#)
  - file, [51](#)
  - info, [51](#)
  - passthru, [51](#)
  - type, [51](#)
- [\\_virDomainSoundCodecDef](#), [51](#)
  - cad, [51](#)
  - type, [51](#)
- [\\_virDomainSoundDef](#), [52](#)

- codecs, [52](#)
- info, [52](#)
- model, [52](#)
- ncodecs, [52](#)
- \_virDomainStateReason, [52](#)
  - reason, [52](#)
  - state, [52](#)
- \_virDomainTimerCatchupDef, [52](#)
  - limit, [53](#)
  - slew, [53](#)
  - threshold, [53](#)
- \_virDomainTimerDef, [53](#)
  - catchup, [53](#)
  - frequency, [53](#)
  - mode, [53](#)
  - name, [53](#)
  - present, [53](#)
  - tickpolicy, [53](#)
  - track, [53](#)
- \_virDomainVcpuPinDef, [54](#)
  - cpumask, [54](#)
  - vcpuid, [54](#)
- \_virDomainVideoAccelDef, [54](#)
  - support2d, [54](#)
  - support3d, [54](#)
- \_virDomainVideoDef, [54](#)
  - accel, [55](#)
  - heads, [55](#)
  - info, [55](#)
  - type, [55](#)
  - vram, [55](#)
- \_virDomainVirtioSerialOpts, [55](#)
  - ports, [55](#)
  - vectors, [55](#)
- \_virDomainWatchdogDef, [55](#)
  - action, [55](#)
  - info, [55](#)
  - model, [55](#)
- \_virDriver, [56](#)
  - close, [59](#)
  - cpuBaseline, [59](#)
  - cpuCompare, [59](#)
  - domainAbortJob, [59](#)
  - domainAttachDevice, [59](#)
  - domainAttachDeviceFlags, [59](#)
  - domainBlockCommit, [59](#)
  - domainBlockJobAbort, [60](#)
  - domainBlockJobSetSpeed, [60](#)
  - domainBlockPeek, [60](#)
  - domainBlockPull, [60](#)
  - domainBlockRebase, [60](#)
  - domainBlockResize, [60](#)
  - domainBlockStats, [60](#)
  - domainBlockStatsFlags, [60](#)
  - domainCoreDump, [60](#)
  - domainCreate, [60](#)
  - domainCreateWithFlags, [60](#)
  - domainCreateXML, [60](#)
  - domainDefineXML, [60](#)
  - domainDestroy, [60](#)
  - domainDestroyFlags, [60](#)
  - domainDetachDevice, [60](#)
  - domainDetachDeviceFlags, [60](#)
  - domainEventDeregister, [60](#)
  - domainEventDeregisterAny, [60](#)
  - domainEventRegister, [60](#)
  - domainEventRegisterAny, [60](#)
  - domainGetAutostart, [60](#)
  - domainGetBlkioParameters, [60](#)
  - domainGetBlockInfo, [60](#)
  - domainGetBlockIoTune, [60](#)
  - domainGetBlockJobInfo, [60](#)
  - domainGetCPUStats, [60](#)
  - domainGetControllInfo, [60](#)
  - domainGetDiskErrors, [61](#)
  - domainGetEmulatorPinInfo, [61](#)
  - domainGetHostname, [61](#)
  - domainGetInfo, [61](#)
  - domainGetInterfaceParameters, [61](#)
  - domainGetJobInfo, [61](#)
  - domainGetMaxMemory, [61](#)
  - domainGetMaxVcpus, [61](#)
  - domainGetMemoryParameters, [61](#)
  - domainGetMetadata, [61](#)
  - domainGetNumaParameters, [61](#)
  - domainGetOSType, [61](#)
  - domainGetSchedulerParameters, [61](#)
  - domainGetSchedulerParametersFlags, [61](#)
  - domainGetSchedulerType, [61](#)
  - domainGetSecurityLabel, [61](#)
  - domainGetSecurityLabelList, [61](#)
  - domainGetState, [61](#)
  - domainGetVcpuPinInfo, [61](#)
  - domainGetVcpus, [61](#)
  - domainGetVcpusFlags, [61](#)
  - domainGetXMLDesc, [61](#)
  - domainHasCurrentSnapshot, [61](#)
  - domainHasManagedSaveImage, [61](#)
  - domainInjectNMI, [61](#)
  - domainInterfaceStats, [61](#)
  - domainsIsActive, [61](#)
  - domainsIsPersistent, [61](#)
  - domainsIsUpdated, [62](#)
  - domainListAllSnapshots, [62](#)
  - domainLookupByID, [62](#)
  - domainLookupByName, [62](#)
  - domainLookupByUUID, [62](#)
  - domainManagedSave, [62](#)
  - domainManagedSaveRemove, [62](#)
  - domainMemoryPeek, [62](#)
  - domainMemoryStats, [62](#)
  - domainMigrateBegin3, [62](#)
  - domainMigrateConfirm3, [62](#)
  - domainMigrateFinish, [62](#)
  - domainMigrateFinish2, [62](#)
  - domainMigrateFinish3, [62](#)

- domainMigrateGetMaxSpeed, 62
- domainMigratePerform, 62
- domainMigratePerform3, 62
- domainMigratePrepare, 62
- domainMigratePrepare2, 62
- domainMigratePrepare3, 62
- domainMigratePrepareTunnel, 62
- domainMigratePrepareTunnel3, 62
- domainMigrateSetMaxDowntime, 62
- domainMigrateSetMaxSpeed, 62
- domainOpenConsole, 62
- domainOpenGraphics, 62
- domainPMSuspendForDuration, 63
- domainPMWakeup, 63
- domainPinEmulator, 62
- domainPinVcpu, 62
- domainPinVcpuFlags, 63
- domainReboot, 63
- domainReset, 63
- domainRestore, 63
- domainRestoreFlags, 63
- domainResume, 63
- domainRevertToSnapshot, 63
- domainSave, 63
- domainSaveFlags, 63
- domainSaveImageDefineXML, 63
- domainSaveImageGetXMLDesc, 63
- domainScreenshot, 63
- domainSendKey, 63
- domainSetAutostart, 63
- domainSetBlkioParameters, 63
- domainSetBlockIoTune, 63
- domainSetInterfaceParameters, 63
- domainSetMaxMemory, 63
- domainSetMemory, 63
- domainSetMemoryFlags, 63
- domainSetMemoryParameters, 63
- domainSetMetadata, 63
- domainSetNumaParameters, 63
- domainSetSchedulerParameters, 63
- domainSetSchedulerParametersFlags, 63
- domainSetVcpus, 63
- domainSetVcpusFlags, 64
- domainShutdown, 64
- domainShutdownFlags, 64
- domainSnapshotCreateXML, 64
- domainSnapshotCurrent, 64
- domainSnapshotDelete, 64
- domainSnapshotGetParent, 64
- domainSnapshotGetXMLDesc, 64
- domainSnapshotHasMetadata, 64
- domainSnapshotIsCurrent, 64
- domainSnapshotListAllChildren, 64
- domainSnapshotListChildrenNames, 64
- domainSnapshotListNames, 64
- domainSnapshotLookupByName, 64
- domainSnapshotNum, 64
- domainSnapshotNumChildren, 64
- domainSuspend, 64
- domainUndefine, 64
- domainUndefineFlags, 64
- domainUpdateDeviceFlags, 64
- domainXMLFromNative, 64
- domainXMLToNative, 64
- getCapabilities, 64
- getHostname, 64
- getMaxVcpus, 64
- getSysinfo, 64
- isAlive, 64
- isEncrypted, 64
- isSecure, 65
- libvirtVersion, 65
- listAllDomains, 65
- listDefinedDomains, 65
- listDomains, 65
- name, 65
- no, 65
- nodeDeviceDetach, 65
- nodeDeviceReAttach, 65
- nodeDeviceReset, 65
- nodeGetCPUStats, 65
- nodeGetCellsFreeMemory, 65
- nodeGetFreeMemory, 65
- nodeGetInfo, 65
- nodeGetMemoryParameters, 65
- nodeGetMemoryStats, 65
- nodeGetSecurityModel, 65
- nodeSetMemoryParameters, 65
- nodeSuspendForDuration, 65
- numOfDefinedDomains, 65
- numOfDomains, 65
- open, 65
- qemuDomainArbitraryAgentCommand, 65
- qemuDomainAttach, 65
- qemuDomainMonitorCommand, 65
- setKeepAlive, 65
- supports\_feature, 65
- type, 65
- version, 66
- \_virInterfaceDriver, 66
  - close, 66
  - interfaceChangeBegin, 66
  - interfaceChangeCommit, 66
  - interfaceChangeRollback, 66
  - interfaceCreate, 67
  - interfaceDefineXML, 67
  - interfaceDestroy, 67
  - interfaceGetXMLDesc, 67
  - interfacelsActive, 67
  - interfaceLookupByMACString, 67
  - interfaceLookupByName, 67
  - interfaceUndefine, 67
  - listAllInterfaces, 67
  - listDefinedInterfaces, 67
  - listInterfaces, 67
  - name, 67

- numOfDefinedInterfaces, 67
- numOfInterfaces, 67
- open, 67
- \_virMemoryParameter
  - libvirt.h, 111
- \_virNWFilterDriver, 78
  - close, 79
  - defineXML, 79
  - getXMLDesc, 79
  - listAllNWFilters, 79
  - listNWFilters, 79
  - name, 79
  - numOfNWFilters, 79
  - nwfilterLookupByName, 79
  - nwfilterLookupByUUID, 79
  - open, 79
  - undefine, 79
- \_virNetworkDHCPHostDef, 69
  - ip, 69
  - mac, 69
  - name, 69
- \_virNetworkDHCPRangeDef, 69
  - end, 70
  - start, 70
- \_virNetworkDNSDef, 70
  - hosts, 70
  - nhosts, 70
  - nsrvrecords, 70
  - ntxtrecords, 70
  - srvrecords, 70
  - txtrecords, 70
- \_virNetworkDNSHostsDef, 70
  - ip, 71
  - names, 71
  - nnames, 71
- \_virNetworkDNSSrvRecordsDef, 71
  - domain, 71
  - port, 71
  - priority, 71
  - protocol, 71
  - service, 71
  - target, 71
  - weight, 71
- \_virNetworkDNSTxtRecordsDef, 72
  - name, 72
  - value, 72
- \_virNetworkDef, 67
  - bandwidth, 68
  - bridge, 68
  - bridge\_type, 68
  - connections, 68
  - delay, 68
  - dns, 68
  - domain, 68
  - forwardIfs, 68
  - forwardPfs, 68
  - forwardType, 68
  - ips, 68
  - mac, 68
  - mac\_specified, 68
  - managed, 68
  - nForwardIfs, 68
  - nForwardPfs, 68
  - nPortGroups, 68
  - nTunnels, 69
  - name, 68
  - nips, 68
  - portGroups, 69
  - source\_bridge, 69
  - stp, 69
  - tunnels, 69
  - uuid, 69
  - uuid\_specified, 69
  - virtPortProfile, 69
  - vlan, 69
- \_virNetworkDriver, 72
  - close, 73
  - listAllNetworks, 73
  - listDefinedNetworks, 73
  - listNetworks, 73
  - name, 73
  - networkCreate, 73
  - networkCreateXML, 73
  - networkDefineXML, 73
  - networkDestroy, 73
  - networkGetAutostart, 73
  - networkGetBridgeName, 73
  - networkGetBridgeType, 73
  - networkGetXMLDesc, 73
  - networkIsActive, 73
  - networkIsPersistent, 73
  - networkLookupByName, 73
  - networkLookupByUUID, 73
  - networkSetAutostart, 73
  - networkUndefine, 73
  - networkUpdate, 73
  - numOfDefinedNetworks, 73
  - numOfNetworks, 73
  - open, 73
- \_virNetworkForwardIfDef, 74
  - connections, 74
  - dev, 74
  - device, 74
  - pci, 74
  - type, 74
- \_virNetworkForwardPfDef, 74
  - connections, 74
  - dev, 74
- \_virNetworkIpDef, 75
  - address, 75
  - bootfile, 75
  - bootserver, 75
  - family, 75
  - hosts, 75
  - netmask, 75
  - nhosts, 75



- nranges, [75](#)
- prefix, [75](#)
- ranges, [75](#)
- tftproot, [75](#)
- `_virNetworkObj`, [76](#)
  - active, [76](#)
  - autostart, [76](#)
  - def, [76](#)
  - dnsmasqPid, [76](#)
  - lock, [76](#)
  - newDef, [76](#)
  - persistent, [76](#)
  - radvdPid, [76](#)
- `_virNetworkObjList`, [76](#)
  - count, [76](#)
  - objs, [76](#)
- `_virNodeCPUStats`, [77](#)
  - field, [77](#)
  - value, [77](#)
- `_virNodeInfo`, [77](#)
  - cores, [77](#)
  - cpus, [77](#)
  - memory, [77](#)
  - mhz, [77](#)
  - model, [77](#)
  - nodes, [78](#)
  - sockets, [78](#)
  - threads, [78](#)
- `_virNodeMemoryStats`, [78](#)
  - field, [78](#)
  - value, [78](#)
- `_virPortGroupDef`, [79](#)
  - bandwidth, [80](#)
  - isDefault, [80](#)
  - name, [80](#)
  - virtPortProfile, [80](#)
  - vlan, [80](#)
- `_virSchedParameter`
  - libvirt.h, [111](#)
- `_virSecretDriver`, [80](#)
  - close, [80](#)
  - defineXML, [80](#)
  - getValue, [80](#)
  - getXMLDesc, [81](#)
  - listAllSecrets, [81](#)
  - listSecrets, [81](#)
  - lookupByUUID, [81](#)
  - lookupByUsage, [81](#)
  - name, [81](#)
  - numOfSecrets, [81](#)
  - open, [81](#)
  - setValue, [81](#)
  - undefine, [81](#)
- `_virSecurityDeviceLabelDef`, [81](#)
  - label, [81](#)
  - model, [81](#)
  - norelabel, [81](#)
- `_virSecurityLabel`, [81](#)
  - enforcing, [82](#)
  - label, [82](#)
- `_virSecurityLabelDef`, [82](#)
  - baselabel, [82](#)
  - imagelabel, [82](#)
  - implicit, [82](#)
  - label, [82](#)
  - model, [82](#)
  - norelabel, [82](#)
  - type, [82](#)
- `_virSecurityModel`, [83](#)
  - doi, [83](#)
  - model, [83](#)
- `_virStorageDriver`, [83](#)
  - close, [84](#)
  - findPoolSources, [84](#)
  - listAllPools, [84](#)
  - listDefinedPools, [84](#)
  - listPools, [84](#)
  - name, [84](#)
  - numOfDefinedPools, [84](#)
  - numOfPools, [84](#)
  - open, [85](#)
  - poolBuild, [85](#)
  - poolCreate, [85](#)
  - poolCreateXML, [85](#)
  - poolDefineXML, [85](#)
  - poolDelete, [85](#)
  - poolDestroy, [85](#)
  - poolGetAutostart, [85](#)
  - poolGetInfo, [85](#)
  - poolGetXMLDesc, [85](#)
  - poolIsActive, [85](#)
  - poolIsPersistent, [85](#)
  - poolListAllVolumes, [85](#)
  - poolListVolumes, [85](#)
  - poolLookupByName, [85](#)
  - poolLookupByUUID, [85](#)
  - poolLookupByVolume, [85](#)
  - poolNumOfVolumes, [85](#)
  - poolRefresh, [85](#)
  - poolSetAutostart, [85](#)
  - poolUndefine, [85](#)
  - volCreateXML, [85](#)
  - volCreateXMLFrom, [85](#)
  - volDelete, [85](#)
  - volDownload, [85](#)
  - volGetInfo, [85](#)
  - volGetPath, [85](#)
  - volGetXMLDesc, [85](#)
  - volLookupByKey, [86](#)
  - volLookupByName, [86](#)
  - volLookupByPath, [86](#)
  - volResize, [86](#)
  - volUpload, [86](#)
  - volWipe, [86](#)
  - volWipePattern, [86](#)
- `_virStoragePoolInfo`, [86](#)

- allocation, 86
- available, 86
- capacity, 86
- state, 86
- \_virStorageVollInfo, 86
  - allocation, 87
  - capacity, 87
  - type, 87
- \_virStreamDriver, 87
  - streamAbort, 87
  - streamAddCallback, 87
  - streamFinish, 87
  - streamRecv, 87
  - streamRemoveCallback, 87
  - streamSend, 87
  - streamUpdateCallback, 87
- \_virTunnelDef, 88
  - device, 88
  - remote\_ip, 88
  - trunks, 88
- \_virTypedParameter, 88
  - b, 88
  - d, 88
  - field, 88
  - i, 88
  - l, 88
  - s, 89
  - type, 89
  - ui, 89
  - ul, 89
  - value, 89
- \_virVcpuInfo, 89
  - cpu, 89
  - cpuTime, 89
  - number, 89
  - state, 89
- ATTRIBUTE\_NONNULL
  - domain\_conf.c, 236
  - virnetdevopenvswitch.h, 444
- accel
  - \_virDomainVideoDef, 55
- accessmode
  - \_virDomainFSDef, 33
- action
  - \_virDomainWatchdogDef, 55
- active
  - \_virNetworkObj, 76
- actual
  - \_virDomainNetDef, 44
- addr
  - \_virDomainChrDef, 14
  - \_virDomainDeviceInfo, 25
- address
  - \_virDomainGraphicsListenDef, 37
  - \_virDomainNetDef, 44
  - \_virNetworkIpDef, 75
- adjustment
  - \_virDomainClockDef, 16
- alias
  - \_virDomainDeviceInfo, 25
- allocation
  - \_virDomainBlockInfo, 12
  - \_virStoragePoolInfo, 86
  - \_virStorageVollInfo, 87
- allow
  - \_virDomainRedirFilterUsbDevDef, 50
- apic\_eoi
  - \_virDomainDef, 19
- arch
  - \_virDomainOSDef, 48
- auth
  - \_virDomainDiskDef, 29
  - \_virDomainGraphicsDef, 35
- autoport
  - \_virDomainGraphicsDef, 35
- autostart
  - \_virDomainObj, 47
  - \_virNetworkObj, 76
- available
  - \_virStoragePoolInfo, 86
- b
  - \_virTypedParameter, 88
- bandwidth
  - \_virDomainActualNetDef, 10
  - \_virDomainBlockJobInfo, 13
  - \_virDomainNetDef, 44
  - \_virNetworkDef, 68
  - \_virPortGroupDef, 80
- baselabel
  - \_virSecurityLabelDef, 82
- basis
  - \_virDomainClockDef, 16
- bindHost
  - \_virDomainChrSourceDef, 15
- bindService
  - \_virDomainChrSourceDef, 15
- bios
  - \_virDomainOSDef, 48
- blkdeviotune
  - \_virDomainDiskDef, 29
- blkio
  - \_virDomainDef, 19
- blockio
  - \_virDomainDiskDef, 29
- bootDevs
  - \_virDomainOSDef, 48
- bootIndex
  - \_virDomainDeviceInfo, 25
- bootfile
  - \_virNetworkIpDef, 75
- bootloader
  - \_virDomainOSDef, 48
- bootloaderArgs
  - \_virDomainOSDef, 48
- bootmenu
  - \_virDomainOSDef, 48

- bootserver
  - \_virNetworkIpDef, 75
- bridge
  - \_virDomainActualNetDef, 10
  - \_virDomainNetDef, 44
  - \_virNetworkDef, 68
- bridge\_driver.c
  - DNSMASQ\_STATE\_DIR, 424
  - driverState, 428
  - NETWORK\_PID\_DIR, 424
  - NETWORK\_STATE\_DIR, 424
  - networkActive, 424
  - networkAddAddrToBridge, 424
  - networkAddGeneralIp6tablesRules, 424
  - networkAddGeneralIptablesRules, 424
  - networkAddIpSpecificIptablesRules, 424
  - networkAddIptablesRules, 424
  - networkAddMasqueradingIptablesRules, 424
  - networkAddRoutingIptablesRules, 424
  - networkAllocateActualDevice, 424
  - networkAutostartConfigs, 424
  - networkBridgeDummyNicName, 424
  - networkBuildDhcpDaemonCommandLine, 424
  - networkBuildDnsmasqArgv, 424
  - networkBuildDnsmasqHostsfile, 424
  - networkCheckRouteCollision, 425
  - networkCloseNetwork, 425
  - networkCreate, 425
  - networkCreateInterfacePool, 425
  - networkDefine, 425
  - networkDestroy, 425
  - networkDnsmasqLeaseFileName, 428
  - networkDnsmasqLeaseFileNameDefault, 425
  - networkDriver, 428
  - networkDriverLock, 425
  - networkDriverUnlock, 425
  - networkEnableIpForwarding, 425
  - networkFindActiveConfigs, 425
  - networkGetAutostart, 425
  - networkGetBridgeName, 425
  - networkGetBridgeType, 425
  - networkGetNetworkAddress, 425
  - networkGetXMLDesc, 425
  - networkIsActive, 425
  - networkIsPersistent, 425
  - networkKillDaemon, 425
  - networkListAllNetworks, 425
  - networkListDefinedNetworks, 425
  - networkListNetworks, 425
  - networkLookupByName, 425
  - networkLookupByUUID, 426
  - networkNotifyActualDevice, 426
  - networkNumDefinedNetworks, 426
  - networkNumNetworks, 426
  - networkOpenNetwork, 426
  - networkRadvdConfContents, 426
  - networkRadvdConfWrite, 426
  - networkRadvdConfigFileName, 426
  - networkRadvdPidfileBasename, 426
  - networkRefreshDaemons, 426
  - networkRefreshDhcpDaemon, 426
  - networkRefreshRadvd, 426
  - networkRegister, 426
  - networkReleaseActualDevice, 426
  - networkReload, 426
  - networkReloadIptablesRules, 426
  - networkRemoveGeneralIp6tablesRules, 426
  - networkRemoveGeneralIptablesRules, 426
  - networkRemoveInactive, 426
  - networkRemoveIpSpecificIptablesRules, 426
  - networkRemoveIptablesRules, 426
  - networkRemoveMasqueradingIptablesRules, 426
  - networkRemoveRoutingIptablesRules, 427
  - networkRestartDhcpDaemon, 427
  - networkSetAutostart, 427
  - networkSetIPv6Sysctls, 427
  - networkShutdown, 427
  - networkShutdownNetwork, 427
  - networkShutdownNetworkExternal, 427
  - networkShutdownNetworkVirtual, 427
  - networkStart, 427
  - networkStartDhcpDaemon, 427
  - networkStartNetwork, 427
  - networkStartNetworkExternal, 427
  - networkStartNetworkVirtual, 427
  - networkStartRadvd, 428
  - networkStartup, 428
  - networkStateDriver, 429
  - networkUndefine, 428
  - networkUpdate, 428
  - networkValidate, 428
  - PROC\_NET\_ROUTE, 424
  - RADVD\_STATE\_DIR, 424
  - SYSCTL\_PATH, 424
  - VIR\_FROM\_THIS, 424
- bridge\_type
  - \_virNetworkDef, 68
- brname
  - \_virDomainActualNetDef, 10
  - \_virDomainNetDef, 44
- brtype
  - \_virDomainActualNetDef, 10
  - \_virDomainNetDef, 44
- bus
  - \_virDomainDeviceDriveAddress, 24
  - \_virDomainDeviceUSBAddress, 26
  - \_virDomainDeviceVirtioSerialAddress, 27
  - \_virDomainDiskDef, 29
  - \_virDomainHostdevSubsys, 38
  - \_virDomainInputDef, 40
  - \_virDomainRedirdevDef, 49
- cachemode
  - \_virDomainDiskDef, 29
- cad
  - \_virDomainSoundCodecDef, 51
- capacity

- [\\_virDomainBlockInfo](#), 12
  - [\\_virStoragePoolInfo](#), 86
  - [\\_virStorageVolInfo](#), 87
- caps
  - [\\_virDomainHostdevDef](#), 37
- catchup
  - [\\_virDomainTimerDef](#), 53
- cb
  - [\\_virConnectAuth](#), 8
- cbdata
  - [\\_virConnectAuth](#), 8
- ccid
  - [\\_virDomainDeviceInfo](#), 25
- cert
  - [\\_virDomainSmartcardDef](#), 51
- challenge
  - [\\_virConnectCredential](#), 8
- channels
  - [\\_virDomainDef](#), 19
  - [\\_virDomainGraphicsDef](#), 35
- chr
  - [\\_virDomainDeviceDef](#), 23
  - [\\_virDomainRedirdevDef](#), 49
- clock
  - [\\_virDomainDef](#), 19
- close
  - [\\_virDeviceMonitor](#), 9
  - [\\_virDriver](#), 59
  - [\\_virInterfaceDriver](#), 66
  - [\\_virNWFilterDriver](#), 79
  - [\\_virNetworkDriver](#), 73
  - [\\_virSecretDriver](#), 80
  - [\\_virStorageDriver](#), 84
- cmdNetworkAutostart
  - [virsh-network.c](#), 450
- cmdNetworkCreate
  - [virsh-network.c](#), 450
- cmdNetworkDefine
  - [virsh-network.c](#), 450
- cmdNetworkDestroy
  - [virsh-network.c](#), 450
- cmdNetworkDumpXML
  - [virsh-network.c](#), 450
- cmdNetworkEdit
  - [virsh-network.c](#), 450
- cmdNetworkInfo
  - [virsh-network.c](#), 450
- cmdNetworkList
  - [virsh-network.c](#), 451
- cmdNetworkName
  - [virsh-network.c](#), 451
- cmdNetworkStart
  - [virsh-network.c](#), 451
- cmdNetworkUndefine
  - [virsh-network.c](#), 451
- cmdNetworkUpdate
  - [virsh-network.c](#), 451
- cmdNetworkUuid
  - [virsh-network.c](#), 451
- cmdline
  - [\\_virDomainOSDef](#), 48
- codecs
  - [\\_virDomainSoundDef](#), 52
- conn
  - [virDomainListData](#), 91
- connectHost
  - [\\_virDomainChrSourceDef](#), 15
- connectService
  - [\\_virDomainChrSourceDef](#), 15
- connected
  - [\\_virDomainGraphicsAuthDef](#), 34
- connections
  - [\\_virNetworkDef](#), 68
  - [\\_virNetworkForwardIfDef](#), 74
  - [\\_virNetworkForwardPfDef](#), 74
- consoles
  - [\\_virDomainDef](#), 19
- controller
  - [\\_virDomainDeviceCcidAddress](#), 23
  - [\\_virDomainDeviceDef](#), 23
  - [\\_virDomainDeviceDriveAddress](#), 24
  - [\\_virDomainDeviceVirtioSerialAddress](#), 27
- controllers
  - [\\_virDomainDef](#), 19
- copy\_on\_read
  - [\\_virDomainDiskDef](#), 29
- copypaste
  - [\\_virDomainGraphicsDef](#), 35
- cores
  - [\\_virNodeInfo](#), 77
- count
  - [\\_virNetworkObjList](#), 76
- cpu
  - [\\_virDomainDef](#), 20
  - [\\_virVcpuInfo](#), 89
- cpuBaseline
  - [\\_virDriver](#), 59
- cpuCompare
  - [\\_virDriver](#), 59
- cpuTime
  - [\\_virDomainInfo](#), 40
  - [\\_virVcpuInfo](#), 89
- cpumask
  - [\\_virDomainDef](#), 20
  - [\\_virDomainVcpuPinDef](#), 54
- cpus
  - [\\_virNodeInfo](#), 77
- cputune
  - [\\_virDomainDef](#), 20
- credtype
  - [\\_virConnectAuth](#), 8
- cur
  - [\\_virDomainBlockJobInfo](#), 13
- cur\_balloon
  - [\\_virDomainDef](#), 20
- current\_snapshot

- [\\_virDomainObj](#), 47
- cylinders
  - [\\_virDomainDiskDef](#), 29
- d
  - [\\_virTypedParameter](#), 88
- DNSMASQ\_STATE\_DIR
  - [bridge\\_driver.c](#), 424
- DUMPXML\_FLAGS
  - [domain\\_conf.c](#), 235
- data
  - [\\_virDomainActualNetDef](#), 10
  - [\\_virDomainChrSourceDef](#), 15
  - [\\_virDomainClockDef](#), 16
  - [\\_virDomainDeviceDef](#), 23
  - [\\_virDomainGraphicsDef](#), 35
  - [\\_virDomainNetDef](#), 44
  - [\\_virDomainSmartcardDef](#), 51
- dataProcessed
  - [\\_virDomainJobInfo](#), 41
- dataRemaining
  - [\\_virDomainJobInfo](#), 41
- dataTotal
  - [\\_virDomainJobInfo](#), 41
- database
  - [\\_virDomainSmartcardDef](#), 51
- def
  - [\\_virDomainActualNetDef](#), 10
  - [\\_virDomainNetDef](#), 45
  - [\\_virDomainObj](#), 47
  - [\\_virNetworkObj](#), 76
- defaultMode
  - [\\_virDomainGraphicsDef](#), 35
- defineXML
  - [\\_virNWFilterDriver](#), 79
  - [\\_virSecretDriver](#), 80
- defresult
  - [\\_virConnectCredential](#), 8
- delay
  - [\\_virNetworkDef](#), 68
- description
  - [\\_virDomainDef](#), 20
- desktop
  - [\\_virDomainGraphicsDef](#), 35
- details
  - [\\_virDomainControllInfo](#), 17
- dev
  - [\\_virDomainNetDef](#), 45
  - [\\_virNetworkForwardIfDef](#), 74
  - [\\_virNetworkForwardPfDef](#), 74
- device
  - [\\_virDomainDiskDef](#), 29
  - [\\_virDomainHostdevSubsys](#), 38
  - [\\_virNetworkForwardIfDef](#), 74
  - [\\_virTunnelDef](#), 88
- deviceCreateXML
  - [\\_virDeviceMonitor](#), 9
- deviceDestroy
  - [\\_virDeviceMonitor](#), 9
- deviceGetParent
  - [\\_virDeviceMonitor](#), 9
- deviceGetXMLDesc
  - [\\_virDeviceMonitor](#), 9
- deviceListCaps
  - [\\_virDeviceMonitor](#), 9
- deviceLookupByName
  - [\\_virDeviceMonitor](#), 9
- deviceNumOfCaps
  - [\\_virDeviceMonitor](#), 9
- deviceType
  - [\\_virDomainChrDef](#), 14
- devices
  - [\\_virDomainDef](#), 20
- direct
  - [\\_virDomainActualNetDef](#), 10
  - [\\_virDomainNetDef](#), 45
- disk
  - [\\_virDomainDeviceDef](#), 24
  - [\\_virDomainDiskError](#), 31
- disks
  - [\\_virDomainDef](#), 20
- display
  - [\\_virDomainGraphicsDef](#), 35
- dns
  - [\\_virNetworkDef](#), 68
- dnsmasqCaps
  - [network\\_driver](#), 90
- dnsmasqPid
  - [\\_virNetworkObj](#), 76
- do\_open
  - [libvirt.c](#), 343
- doi
  - [\\_virSecurityModel](#), 83
- domain
  - [\\_virNetworkDNSSrvRecordsDef](#), 71
  - [\\_virNetworkDef](#), 68
- domain\_conf.c
  - [VIR\\_DOMAIN\\_XML\\_INTERNAL\\_ACTUAL\\_NET](#), 236
  - [VIR\\_DOMAIN\\_XML\\_INTERNAL\\_ALLOW\\_BOOT](#), 236
  - [VIR\\_DOMAIN\\_XML\\_INTERNAL\\_ALLOW\\_ROM](#), 236
  - [VIR\\_DOMAIN\\_XML\\_INTERNAL\\_PCI\\_ORIG\\_STATES](#), 236
  - [VIR\\_DOMAIN\\_XML\\_INTERNAL\\_STATUS](#), 236
- domain\_conf.h
  - [VIR\\_DOMAIN\\_APIC\\_EOI\\_DEFAULT](#), 269
  - [VIR\\_DOMAIN\\_APIC\\_EOI\\_LAST](#), 269
  - [VIR\\_DOMAIN\\_APIC\\_EOI\\_OFF](#), 269
  - [VIR\\_DOMAIN\\_APIC\\_EOI\\_ON](#), 269
  - [VIR\\_DOMAIN\\_BIOS\\_USESERIAL\\_DEFAULT](#), 270
  - [VIR\\_DOMAIN\\_BIOS\\_USESERIAL\\_NO](#), 270
  - [VIR\\_DOMAIN\\_BIOS\\_USESERIAL\\_YES](#), 270
  - [VIR\\_DOMAIN\\_BOOT\\_CDROM](#), 270
  - [VIR\\_DOMAIN\\_BOOT\\_DISK](#), 270

- VIR\_DOMAIN\_BOOT\_FLOPPY, [270](#)
- VIR\_DOMAIN\_BOOT\_LAST, [270](#)
- VIR\_DOMAIN\_BOOT\_MENU\_DEFAULT, [270](#)
- VIR\_DOMAIN\_BOOT\_MENU\_DISABLED, [270](#)
- VIR\_DOMAIN\_BOOT\_MENU\_ENABLED, [270](#)
- VIR\_DOMAIN\_BOOT\_MENU\_LAST, [270](#)
- VIR\_DOMAIN\_BOOT\_NET, [270](#)
- VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE-  
\_GUESTFWD, [270](#)
- VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE-  
\_LAST, [270](#)
- VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE-  
\_NONE, [270](#)
- VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE-  
\_VIRTIO, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE-  
\_LAST, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE-  
\_LXC, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE-  
\_OPENVZ, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE-  
\_SERIAL, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE-  
\_UML, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE-  
\_VIRTIO, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE-  
\_XEN, [270](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_CHANNEL, [271](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_CONSOLE, [271](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_LAST, [271](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_PARALLEL, [271](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_SERIAL, [271](#)
- VIR\_DOMAIN\_CHR\_SPICEVMC\_LAST, [271](#)
- VIR\_DOMAIN\_CHR\_SPICEVMC\_SMARTCARD, [271](#)
- VIR\_DOMAIN\_CHR\_SPICEVMC\_USBREDIR, [271](#)
- VIR\_DOMAIN\_CHR\_SPICEVMC\_VDAGENT, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_LAST, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_RAW, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TELNET, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TELNET-  
S, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TLS, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_DEV, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_FILE, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_LAST, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_NULL, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_PIPE, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_PTY, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_SPICEVMC, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_STDIO, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_TCP, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_UDP, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_UNIX, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_VC, [271](#)
- VIR\_DOMAIN\_CLOCK\_BASIS\_LAST, [272](#)
- VIR\_DOMAIN\_CLOCK\_BASIS\_LOCALTIME, [272](#)
- VIR\_DOMAIN\_CLOCK\_BASIS\_UTC, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_LAST, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_LOCALTIME, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_TIMEZONE, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_UTC, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_VARIABLE, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MASTER\_LAST, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MASTER\_NONE, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MASTER\_USB, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_A-  
UTO, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_B-  
USLOGIC, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_IB-  
MVSCSI, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_L-  
AST, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_L-  
SILOGIC, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_L-  
SISAS1068, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_VI-  
RTIO\_SCSI, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_V-  
MPVSCSI, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_E-  
HCI, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_IC-  
H9\_EHCI1, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_IC-  
H9\_UHCI1, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_IC-  
H9\_UHCI2, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_IC-  
H9\_UHCI3, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_LA-  
ST, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_N-  
EC\_XHCI, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_N-  
ONE, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_P-  
CI\_OHCI, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_PII-

- X3\_UHCI, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_PII-X4\_UHCI, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_V-T82C686B\_UHCI, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_CCID, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_FDC, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_IDE, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_LAST, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_SATA, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_SCSI, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_USB, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_VIRTIO\_SERIAL, [273](#)
- VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_AUTO, [273](#)
- VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_LAST, [273](#)
- VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_STATIC, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_CCID, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_DRIVE, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_LAST, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_NON-E, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_PCI, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_SPARVIO, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_USB, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_VIRTIO\_S390, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_VIRTIO\_SERIAL, [273](#)
- VIR\_DOMAIN\_DEVICE\_CHR, [274](#)
- VIR\_DOMAIN\_DEVICE\_CONTROLLER, [274](#)
- VIR\_DOMAIN\_DEVICE\_DISK, [273](#)
- VIR\_DOMAIN\_DEVICE\_FS, [273](#)
- VIR\_DOMAIN\_DEVICE\_GRAPHICS, [274](#)
- VIR\_DOMAIN\_DEVICE\_HOSTDEV, [274](#)
- VIR\_DOMAIN\_DEVICE\_HUB, [274](#)
- VIR\_DOMAIN\_DEVICE\_INPUT, [274](#)
- VIR\_DOMAIN\_DEVICE\_LAST, [274](#)
- VIR\_DOMAIN\_DEVICE\_LEASE, [273](#)
- VIR\_DOMAIN\_DEVICE\_MEMBALLOON, [274](#)
- VIR\_DOMAIN\_DEVICE\_NET, [274](#)
- VIR\_DOMAIN\_DEVICE\_NONE, [273](#)
- VIR\_DOMAIN\_DEVICE\_REDIRDEV, [274](#)
- VIR\_DOMAIN\_DEVICE\_SMARTCARD, [274](#)
- VIR\_DOMAIN\_DEVICE\_SOUND, [274](#)
- VIR\_DOMAIN\_DEVICE\_VIDEO, [274](#)
- VIR\_DOMAIN\_DEVICE\_WATCHDOG, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_FDC, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_IDE, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_LAST, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_SATA, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_SCSI, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_UML, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_USB, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_VIRTIO, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_XEN, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_DEFAULT, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_DIRECTSYNC, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_DISABLE, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_LAST, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_UNSAFE, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_WRITEBACK, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_WRITETHRU, [274](#)
- VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_DEFAULT, [274](#)
- VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_LAST, [274](#)
- VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_OFF, [274](#)
- VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_ON, [274](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_CDROM, [275](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_DISK, [275](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_FLOPPY, [275](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_LAST, [275](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_LUN, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_DEFAULT, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_ENOSPACE, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_IGNORE, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_LAST, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_REPORT, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_STOP, [275](#)
- VIR\_DOMAIN\_DISK\_IO\_DEFAULT, [275](#)
- VIR\_DOMAIN\_DISK\_IO\_LAST, [275](#)
- VIR\_DOMAIN\_DISK\_IO\_NATIVE, [275](#)
- VIR\_DOMAIN\_DISK\_IO\_THREADS, [275](#)
- VIR\_DOMAIN\_DISK\_PROTOCOL\_LAST, [275](#)
- VIR\_DOMAIN\_DISK\_PROTOCOL\_NBD, [275](#)
- VIR\_DOMAIN\_DISK\_PROTOCOL\_RBD, [275](#)
- VIR\_DOMAIN\_DISK\_PROTOCOL\_SHEEPDOG, [275](#)
- VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_LAST, [276](#)
- VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_NONE, [276](#)
- VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_USAGE, [276](#)
- VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_UUID, [276](#)
- VIR\_DOMAIN\_DISK\_TRANS\_AUTO, [275](#)
- VIR\_DOMAIN\_DISK\_TRANS\_DEFAULT, [275](#)
- VIR\_DOMAIN\_DISK\_TRANS\_LAST, [275](#)
- VIR\_DOMAIN\_DISK\_TRANS\_LBA, [275](#)
- VIR\_DOMAIN\_DISK\_TRANS\_NONE, [275](#)
- VIR\_DOMAIN\_DISK\_TRAY\_CLOSED, [276](#)



- VIR\_DOMAIN\_DISK\_TRAY\_LAST, [276](#)
- VIR\_DOMAIN\_DISK\_TRAY\_OPEN, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_BLOCK, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_DIR, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_FILE, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_LAST, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_NETWORK, [276](#)
- VIR\_DOMAIN\_FEATURE\_ACPI, [276](#)
- VIR\_DOMAIN\_FEATURE\_APIC, [276](#)
- VIR\_DOMAIN\_FEATURE\_HAP, [276](#)
- VIR\_DOMAIN\_FEATURE\_LAST, [276](#)
- VIR\_DOMAIN\_FEATURE\_PAE, [276](#)
- VIR\_DOMAIN\_FEATURE\_PRIVNET, [276](#)
- VIR\_DOMAIN\_FEATURE\_VIRIDIAN, [276](#)
- VIR\_DOMAIN\_FS\_ACCESSMODE\_LAST, [276](#)
- VIR\_DOMAIN\_FS\_ACCESSMODE\_MAPPED, [276](#)
- VIR\_DOMAIN\_FS\_ACCESSMODE\_PASSTHROUGH, [276](#)
- VIR\_DOMAIN\_FS\_ACCESSMODE\_SQUASH, [276](#)
- VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_DEFAULT, [277](#)
- VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_HANDLE, [277](#)
- VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_LAST, [277](#)
- VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_PATH, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_BIND, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_BLOCK, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_FILE, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_LAST, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_MOUNT, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_RAM, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_TEMPLATE, [277](#)
- VIR\_DOMAIN\_FS\_WRPOLICY\_DEFAULT, [277](#)
- VIR\_DOMAIN\_FS\_WRPOLICY\_IMMEDIATE, [277](#)
- VIR\_DOMAIN\_FS\_WRPOLICY\_LAST, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_DEFAULT, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_DISCONNECT, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_FAIL, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_KEEP, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_LAST, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_ADDRESS, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_LAST, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_NETWORK, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_NONE, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_CURSOR, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_DISPLAY, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_INPUT, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_LAST, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MAIN, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_ANY, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_INSECURE, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_LAST, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_SECURE, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_PLAYBACK, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_RECORD, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_SMARTCARD, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_USBREDIR, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_DEFAULT, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_LAST, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_NO, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_YES, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_AUTO\_GLZ, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_AUTO\_LZ, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_DEFAULT, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_GLZ, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_LAST, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_LZ, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_OFF, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_QUIC, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_ALWAYS, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_AUTO, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_DEFAULT, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_LAST, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_NEVER, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_CLIENT, [279](#)



- VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_DEFAULT, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_LAST, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_SERVER, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_DEFAULT, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_LAST, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_OFF, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_ON, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_ALL, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_DEFAULT, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_FILTER, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_LAST, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_OFF, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_ALWAYS, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_AUTO, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_DEFAULT, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_LAST, 279
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_NEVER, 279
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_DESKTOP, 280
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_LAST, 280
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_RDP, 280
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_SDL, 280
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_SPICE, 280
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_VNC, 280
- VIR\_DOMAIN\_HOSTDEV\_MODE\_CAPABILITIES, 280
- VIR\_DOMAIN\_HOSTDEV\_MODE\_LAST, 280
- VIR\_DOMAIN\_HOSTDEV\_MODE\_SUBSYS, 280
- VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_LAST, 280
- VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_PCI, 280
- VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_USB, 280
- VIR\_DOMAIN\_HUB\_TYPE\_LAST, 280
- VIR\_DOMAIN\_HUB\_TYPE\_USB, 280
- VIR\_DOMAIN\_INPUT\_BUS\_LAST, 280
- VIR\_DOMAIN\_INPUT\_BUS\_PS2, 280
- VIR\_DOMAIN\_INPUT\_BUS\_USB, 280
- VIR\_DOMAIN\_INPUT\_BUS\_XEN, 280
- VIR\_DOMAIN\_INPUT\_TYPE\_LAST, 280
- VIR\_DOMAIN\_INPUT\_TYPE\_MOUSE, 280
- VIR\_DOMAIN\_INPUT\_TYPE\_TABLET, 280
- VIR\_DOMAIN\_IO\_EVENT\_FD\_DEFAULT, 281
- VIR\_DOMAIN\_IO\_EVENT\_FD\_LAST, 281
- VIR\_DOMAIN\_IO\_EVENT\_FD\_OFF, 281
- VIR\_DOMAIN\_IO\_EVENT\_FD\_ON, 281
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_COREDUMP\_DESTROY, 281
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_COREDUMP\_RESTART, 281
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_DESTROY, 281
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_LAST, 281
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_PRESERVE, 281
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_RESTART, 281
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_RESTART\_RENAME, 281
- VIR\_DOMAIN\_LIFECYCLE\_DESTROY, 281
- VIR\_DOMAIN\_LIFECYCLE\_LAST, 281
- VIR\_DOMAIN\_LIFECYCLE\_PRESERVE, 281
- VIR\_DOMAIN\_LIFECYCLE\_RESTART, 281
- VIR\_DOMAIN\_LIFECYCLE\_RESTART\_RENAME, 281
- VIR\_DOMAIN\_MEM\_DUMP\_DEFAULT, 281
- VIR\_DOMAIN\_MEM\_DUMP\_LAST, 281
- VIR\_DOMAIN\_MEM\_DUMP\_OFF, 281
- VIR\_DOMAIN\_MEM\_DUMP\_ON, 281
- VIR\_DOMAIN\_MEMBALLOON\_MODEL\_LAST, 269
- VIR\_DOMAIN\_MEMBALLOON\_MODEL\_NONE, 269
- VIR\_DOMAIN\_MEMBALLOON\_MODEL\_VIRTIO, 269
- VIR\_DOMAIN\_MEMBALLOON\_MODEL\_XEN, 269
- VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_DEFAULT, 281
- VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_LAST, 281
- VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_QEMU, 281
- VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_VHOST, 281
- VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_DEFAULT, 282
- VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_DOWN, 282
- VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_LAST, 282
- VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_UP, 282
- VIR\_DOMAIN\_NET\_TYPE\_BRIDGE, 282
- VIR\_DOMAIN\_NET\_TYPE\_CLIENT, 282
- VIR\_DOMAIN\_NET\_TYPE\_DIRECT, 282
- VIR\_DOMAIN\_NET\_TYPE\_ETHERNET, 282
- VIR\_DOMAIN\_NET\_TYPE\_HOSTDEV, 282

- VIR\_DOMAIN\_NET\_TYPE\_INTERNAL, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_LAST, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_MCAST, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_NETWORK, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_SERVER, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_USER, [282](#)
- VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_DEFAULT, [282](#)
- VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_IOTHEAD, [282](#)
- VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_LAST, [282](#)
- VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_TIMER, [282](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_AUTO, [282](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_DEFAULT, [282](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_LAST, [282](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_STATIC, [282](#)
- VIR\_DOMAIN\_PCI\_ROMBAR\_DEFAULT, [282](#)
- VIR\_DOMAIN\_PCI\_ROMBAR\_LAST, [282](#)
- VIR\_DOMAIN\_PCI\_ROMBAR\_OFF, [282](#)
- VIR\_DOMAIN\_PCI\_ROMBAR\_ON, [282](#)
- VIR\_DOMAIN\_PM\_STATE\_DEFAULT, [283](#)
- VIR\_DOMAIN\_PM\_STATE\_DISABLED, [283](#)
- VIR\_DOMAIN\_PM\_STATE\_ENABLED, [283](#)
- VIR\_DOMAIN\_PM\_STATE\_LAST, [283](#)
- VIR\_DOMAIN\_REDIRDEV\_BUS\_LAST, [283](#)
- VIR\_DOMAIN\_REDIRDEV\_BUS\_USB, [283](#)
- VIR\_DOMAIN\_SECLABEL\_DEFAULT, [283](#)
- VIR\_DOMAIN\_SECLABEL\_DYNAMIC, [283](#)
- VIR\_DOMAIN\_SECLABEL\_LAST, [283](#)
- VIR\_DOMAIN\_SECLABEL\_NONE, [283](#)
- VIR\_DOMAIN\_SECLABEL\_STATIC, [283](#)
- VIR\_DOMAIN\_SMARTCARD\_TYPE\_HOST, [283](#)
- VIR\_DOMAIN\_SMARTCARD\_TYPE\_HOST\_CERTIFICATES, [283](#)
- VIR\_DOMAIN\_SMARTCARD\_TYPE\_LAST, [283](#)
- VIR\_DOMAIN\_SMARTCARD\_TYPE\_PASSTHROUGH, [283](#)
- VIR\_DOMAIN\_SMBIOS\_EMULATE, [283](#)
- VIR\_DOMAIN\_SMBIOS\_HOST, [283](#)
- VIR\_DOMAIN\_SMBIOS\_LAST, [283](#)
- VIR\_DOMAIN\_SMBIOS\_NONE, [283](#)
- VIR\_DOMAIN\_SMBIOS\_SYSINFO, [283](#)
- VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_DUPLEX, [283](#)
- VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_LAST, [283](#)
- VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_MICRO, [283](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_AC97, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_ES1370, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_ICH6, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_LAST, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_PCSPK, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_SB16, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_DEFAULT, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_LAST, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_MANDATORY, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_OPTIONAL, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_REQUISITE, [284](#)
- VIR\_DOMAIN\_TAINT\_CUSTOM\_ARGV, [284](#)
- VIR\_DOMAIN\_TAINT\_CUSTOM\_MONITOR, [284](#)
- VIR\_DOMAIN\_TAINT\_DISK\_PROBING, [284](#)
- VIR\_DOMAIN\_TAINT\_EXTERNAL\_LAUNCH, [284](#)
- VIR\_DOMAIN\_TAINT\_HIGH\_PRIVILEGES, [284](#)
- VIR\_DOMAIN\_TAINT\_HOST\_CPU, [284](#)
- VIR\_DOMAIN\_TAINT\_LAST, [284](#)
- VIR\_DOMAIN\_TAINT\_SHELL\_SCRIPTS, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_AUTO, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_EMULATE, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_LAST, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_NATIVE, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_PARAVIRT, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_SMPSAFE, [284](#)
- VIR\_DOMAIN\_TIMER\_NAME\_HPET, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_KVMCLOCK, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_LAST, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_PIT, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_PLATFORM, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_RTC, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_TSC, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_CATCHUP, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_DELAY, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_DISCARD, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_LAST, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_MERGE, [285](#)
- VIR\_DOMAIN\_TIMER\_TRACK\_BOOT, [285](#)
- VIR\_DOMAIN\_TIMER\_TRACK\_GUEST, [285](#)
- VIR\_DOMAIN\_TIMER\_TRACK\_LAST, [285](#)
- VIR\_DOMAIN\_TIMER\_TRACK\_WALL, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_CIRRUS, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_LAST, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_QXL, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_VBOX, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_VGA, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_VMVGA, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_XEN, [285](#)
- VIR\_DOMAIN\_VIRT\_HYPERV, [286](#)
- VIR\_DOMAIN\_VIRT\_KQEMU, [286](#)
- VIR\_DOMAIN\_VIRT\_KVM, [286](#)
- VIR\_DOMAIN\_VIRT\_LAST, [286](#)
- VIR\_DOMAIN\_VIRT\_LXC, [286](#)
- VIR\_DOMAIN\_VIRT\_OPENVZ, [286](#)
- VIR\_DOMAIN\_VIRT\_PARALLELS, [286](#)

- VIR\_DOMAIN\_VIRT\_PHY, [286](#)
- VIR\_DOMAIN\_VIRT\_QEMU, [286](#)
- VIR\_DOMAIN\_VIRT\_TEST, [286](#)
- VIR\_DOMAIN\_VIRT\_UML, [286](#)
- VIR\_DOMAIN\_VIRT\_VBOX, [286](#)
- VIR\_DOMAIN\_VIRT\_VMWARE, [286](#)
- VIR\_DOMAIN\_VIRT\_XEN, [286](#)
- VIR\_DOMAIN\_VIRTIO\_EVENT\_IDX\_DEFAULT, [285](#)
- VIR\_DOMAIN\_VIRTIO\_EVENT\_IDX\_LAST, [285](#)
- VIR\_DOMAIN\_VIRTIO\_EVENT\_IDX\_OFF, [285](#)
- VIR\_DOMAIN\_VIRTIO\_EVENT\_IDX\_ON, [285](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_DUMP, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_LAST, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_NONE, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_PAUSE, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_POWEROFF, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_RESET, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_SHUTDOWN, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_MODEL\_I6300ESB, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_MODEL\_IB700, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_MODEL\_LAST, [286](#)
- domain\_conf.c
  - ATTRIBUTE\_NONNULL, [236](#)
  - DUMPXML\_FLAGS, [235](#)
  - MATCH, [235](#)
  - NET\_MODEL\_CHARS, [236](#)
  - VIR\_ENUM\_IMPL, [236](#)
  - VIR\_FROM\_THIS, [236](#)
  - verify, [236](#)
  - virBlkioDeviceWeightArrayClear, [236](#)
  - virDiskNameToBusDeviceIndex, [237](#)
  - virDomainActualNetDefFormat, [237](#)
  - virDomainActualNetDefFree, [237](#)
  - virDomainActualNetDefParseXML, [237](#)
  - virDomainAssignDef, [237](#)
  - virDomainBlkioDeviceWeightParseXML, [237](#)
  - virDomainChannelDefCheckABIStability, [237](#)
  - virDomainChrDefForeach, [237](#)
  - virDomainChrDefFormat, [238](#)
  - virDomainChrDefFree, [238](#)
  - virDomainChrDefGetSecurityLabelDef, [238](#)
  - virDomainChrDefNew, [238](#)
  - virDomainChrDefParseTargetXML, [238](#)
  - virDomainChrDefParseXML, [238](#)
  - virDomainChrDefaultTargetType, [237](#)
  - virDomainChrSourceDefCopy, [238](#)
  - virDomainChrSourceDefFormat, [238](#)
  - virDomainChrSourceDefFree, [238](#)
  - virDomainChrSourceDefIsEqual, [238](#)
  - virDomainChrSourceDefParseXML, [238](#)
  - virDomainChrTargetTypeFromString, [238](#)
  - virDomainChrTargetTypeToString, [238](#)
  - virDomainClockDefClear, [238](#)
  - virDomainConfigFile, [238](#)
  - virDomainConsoleDefCheckABIStability, [238](#)
  - virDomainControllerDefCheckABIStability, [238](#)
  - virDomainControllerDefFormat, [238](#)
  - virDomainControllerDefFree, [238](#)
  - virDomainControllerDefParseXML, [238](#)
  - virDomainControllerFind, [238](#)
  - virDomainControllerInsert, [239](#)
  - virDomainControllerInsertPreAllocated, [239](#)
  - virDomainControllerModelTypeFromString, [239](#)
  - virDomainControllerModelTypeToString, [239](#)
  - virDomainControllerRemove, [239](#)
  - virDomainDefAddDiskControllersForType, [239](#)
  - virDomainDefAddImplicitControllers, [239](#)
  - virDomainDefAddSecurityLabelDef, [239](#)
  - virDomainDefCheckABIStability, [239](#)
  - virDomainDefClearDeviceAliases, [239](#)
  - virDomainDefClearPCIAddresses, [239](#)
  - virDomainDefCompatibleDevice, [239](#)
  - virDomainDefDefaultEmulator, [239](#)
  - virDomainDefFormat, [239](#)
  - virDomainDefFormatInternal, [239](#)
  - virDomainDefFree, [239](#)
  - virDomainDefGetSecurityLabelDef, [239](#)
  - virDomainDefHasUSB, [239](#)
  - virDomainDefMaybeAddController, [239](#)
  - virDomainDefMaybeAddSmartcardController, [239](#)
  - virDomainDefMaybeAddVirtioSerialController, [239](#)
  - virDomainDefParse, [239](#)
  - virDomainDefParseBootXML, [239](#)
  - virDomainDefParseFile, [239](#)
  - virDomainDefParseNode, [239](#)
  - virDomainDefParseString, [240](#)
  - virDomainDefParseXML, [240](#)
  - virDomainDeleteConfig, [240](#)
  - virDomainDeviceAddressIsValid, [240](#)
  - virDomainDeviceBootParseXML, [240](#)
  - virDomainDeviceCcidAddressParseXML, [240](#)
  - virDomainDeviceDefCopy, [240](#)
  - virDomainDeviceDefFree, [240](#)
  - virDomainDeviceDefParse, [240](#)
  - virDomainDeviceDriveAddressParseXML, [240](#)
  - virDomainDeviceInfoCheckABIStability, [240](#)
  - virDomainDeviceInfoClear, [240](#)
  - virDomainDeviceInfoClearAlias, [240](#)
  - virDomainDeviceInfoClearPCIAddress, [240](#)
  - virDomainDeviceInfoFree, [240](#)
  - virDomainDeviceInfoSet, [240](#)
  - virDomainDeviceInfoIterate, [240](#)
  - virDomainDeviceInfoParseXML, [240](#)
  - virDomainDeviceIsUSB, [240](#)
  - virDomainDeviceSpaprVioAddressParseXML, [240](#)
  - virDomainDeviceUSBAddressParseXML, [240](#)
  - virDomainDeviceUSBMasterParseXML, [241](#)

- virDomainDeviceVirtioSerialAddressParseXML, 241
- virDomainDiskBlockIoDefFormat, 241
- virDomainDiskDefAssignAddress, 241
- virDomainDiskDefCheckABIStability, 241
- virDomainDiskDefForeachPath, 241
- virDomainDiskDefFormat, 241
- virDomainDiskDefFree, 241
- virDomainDiskDefGetSecurityLabelDef, 241
- virDomainDiskDefParseXML, 241
- virDomainDiskFindControllerModel, 241
- virDomainDiskGeometryDefFormat, 241
- virDomainDiskHostDefFree, 241
- virDomainDiskIndexByName, 241
- virDomainDiskInsert, 241
- virDomainDiskInsertPreAllocated, 241
- virDomainDiskPathByName, 241
- virDomainDiskRemove, 241
- virDomainDiskRemoveByName, 241
- virDomainEmulatorPinAdd, 241
- virDomainEmulatorPinDel, 241
- virDomainFSDefFormat, 242
- virDomainFSDefFree, 242
- virDomainFSDefParseXML, 242
- virDomainFSIndexByName, 242
- virDomainFindByID, 241
- virDomainFindByName, 241
- virDomainFindByUUID, 241
- virDomainFsDefCheckABIStability, 242
- virDomainGetRootFilesystem, 242
- virDomainGraphicsAuthDefClear, 242
- virDomainGraphicsAuthDefFormatAttr, 242
- virDomainGraphicsAuthDefParseXML, 242
- virDomainGraphicsDefFormat, 242
- virDomainGraphicsDefFree, 242
- virDomainGraphicsDefParseXML, 242
- virDomainGraphicsGetListen, 242
- virDomainGraphicsListenDefClear, 242
- virDomainGraphicsListenDefFormat, 242
- virDomainGraphicsListenDefParseXML, 242
- virDomainGraphicsListenGetAddress, 242
- virDomainGraphicsListenGetNetwork, 242
- virDomainGraphicsListenGetType, 242
- virDomainGraphicsListenSetAddress, 242
- virDomainGraphicsListenSetNetwork, 242
- virDomainGraphicsListenSetType, 242
- virDomainHostdevDefAlloc, 242
- virDomainHostdevDefCheckABIStability, 243
- virDomainHostdevDefClear, 243
- virDomainHostdevDefFormat, 243
- virDomainHostdevDefFree, 243
- virDomainHostdevDefParseXML, 243
- virDomainHostdevFind, 243
- virDomainHostdevInsert, 243
- virDomainHostdevPartsParse, 243
- virDomainHostdevRemove, 243
- virDomainHostdevSourceFormat, 243
- virDomainHostdevSubsysPciDefParseXML, 243
- virDomainHostdevSubsysPciOrigStatesDefParseXML, 243
- virDomainHostdevSubsysUsbDefParseXML, 243
- virDomainHubDefCheckABIStability, 243
- virDomainHubDefFormat, 243
- virDomainHubDefFree, 243
- virDomainHubDefParseXML, 243
- virDomainInputDefCheckABIStability, 243
- virDomainInputDefFormat, 243
- virDomainInputDefFree, 243
- virDomainInputDefParseXML, 243
- virDomainLeaseDefFormat, 244
- virDomainLeaseDefFree, 244
- virDomainLeaseDefParseXML, 244
- virDomainLeaseIndex, 244
- virDomainLeaseInsert, 244
- virDomainLeaseInsertPreAlloc, 244
- virDomainLeaseInsertPreAllocated, 244
- virDomainLeaseRemove, 244
- virDomainLeaseRemoveAt, 244
- virDomainLifecycleDefFormat, 244
- virDomainLifecycleParseXML, 244
- virDomainList, 244
- virDomainListPopulate, 244
- virDomainLiveConfigHelperMethod, 244
- virDomainLoadAllConfigs, 244
- virDomainLoadConfig, 244
- virDomainLoadStatus, 244
- virDomainMemballoonDefCheckABIStability, 244
- virDomainMemballoonDefFormat, 244
- virDomainMemballoonDefFree, 244
- virDomainMemballoonDefParseXML, 244
- virDomainNetDefCheckABIStability, 245
- virDomainNetDefFormat, 245
- virDomainNetDefFree, 245
- virDomainNetDefParseXML, 245
- virDomainNetFind, 245
- virDomainNetGetActualBandwidth, 245
- virDomainNetGetActualBridgeName, 246
- virDomainNetGetActualBridgeType, 246
- virDomainNetGetActualDirectDev, 246
- virDomainNetGetActualDirectMode, 246
- virDomainNetGetActualHostdev, 246
- virDomainNetGetActualType, 246
- virDomainNetGetActualVirtPortProfile, 246
- virDomainNetGetActualVlan, 246
- virDomainNetIndexByMac, 246
- virDomainNetInsert, 246
- virDomainNetRemove, 246
- virDomainNetRemoveByMac, 246
- virDomainObjAssignDef, 246
- virDomainObjCopyPersistentDef, 246
- virDomainObjDispose, 246
- virDomainObjFormat, 246
- virDomainObjGetPersistentDef, 246
- virDomainObjGetState, 246
- virDomainObjIsDuplicate, 247
- virDomainObjListCopyActiveIDs, 247

- virDomainObjListCopyInactiveNames, 247
- virDomainObjListCountActive, 247
- virDomainObjListCountInactive, 247
- virDomainObjListDataFree, 247
- virDomainObjListDeinit, 247
- virDomainObjListGetActiveIDs, 247
- virDomainObjListGetInactiveNames, 247
- virDomainObjListInit, 247
- virDomainObjListNumOfDomains, 247
- virDomainObjListSearchID, 247
- virDomainObjListSearchName, 247
- virDomainObjLock, 247
- virDomainObjNew, 247
- virDomainObjParseFile, 247
- virDomainObjParseNode, 247
- virDomainObjParseXML, 247
- virDomainObjSetDefTransient, 247
- virDomainObjSetState, 247
- virDomainObjTaint, 247
- virDomainObjUnlock, 247
- virDomainPMStateParseXML, 248
- virDomainParallelDefCheckABIStability, 247
- virDomainParseLegacyDeviceAddress, 248
- virDomainParseMemory, 248
- virDomainParseScaledValue, 248
- virDomainRedirFilterDefCheckABIStability, 248
- virDomainRedirFilterDefFormat, 248
- virDomainRedirFilterDefFree, 248
- virDomainRedirFilterDefParseXML, 248
- virDomainRedirFilterUsbDevDefParseXML, 248
- virDomainRedirFilterUsbVersionHelper, 248
- virDomainRedirdevDefFormat, 248
- virDomainRedirdevDefFree, 248
- virDomainRedirdevDefParseXML, 248
- virDomainRemoveInactive, 248
- virDomainSaveConfig, 248
- virDomainSaveStatus, 248
- virDomainSaveXML, 248
- virDomainSerialDefCheckABIStability, 248
- virDomainSmartcardDefCheckABIStability, 248
- virDomainSmartcardDefForeach, 248
- virDomainSmartcardDefFormat, 248
- virDomainSmartcardDefFree, 248
- virDomainSmartcardDefParseXML, 249
- virDomainSoundCodecDefFormat, 249
- virDomainSoundCodecDefFree, 249
- virDomainSoundCodecDefParseXML, 249
- virDomainSoundDefCheckABIStability, 249
- virDomainSoundDefFormat, 249
- virDomainSoundDefFree, 249
- virDomainSoundDefParseXML, 249
- virDomainStateReasonFromString, 249
- virDomainStateReasonToString, 249
- virDomainSysinfoDefFormat, 249
- virDomainTimerDefCheckABIStability, 249
- virDomainTimerDefFormat, 249
- virDomainTimerDefParseXML, 249
- virDomainVcpuPinAdd, 249
- virDomainVcpuPinDefArrayFree, 249
- virDomainVcpuPinDefCopy, 249
- virDomainVcpuPinDefFree, 249
- virDomainVcpuPinDefParseXML, 249
- virDomainVcpuPinDel, 249
- virDomainVcpuPinFindByVcpu, 249
- virDomainVcpuPinIsDuplicate, 249
- virDomainVideoAccelDefFormat, 249
- virDomainVideoAccelDefParseXML, 250
- virDomainVideoDefCheckABIStability, 250
- virDomainVideoDefFormat, 250
- virDomainVideoDefFree, 250
- virDomainVideoDefParseXML, 250
- virDomainVideoDefaultRAM, 250
- virDomainVideoDefaultType, 250
- virDomainWatchdogDefCheckABIStability, 250
- virDomainWatchdogDefFormat, 250
- virDomainWatchdogDefFree, 250
- virDomainWatchdogDefParseXML, 250
- virDomainXMLInternalFlags, 236
- virSecurityDeviceLabelDefFormat, 250
- virSecurityDeviceLabelDefFree, 250
- virSecurityDeviceLabelDefParseXML, 250
- virSecurityLabelDefFormat, 250
- virSecurityLabelDefFree, 250
- virSecurityLabelDefParseXML, 250
- virSecurityLabelDefsParseXML, 250
- virSysinfoParseXML, 250
- domain\_conf.h
  - virBlkioDeviceWeight, 265
  - virBlkioDeviceWeightArrayClear, 286
  - virBlkioDeviceWeightPtr, 265
  - virDiskNameToBusDeviceIndex, 286
  - virDomainActualNetDef, 265
  - virDomainActualNetDefFree, 286
  - virDomainActualNetDefPtr, 265
  - virDomainApicEoi, 269
  - virDomainAssignDef, 287
  - virDomainBIOSDef, 265
  - virDomainBIOSDefPtr, 265
  - virDomainBIOSUseserial, 269
  - virDomainBlockIoTunelInfo, 265
  - virDomainBlockIoTunelInfoPtr, 265
  - virDomainBootMenu, 270
  - virDomainBootOrder, 270
  - virDomainChrChannelTargetType, 270
  - virDomainChrConsoleTargetType, 270
  - virDomainChrDef, 265
  - virDomainChrDefForeach, 287
  - virDomainChrDefFree, 287
  - virDomainChrDefGetSecurityLabelDef, 287
  - virDomainChrDefIterator, 265
  - virDomainChrDefNew, 287
  - virDomainChrDefPtr, 265
  - virDomainChrDeviceType, 270
  - virDomainChrSourceDef, 265
  - virDomainChrSourceDefCopy, 287
  - virDomainChrSourceDefFree, 287



- virDomainChrSourceDefPtr, 265
- virDomainChrSpicevmcName, 271
- virDomainChrTcpProtocol, 271
- virDomainChrType, 271
- virDomainClockBasis, 271
- virDomainClockDef, 265
- virDomainClockDefPtr, 266
- virDomainClockOffsetType, 272
- virDomainConfigFile, 287
- virDomainControllerDef, 266
- virDomainControllerDefFree, 287
- virDomainControllerDefPtr, 266
- virDomainControllerFind, 287
- virDomainControllerInsert, 287
- virDomainControllerInsertPreAlloced, 287
- virDomainControllerMaster, 272
- virDomainControllerModelSCSI, 272
- virDomainControllerModelUSB, 272
- virDomainControllerRemove, 287
- virDomainControllerType, 273
- virDomainCpuPlacementMode, 273
- virDomainDef, 266
- virDomainDefAddImplicitControllers, 287
- virDomainDefAddSecurityLabelDef, 287
- virDomainDefCheckABIStability, 287
- virDomainDefClearDeviceAliases, 287
- virDomainDefClearPCIAddresses, 287
- virDomainDefCompatibleDevice, 287
- virDomainDefFormat, 287
- virDomainDefFormatInternal, 287
- virDomainDefFree, 287
- virDomainDefGetSecurityLabelDef, 287
- virDomainDefParseFile, 288
- virDomainDefParseNode, 288
- virDomainDefParseString, 288
- virDomainDefPtr, 266
- virDomainDeleteConfig, 288
- virDomainDeviceAddressIsValid, 288
- virDomainDeviceAddressType, 273
- virDomainDeviceCcidAddress, 266
- virDomainDeviceCcidAddressPtr, 266
- virDomainDeviceDef, 266
- virDomainDeviceDefCopy, 288
- virDomainDeviceDefFree, 288
- virDomainDeviceDefParse, 288
- virDomainDeviceDefPtr, 266
- virDomainDeviceDriveAddress, 266
- virDomainDeviceDriveAddressPtr, 266
- virDomainDeviceInfo, 266
- virDomainDeviceInfoCallback, 266
- virDomainDeviceInfoClear, 288
- virDomainDeviceInfoIterate, 288
- virDomainDeviceInfoPtr, 266
- virDomainDeviceSpaprVioAddress, 266
- virDomainDeviceSpaprVioAddressPtr, 266
- virDomainDeviceType, 273
- virDomainDeviceUSBAddress, 266
- virDomainDeviceUSBAddressPtr, 266
- virDomainDeviceUSBMaster, 266
- virDomainDeviceUSBMasterPtr, 266
- virDomainDeviceVirtioSerialAddress, 266
- virDomainDeviceVirtioSerialAddressPtr, 266
- virDomainDiskBus, 274
- virDomainDiskCache, 274
- virDomainDiskCopyOnRead, 274
- virDomainDiskDef, 266
- virDomainDiskDefAssignAddress, 288
- virDomainDiskDefForeachPath, 288
- virDomainDiskDefFree, 288
- virDomainDiskDefGetSecurityLabelDef, 288
- virDomainDiskDefPathIterator, 266
- virDomainDiskDefPtr, 266
- virDomainDiskDevice, 274
- virDomainDiskErrorPolicy, 275
- virDomainDiskFindControllerModel, 288
- virDomainDiskGeometryTrans, 275
- virDomainDiskHostDef, 266
- virDomainDiskHostDefFree, 288
- virDomainDiskHostDefPtr, 266
- virDomainDiskIndexByName, 288
- virDomainDiskInsert, 288
- virDomainDiskInsertPreAlloced, 288
- virDomainDiskIo, 275
- virDomainDiskPathByName, 288
- virDomainDiskProtocol, 275
- virDomainDiskRemove, 288
- virDomainDiskRemoveByName, 288
- virDomainDiskSecretType, 275
- virDomainDiskTray, 276
- virDomainDiskType, 276
- virDomainEmulatorPinAdd, 288
- virDomainEmulatorPinDel, 288
- virDomainFSAccessMode, 276
- virDomainFSDef, 267
- virDomainFSDefFree, 289
- virDomainFSDefPtr, 267
- virDomainFSDriverType, 276
- virDomainFSIndexByName, 289
- virDomainFSType, 277
- virDomainFSWpolicy, 277
- virDomainFeature, 276
- virDomainFindByID, 288
- virDomainFindByName, 289
- virDomainFindByUUID, 289
- virDomainGetRootFilesystem, 289
- virDomainGraphicsAuthConnectedType, 277
- virDomainGraphicsAuthDef, 267
- virDomainGraphicsAuthDefPtr, 267
- virDomainGraphicsDef, 267
- virDomainGraphicsDefFree, 289
- virDomainGraphicsDefPtr, 267
- virDomainGraphicsListenDef, 267
- virDomainGraphicsListenDefPtr, 267
- virDomainGraphicsListenGetAddress, 289
- virDomainGraphicsListenGetNetwork, 289
- virDomainGraphicsListenGetType, 289

- virDomainGraphicsListenSetAddress, 289
- virDomainGraphicsListenSetNetwork, 289
- virDomainGraphicsListenSetType, 289
- virDomainGraphicsListenType, 277
- virDomainGraphicsSpiceChannelMode, 277
- virDomainGraphicsSpiceChannelName, 278
- virDomainGraphicsSpiceClipboardCypypaste, 278
- virDomainGraphicsSpiceImageCompression, 278
- virDomainGraphicsSpiceJpegCompression, 278
- virDomainGraphicsSpiceMouseMode, 279
- virDomainGraphicsSpicePlaybackCompression, 279
- virDomainGraphicsSpiceStreamingMode, 279
- virDomainGraphicsSpiceZlibCompression, 279
- virDomainGraphicsType, 279
- virDomainHostdevDef, 267
- virDomainHostdevDefAlloc, 289
- virDomainHostdevDefClear, 289
- virDomainHostdevDefFree, 289
- virDomainHostdevDefPtr, 267
- virDomainHostdevFind, 289
- virDomainHostdevInsert, 289
- virDomainHostdevMode, 280
- virDomainHostdevOrigStates, 267
- virDomainHostdevOrigStatesPtr, 267
- virDomainHostdevRemove, 289
- virDomainHostdevSubsys, 267
- virDomainHostdevSubsysPtr, 267
- virDomainHostdevSubsysType, 280
- virDomainHubDef, 267
- virDomainHubDefFree, 289
- virDomainHubDefPtr, 267
- virDomainHubType, 280
- virDomainInputBus, 280
- virDomainInputDef, 267
- virDomainInputDefFree, 289
- virDomainInputDefPtr, 267
- virDomainInputType, 280
- virDomainIoEventFd, 280
- virDomainLeaseDef, 267
- virDomainLeaseDefFree, 289
- virDomainLeaseDefPtr, 267
- virDomainLeaseIndex, 289
- virDomainLeaseInsert, 289
- virDomainLeaseInsertPreAlloc, 289
- virDomainLeaseInsertPreAlloced, 289
- virDomainLeaseRemove, 289
- virDomainLeaseRemoveAt, 289
- virDomainLifecycleAction, 281
- virDomainLifecycleCrashAction, 281
- virDomainList, 290
- virDomainLiveConfigHelperMethod, 290
- virDomainLoadAllConfigs, 290
- virDomainLoadConfigNotify, 267
- virDomainMemDump, 281
- virDomainMemballoonDef, 267
- virDomainMemballoonDefFree, 290
- virDomainMemballoonDefPtr, 267
- virDomainNetBackendType, 281
- virDomainNetDef, 267
- virDomainNetDefFree, 290
- virDomainNetDefPtr, 267
- virDomainNetFind, 290
- virDomainNetGetActualBandwidth, 290
- virDomainNetGetActualBridgeName, 290
- virDomainNetGetActualBridgeType, 290
- virDomainNetGetActualDirectDev, 290
- virDomainNetGetActualDirectMode, 290
- virDomainNetGetActualHostdev, 290
- virDomainNetGetActualType, 290
- virDomainNetGetActualVirtPortProfile, 290
- virDomainNetGetActualVlan, 291
- virDomainNetIndexByMac, 291
- virDomainNetInsert, 291
- virDomainNetInterfaceLinkState, 281
- virDomainNetRemove, 291
- virDomainNetRemoveByMac, 291
- virDomainNetType, 282
- virDomainNetVirtioTxModeType, 282
- virDomainNumatuneDef, 267
- virDomainNumatuneDefPtr, 267
- virDomainNumatuneMemPlacementMode, 282
- virDomainOSDef, 268
- virDomainOSDefPtr, 268
- virDomainObj, 267
- virDomainObjAssignDef, 291
- virDomainObjCopyPersistentDef, 291
- virDomainObjGetPersistentDef, 291
- virDomainObjGetState, 291
- virDomainObjsActive, 291
- virDomainObjsDuplicate, 291
- virDomainObjList, 268
- virDomainObjListDeinit, 291
- virDomainObjListGetActiveIDs, 291
- virDomainObjListGetInactiveNames, 291
- virDomainObjListInit, 291
- virDomainObjListNumOfDomains, 291
- virDomainObjListPtr, 268
- virDomainObjLock, 291
- virDomainObjNew, 291
- virDomainObjPtr, 268
- virDomainObjSetDefTransient, 291
- virDomainObjSetState, 291
- virDomainObjTaint, 291
- virDomainObjUnlock, 291
- virDomainPMState, 282
- virDomainPciRombarMode, 282
- virDomainRedirFilterDef, 268
- virDomainRedirFilterDefFree, 291
- virDomainRedirFilterDefPtr, 268
- virDomainRedirFilterUsbDevDef, 268
- virDomainRedirFilterUsbDevDefPtr, 268
- virDomainRedirdevBus, 283
- virDomainRedirdevDef, 268
- virDomainRedirdevDefFree, 291
- virDomainRedirdevDefPtr, 268

- virDomainRemoveInactive, 292
- virDomainSaveConfig, 292
- virDomainSaveStatus, 292
- virDomainSaveXML, 292
- virDomainSeclabelType, 283
- virDomainSmartcardDef, 268
- virDomainSmartcardDefForeach, 292
- virDomainSmartcardDefFree, 292
- virDomainSmartcardDefIterator, 268
- virDomainSmartcardDefPtr, 268
- virDomainSmartcardType, 283
- virDomainSmbiosMode, 283
- virDomainSnapshotObj, 268
- virDomainSnapshotObjList, 268
- virDomainSnapshotObjListPtr, 268
- virDomainSnapshotObjPtr, 268
- virDomainSoundCodecDef, 268
- virDomainSoundCodecDefFree, 292
- virDomainSoundCodecDefPtr, 268
- virDomainSoundCodecType, 283
- virDomainSoundDef, 268
- virDomainSoundDefFree, 292
- virDomainSoundDefPtr, 268
- virDomainSoundModel, 283
- virDomainStartupPolicy, 284
- virDomainStateReason, 268
- virDomainStateReasonFromString, 292
- virDomainStateReasonToString, 292
- virDomainTaintFlags, 284
- virDomainTimerCatchupDef, 268
- virDomainTimerCatchupDefPtr, 268
- virDomainTimerDef, 268
- virDomainTimerDefPtr, 268
- virDomainTimerModeType, 284
- virDomainTimerNameType, 284
- virDomainTimerTickpolicyType, 285
- virDomainTimerTrackType, 285
- virDomainVcpuPinAdd, 292
- virDomainVcpuPinDef, 268
- virDomainVcpuPinDefArrayFree, 292
- virDomainVcpuPinDefCopy, 292
- virDomainVcpuPinDefFree, 292
- virDomainVcpuPinDefPtr, 269
- virDomainVcpuPinDel, 292
- virDomainVcpuPinFindByVcpu, 292
- virDomainVcpuPinsDuplicate, 292
- virDomainVideoAccelDef, 269
- virDomainVideoAccelDefPtr, 269
- virDomainVideoDef, 269
- virDomainVideoDefFree, 292
- virDomainVideoDefPtr, 269
- virDomainVideoDefaultRAM, 292
- virDomainVideoDefaultType, 292
- virDomainVideoType, 285
- virDomainVirtType, 285
- virDomainVirtioEventIdx, 285
- virDomainVirtioSerialOpts, 269
- virDomainVirtioSerialOptsPtr, 269
- virDomainWatchdogAction, 286
- virDomainWatchdogDef, 269
- virDomainWatchdogDefFree, 292
- virDomainWatchdogDefPtr, 269
- virDomainWatchdogModel, 286
- virLifecycleFromStringFunc, 269
- virLifecycleToStringFunc, 269
- virSecurityDeviceLabelDef, 269
- virSecurityDeviceLabelDefPtr, 269
- virSecurityLabelDef, 269
- virSecurityLabelDefPtr, 269
- domainAbortJob
  - \_virDriver, 59
- domainAttachDevice
  - \_virDriver, 59
- domainAttachDeviceFlags
  - \_virDriver, 59
- domainBlockCommit
  - \_virDriver, 59
- domainBlockJobAbort
  - \_virDriver, 60
- domainBlockJobSetSpeed
  - \_virDriver, 60
- domainBlockPeek
  - \_virDriver, 60
- domainBlockPull
  - \_virDriver, 60
- domainBlockRebase
  - \_virDriver, 60
- domainBlockResize
  - \_virDriver, 60
- domainBlockStats
  - \_virDriver, 60
- domainBlockStatsFlags
  - \_virDriver, 60
- domainCoreDump
  - \_virDriver, 60
- domainCreate
  - \_virDriver, 60
- domainCreateWithFlags
  - \_virDriver, 60
- domainCreateXML
  - \_virDriver, 60
- domainDefineXML
  - \_virDriver, 60
- domainDestroy
  - \_virDriver, 60
- domainDestroyFlags
  - \_virDriver, 60
- domainDetachDevice
  - \_virDriver, 60
- domainDetachDeviceFlags
  - \_virDriver, 60
- domainEventDeregister
  - \_virDriver, 60
- domainEventDeregisterAny
  - \_virDriver, 60
- domainEventRegister



[\\_virDriver](#), [60](#)  
domainEventRegisterAny  
    [\\_virDriver](#), [60](#)  
domainGetAutostart  
    [\\_virDriver](#), [60](#)  
domainGetBlkioParameters  
    [\\_virDriver](#), [60](#)  
domainGetBlockInfo  
    [\\_virDriver](#), [60](#)  
domainGetBlockioTune  
    [\\_virDriver](#), [60](#)  
domainGetBlockJobInfo  
    [\\_virDriver](#), [60](#)  
domainGetCPUStats  
    [\\_virDriver](#), [60](#)  
domainGetControllInfo  
    [\\_virDriver](#), [60](#)  
domainGetDiskErrors  
    [\\_virDriver](#), [61](#)  
domainGetEmulatorPinInfo  
    [\\_virDriver](#), [61](#)  
domainGetHostname  
    [\\_virDriver](#), [61](#)  
domainGetInfo  
    [\\_virDriver](#), [61](#)  
domainGetInterfaceParameters  
    [\\_virDriver](#), [61](#)  
domainGetJobInfo  
    [\\_virDriver](#), [61](#)  
domainGetMaxMemory  
    [\\_virDriver](#), [61](#)  
domainGetMaxVcpus  
    [\\_virDriver](#), [61](#)  
domainGetMemoryParameters  
    [\\_virDriver](#), [61](#)  
domainGetMetadata  
    [\\_virDriver](#), [61](#)  
domainGetNumaParameters  
    [\\_virDriver](#), [61](#)  
domainGetOSType  
    [\\_virDriver](#), [61](#)  
domainGetSchedulerParameters  
    [\\_virDriver](#), [61](#)  
domainGetSchedulerParametersFlags  
    [\\_virDriver](#), [61](#)  
domainGetSchedulerType  
    [\\_virDriver](#), [61](#)  
domainGetSecurityLabel  
    [\\_virDriver](#), [61](#)  
domainGetSecurityLabelList  
    [\\_virDriver](#), [61](#)  
domainGetState  
    [\\_virDriver](#), [61](#)  
domainGetVcpuPinInfo  
    [\\_virDriver](#), [61](#)  
domainGetVcpus  
    [\\_virDriver](#), [61](#)  
domainGetVcpusFlags  
    [\\_virDriver](#), [61](#)  
domainGetXMLDesc  
    [\\_virDriver](#), [61](#)  
domainHasCurrentSnapshot  
    [\\_virDriver](#), [61](#)  
domainHasManagedSaveImage  
    [\\_virDriver](#), [61](#)  
domainInjectNMI  
    [\\_virDriver](#), [61](#)  
domainInterfaceStats  
    [\\_virDriver](#), [61](#)  
domainsIsActive  
    [\\_virDriver](#), [61](#)  
domainsIsPersistent  
    [\\_virDriver](#), [61](#)  
domainsIsUpdated  
    [\\_virDriver](#), [62](#)  
domainListAllSnapshots  
    [\\_virDriver](#), [62](#)  
domainLookupByID  
    [\\_virDriver](#), [62](#)  
domainLookupByName  
    [\\_virDriver](#), [62](#)  
domainLookupByUUID  
    [\\_virDriver](#), [62](#)  
domainManagedSave  
    [\\_virDriver](#), [62](#)  
domainManagedSaveRemove  
    [\\_virDriver](#), [62](#)  
domainMemoryPeek  
    [\\_virDriver](#), [62](#)  
domainMemoryStats  
    [\\_virDriver](#), [62](#)  
domainMigrateBegin3  
    [\\_virDriver](#), [62](#)  
domainMigrateConfirm3  
    [\\_virDriver](#), [62](#)  
domainMigrateFinish  
    [\\_virDriver](#), [62](#)  
domainMigrateFinish2  
    [\\_virDriver](#), [62](#)  
domainMigrateFinish3  
    [\\_virDriver](#), [62](#)  
domainMigrateGetMaxSpeed  
    [\\_virDriver](#), [62](#)  
domainMigratePerform  
    [\\_virDriver](#), [62](#)  
domainMigratePerform3  
    [\\_virDriver](#), [62](#)  
domainMigratePrepare  
    [\\_virDriver](#), [62](#)  
domainMigratePrepare2  
    [\\_virDriver](#), [62](#)  
domainMigratePrepare3  
    [\\_virDriver](#), [62](#)  
domainMigratePrepareTunnel  
    [\\_virDriver](#), [62](#)  
domainMigratePrepareTunnel3

- [\\_virDriver, 62](#)
- domainMigrateSetMaxDowntime
  - [\\_virDriver, 62](#)
- domainMigrateSetMaxSpeed
  - [\\_virDriver, 62](#)
- domainOpenConsole
  - [\\_virDriver, 62](#)
- domainOpenGraphics
  - [\\_virDriver, 62](#)
- domainPMSuspendForDuration
  - [\\_virDriver, 63](#)
- domainPMWakeup
  - [\\_virDriver, 63](#)
- domainPinEmulator
  - [\\_virDriver, 62](#)
- domainPinVcpu
  - [\\_virDriver, 62](#)
- domainPinVcpuFlags
  - [\\_virDriver, 63](#)
- domainReboot
  - [\\_virDriver, 63](#)
- domainReset
  - [\\_virDriver, 63](#)
- domainRestore
  - [\\_virDriver, 63](#)
- domainRestoreFlags
  - [\\_virDriver, 63](#)
- domainResume
  - [\\_virDriver, 63](#)
- domainRevertToSnapshot
  - [\\_virDriver, 63](#)
- domainSave
  - [\\_virDriver, 63](#)
- domainSaveFlags
  - [\\_virDriver, 63](#)
- domainSavelImageDefineXML
  - [\\_virDriver, 63](#)
- domainSavelImageGetXMLDesc
  - [\\_virDriver, 63](#)
- domainScreenshot
  - [\\_virDriver, 63](#)
- domainSendKey
  - [\\_virDriver, 63](#)
- domainSetAutostart
  - [\\_virDriver, 63](#)
- domainSetBlkioParameters
  - [\\_virDriver, 63](#)
- domainSetBlockioTune
  - [\\_virDriver, 63](#)
- domainSetInterfaceParameters
  - [\\_virDriver, 63](#)
- domainSetMaxMemory
  - [\\_virDriver, 63](#)
- domainSetMemory
  - [\\_virDriver, 63](#)
- domainSetMemoryFlags
  - [\\_virDriver, 63](#)
- domainSetMemoryParameters
  - [\\_virDriver, 63](#)
- [\\_virDriver, 63](#)
- domainSetMetadata
  - [\\_virDriver, 63](#)
- domainSetNumaParameters
  - [\\_virDriver, 63](#)
- domainSetSchedulerParameters
  - [\\_virDriver, 63](#)
- domainSetSchedulerParametersFlags
  - [\\_virDriver, 63](#)
- domainSetVcpus
  - [\\_virDriver, 63](#)
- domainSetVcpusFlags
  - [\\_virDriver, 64](#)
- domainShutdown
  - [\\_virDriver, 64](#)
- domainShutdownFlags
  - [\\_virDriver, 64](#)
- domainSnapshotCreateXML
  - [\\_virDriver, 64](#)
- domainSnapshotCurrent
  - [\\_virDriver, 64](#)
- domainSnapshotDelete
  - [\\_virDriver, 64](#)
- domainSnapshotGetParent
  - [\\_virDriver, 64](#)
- domainSnapshotGetXMLDesc
  - [\\_virDriver, 64](#)
- domainSnapshotHasMetadata
  - [\\_virDriver, 64](#)
- domainSnapshotIsCurrent
  - [\\_virDriver, 64](#)
- domainSnapshotListAllChildren
  - [\\_virDriver, 64](#)
- domainSnapshotListChildrenNames
  - [\\_virDriver, 64](#)
- domainSnapshotListNames
  - [\\_virDriver, 64](#)
- domainSnapshotLookupByName
  - [\\_virDriver, 64](#)
- domainSnapshotNum
  - [\\_virDriver, 64](#)
- domainSnapshotNumChildren
  - [\\_virDriver, 64](#)
- domainSuspend
  - [\\_virDriver, 64](#)
- domainUndefine
  - [\\_virDriver, 64](#)
- domainUndefineFlags
  - [\\_virDriver, 64](#)
- domainUpdateDeviceFlags
  - [\\_virDriver, 64](#)
- domainXMLFromNative
  - [\\_virDriver, 64](#)
- domainXMLToNative
  - [\\_virDriver, 64](#)
- domains
  - [virDomainListData, 91](#)
- drive

- [\\_virDomainDeviceInfo](#), 25
- driver
  - [\\_virDomainNetDef](#), 45
- driver.h
  - [VIR\\_DRV\\_ESX](#), 328
  - [VIR\\_DRV\\_HYPERV](#), 329
  - [VIR\\_DRV\\_LIBXL](#), 328
  - [VIR\\_DRV\\_LXC](#), 328
  - [VIR\\_DRV\\_ONE](#), 328
  - [VIR\\_DRV\\_OPEN\\_DECLINED](#), 329
  - [VIR\\_DRV\\_OPEN\\_ERROR](#), 329
  - [VIR\\_DRV\\_OPEN\\_SUCCESS](#), 329
  - [VIR\\_DRV\\_OPENVZ](#), 328
  - [VIR\\_DRV\\_PARALLELS](#), 329
  - [VIR\\_DRV\\_PHYP](#), 328
  - [VIR\\_DRV\\_QEMU](#), 328
  - [VIR\\_DRV\\_REMOTE](#), 328
  - [VIR\\_DRV\\_TEST](#), 328
  - [VIR\\_DRV\\_UML](#), 328
  - [VIR\\_DRV\\_VBOX](#), 328
  - [VIR\\_DRV\\_VMWARE](#), 328
  - [VIR\\_DRV\\_XEN\\_UNIFIED](#), 328
  - [VIR\\_DRV\\_XENAPI](#), 328
  - [VIR\\_SECRET\\_GET\\_VALUE\\_INTERNAL\\_CALL](#), 328
- driver.h
  - [virDevMonDeviceGetParent](#), 316
  - [virDevMonDeviceGetXMLDesc](#), 316
  - [virDevMonDeviceListCaps](#), 316
  - [virDevMonDeviceLookupByName](#), 316
  - [virDevMonDeviceNumOfCaps](#), 316
  - [virDevMonListAllNodeDevices](#), 316
  - [virDevMonListDevices](#), 316
  - [virDevMonNumOfDevices](#), 316
  - [virDeviceMonitor](#), 316
  - [virDeviceMonitorPtr](#), 316
  - [virDriver](#), 316
  - [virDriverLoadModule](#), 329
  - [virDriverModuleInitialize](#), 329
  - [virDriverPtr](#), 316
  - [virDrvBaselineCPU](#), 316
  - [virDrvClose](#), 316
  - [virDrvCompareCPU](#), 316
  - [virDrvConnectDomainXMLFromNative](#), 316
  - [virDrvConnectDomainXMLToNative](#), 316
  - [virDrvConnectFindStoragePoolSources](#), 316
  - [virDrvConnectIsAlive](#), 316
  - [virDrvConnectIsEncrypted](#), 316
  - [virDrvConnectIsSecure](#), 316
  - [virDrvConnectListAllNWFilters](#), 316
  - [virDrvConnectListAllStoragePools](#), 316
  - [virDrvConnectListDefinedStoragePools](#), 316
  - [virDrvConnectListNWFilters](#), 316
  - [virDrvConnectListStoragePools](#), 317
  - [virDrvConnectNumOfDefinedStoragePools](#), 317
  - [virDrvConnectNumOfNWFilters](#), 317
  - [virDrvConnectNumOfStoragePools](#), 317
  - [virDrvDomainAbortJob](#), 317
  - [virDrvDomainAttachDevice](#), 317
  - [virDrvDomainAttachDeviceFlags](#), 317
  - [virDrvDomainBlockCommit](#), 317
  - [virDrvDomainBlockJobAbort](#), 317
  - [virDrvDomainBlockJobSetSpeed](#), 317
  - [virDrvDomainBlockPeek](#), 317
  - [virDrvDomainBlockPull](#), 317
  - [virDrvDomainBlockRebase](#), 317
  - [virDrvDomainBlockResize](#), 317
  - [virDrvDomainBlockStats](#), 317
  - [virDrvDomainBlockStatsFlags](#), 317
  - [virDrvDomainCoreDump](#), 317
  - [virDrvDomainCreate](#), 317
  - [virDrvDomainCreateWithFlags](#), 317
  - [virDrvDomainCreateXML](#), 317
  - [virDrvDomainDefineXML](#), 317
  - [virDrvDomainDestroy](#), 317
  - [virDrvDomainDestroyFlags](#), 317
  - [virDrvDomainDetachDevice](#), 317
  - [virDrvDomainDetachDeviceFlags](#), 318
  - [virDrvDomainEventDeregister](#), 318
  - [virDrvDomainEventDeregisterAny](#), 318
  - [virDrvDomainEventRegister](#), 318
  - [virDrvDomainEventRegisterAny](#), 318
  - [virDrvDomainGetAutostart](#), 318
  - [virDrvDomainGetBlkioParameters](#), 318
  - [virDrvDomainGetBlockInfo](#), 318
  - [virDrvDomainGetBlockIoTune](#), 318
  - [virDrvDomainGetBlockJobInfo](#), 318
  - [virDrvDomainGetCPUStats](#), 318
  - [virDrvDomainGetControllInfo](#), 318
  - [virDrvDomainGetDiskErrors](#), 318
  - [virDrvDomainGetEmulatorPinInfo](#), 318
  - [virDrvDomainGetHostname](#), 318
  - [virDrvDomainGetInfo](#), 318
  - [virDrvDomainGetInterfaceParameters](#), 318
  - [virDrvDomainGetJobInfo](#), 318
  - [virDrvDomainGetMaxMemory](#), 318
  - [virDrvDomainGetMaxVcpus](#), 318
  - [virDrvDomainGetMemoryParameters](#), 318
  - [virDrvDomainGetMetadata](#), 318
  - [virDrvDomainGetNumaParameters](#), 319
  - [virDrvDomainGetOSType](#), 319
  - [virDrvDomainGetSchedulerParameters](#), 319
  - [virDrvDomainGetSchedulerParametersFlags](#), 319
  - [virDrvDomainGetSchedulerType](#), 319
  - [virDrvDomainGetSecurityLabel](#), 319
  - [virDrvDomainGetSecurityLabelList](#), 319
  - [virDrvDomainGetState](#), 319
  - [virDrvDomainGetVcpuPinInfo](#), 319
  - [virDrvDomainGetVcpus](#), 319
  - [virDrvDomainGetVcpusFlags](#), 319
  - [virDrvDomainGetXMLDesc](#), 319
  - [virDrvDomainHasCurrentSnapshot](#), 319
  - [virDrvDomainHasManagedSavableImage](#), 319
  - [virDrvDomainInjectNMI](#), 319
  - [virDrvDomainInterfaceStats](#), 319
  - [virDrvDomainIsActive](#), 319

- virDrvDomainIsPersistent, 319
- virDrvDomainIsUpdated, 319
- virDrvDomainListAllSnapshots, 319
- virDrvDomainLookupByID, 319
- virDrvDomainLookupByName, 319
- virDrvDomainLookupByUUID, 319
- virDrvDomainManagedSave, 319
- virDrvDomainManagedSaveRemove, 319
- virDrvDomainMemoryPeek, 320
- virDrvDomainMemoryStats, 320
- virDrvDomainMigrateBegin3, 320
- virDrvDomainMigrateConfirm3, 320
- virDrvDomainMigrateFinish, 320
- virDrvDomainMigrateFinish2, 320
- virDrvDomainMigrateFinish3, 320
- virDrvDomainMigrateGetMaxSpeed, 320
- virDrvDomainMigratePerform, 320
- virDrvDomainMigratePerform3, 320
- virDrvDomainMigratePrepare, 320
- virDrvDomainMigratePrepare2, 320
- virDrvDomainMigratePrepare3, 320
- virDrvDomainMigratePrepareTunnel, 320
- virDrvDomainMigratePrepareTunnel3, 320
- virDrvDomainMigrateSetMaxDowntime, 320
- virDrvDomainMigrateSetMaxSpeed, 320
- virDrvDomainOpenConsole, 320
- virDrvDomainOpenGraphics, 321
- virDrvDomainPMSuspendForDuration, 321
- virDrvDomainPMWakeup, 321
- virDrvDomainPinEmulator, 321
- virDrvDomainPinVcpu, 321
- virDrvDomainPinVcpuFlags, 321
- virDrvDomainQemuAgentCommand, 321
- virDrvDomainQemuAttach, 321
- virDrvDomainQemuMonitorCommand, 321
- virDrvDomainReboot, 321
- virDrvDomainReset, 321
- virDrvDomainRestore, 321
- virDrvDomainRestoreFlags, 321
- virDrvDomainResume, 321
- virDrvDomainRevertToSnapshot, 321
- virDrvDomainSave, 321
- virDrvDomainSaveFlags, 321
- virDrvDomainSaveImageDefineXML, 321
- virDrvDomainSaveImageGetXMLDesc, 321
- virDrvDomainScreenshot, 321
- virDrvDomainSendKey, 321
- virDrvDomainSetAutostart, 321
- virDrvDomainSetBlkioParameters, 321
- virDrvDomainSetBlockIoTune, 322
- virDrvDomainSetInterfaceParameters, 322
- virDrvDomainSetMaxMemory, 322
- virDrvDomainSetMemory, 322
- virDrvDomainSetMemoryFlags, 322
- virDrvDomainSetMemoryParameters, 322
- virDrvDomainSetMetadata, 322
- virDrvDomainSetNumaParameters, 322
- virDrvDomainSetSchedulerParameters, 322
- virDrvDomainSetSchedulerParametersFlags, 322
- virDrvDomainSetVcpus, 322
- virDrvDomainSetVcpusFlags, 322
- virDrvDomainShutdown, 322
- virDrvDomainShutdownFlags, 322
- virDrvDomainSnapshotCreateXML, 322
- virDrvDomainSnapshotCurrent, 322
- virDrvDomainSnapshotDelete, 322
- virDrvDomainSnapshotGetParent, 322
- virDrvDomainSnapshotGetXMLDesc, 322
- virDrvDomainSnapshotHasMetadata, 322
- virDrvDomainSnapshotIsCurrent, 322
- virDrvDomainSnapshotListAllChildren, 322
- virDrvDomainSnapshotListChildrenNames, 322
- virDrvDomainSnapshotListNames, 323
- virDrvDomainSnapshotLookupByName, 323
- virDrvDomainSnapshotNum, 323
- virDrvDomainSnapshotNumChildren, 323
- virDrvDomainSuspend, 323
- virDrvDomainUndefine, 323
- virDrvDomainUndefineFlags, 323
- virDrvDomainUpdateDeviceFlags, 323
- virDrvDomainSupportsFeature, 323
- virDrvGetCapabilities, 323
- virDrvGetHostname, 323
- virDrvGetLibVersion, 323
- virDrvGetMaxVcpus, 323
- virDrvGetSysinfo, 323
- virDrvGetType, 323
- virDrvGetURI, 323
- virDrvGetVersion, 323
- virDrvInterfaceChangeBegin, 323
- virDrvInterfaceChangeCommit, 323
- virDrvInterfaceChangeRollback, 323
- virDrvInterfaceCreate, 323
- virDrvInterfaceDefineXML, 323
- virDrvInterfaceDestroy, 323
- virDrvInterfaceGetXMLDesc, 323
- virDrvInterfaceIsActive, 323
- virDrvInterfaceLookupByMACString, 323
- virDrvInterfaceLookupByName, 324
- virDrvInterfaceUndefine, 324
- virDrvListAllDomains, 324
- virDrvListAllInterfaces, 324
- virDrvListAllNetworks, 324
- virDrvListAllSecrets, 324
- virDrvListDefinedDomains, 324
- virDrvListDefinedInterfaces, 324
- virDrvListDefinedNetworks, 324
- virDrvListDomains, 324
- virDrvListInterfaces, 324
- virDrvListNetworks, 324
- virDrvListSecrets, 324
- virDrvNWFilterDefineXML, 325
- virDrvNWFilterGetXMLDesc, 325
- virDrvNWFilterLookupByName, 325
- virDrvNWFilterLookupByUUID, 326
- virDrvNWFilterUndefine, 326

- virDrvNetworkCreate, [324](#)
- virDrvNetworkCreateXML, [324](#)
- virDrvNetworkDefineXML, [324](#)
- virDrvNetworkDestroy, [324](#)
- virDrvNetworkGetAutostart, [324](#)
- virDrvNetworkGetBridgeName, [324](#)
- virDrvNetworkGetBridgeType, [324](#)
- virDrvNetworkGetXMLDesc, [324](#)
- virDrvNetworkIsActive, [324](#)
- virDrvNetworkIsPersistent, [324](#)
- virDrvNetworkLookupByName, [324](#)
- virDrvNetworkLookupByUUID, [324](#)
- virDrvNetworkSetAutostart, [324](#)
- virDrvNetworkUndefine, [324](#)
- virDrvNetworkUpdate, [324](#)
- virDrvNo, [328](#)
- virDrvNodeDeviceCreateXML, [325](#)
- virDrvNodeDeviceDestroy, [325](#)
- virDrvNodeDeviceDetach, [325](#)
- virDrvNodeDeviceReAttach, [325](#)
- virDrvNodeDeviceReset, [325](#)
- virDrvNodeGetCPUStats, [325](#)
- virDrvNodeGetCellsFreeMemory, [325](#)
- virDrvNodeGetFreeMemory, [325](#)
- virDrvNodeGetInfo, [325](#)
- virDrvNodeGetMemoryParameters, [325](#)
- virDrvNodeGetMemoryStats, [325](#)
- virDrvNodeGetSecurityModel, [325](#)
- virDrvNodeSetMemoryParameters, [325](#)
- virDrvNodeSuspendForDuration, [325](#)
- virDrvNumOfDefinedDomains, [325](#)
- virDrvNumOfDefinedInterfaces, [325](#)
- virDrvNumOfDefinedNetworks, [325](#)
- virDrvNumOfDomains, [325](#)
- virDrvNumOfInterfaces, [325](#)
- virDrvNumOfNetworks, [325](#)
- virDrvNumOfSecrets, [325](#)
- virDrvOpen, [326](#)
- virDrvOpenStatus, [329](#)
- virDrvSecretDefineXML, [326](#)
- virDrvSecretGetValue, [326](#)
- virDrvSecretGetXMLDesc, [326](#)
- virDrvSecretLookupByUUID, [326](#)
- virDrvSecretLookupByUsage, [326](#)
- virDrvSecretSetValue, [326](#)
- virDrvSecretUndefine, [326](#)
- virDrvSetKeepAlive, [326](#)
- virDrvStoragePoolBuild, [326](#)
- virDrvStoragePoolCreate, [326](#)
- virDrvStoragePoolCreateXML, [326](#)
- virDrvStoragePoolDefineXML, [326](#)
- virDrvStoragePoolDelete, [326](#)
- virDrvStoragePoolDestroy, [326](#)
- virDrvStoragePoolGetAutostart, [326](#)
- virDrvStoragePoolGetInfo, [326](#)
- virDrvStoragePoolGetXMLDesc, [326](#)
- virDrvStoragePoolIsActive, [326](#)
- virDrvStoragePoolIsPersistent, [326](#)
- virDrvStoragePoolListAllVolumes, [326](#)
- virDrvStoragePoolListVolumes, [326](#)
- virDrvStoragePoolLookupByName, [326](#)
- virDrvStoragePoolLookupByUUID, [326](#)
- virDrvStoragePoolLookupByVolume, [327](#)
- virDrvStoragePoolNumOfVolumes, [327](#)
- virDrvStoragePoolRefresh, [327](#)
- virDrvStoragePoolSetAutostart, [327](#)
- virDrvStoragePoolUndefine, [327](#)
- virDrvStorageVolCreateXML, [327](#)
- virDrvStorageVolCreateXMLFrom, [327](#)
- virDrvStorageVolDelete, [327](#)
- virDrvStorageVolDownload, [327](#)
- virDrvStorageVolGetInfo, [327](#)
- virDrvStorageVolGetPath, [327](#)
- virDrvStorageVolGetXMLDesc, [327](#)
- virDrvStorageVolLookupByKey, [327](#)
- virDrvStorageVolLookupByName, [327](#)
- virDrvStorageVolLookupByPath, [327](#)
- virDrvStorageVolResize, [327](#)
- virDrvStorageVolUpload, [327](#)
- virDrvStorageVolWipe, [327](#)
- virDrvStorageVolWipePattern, [327](#)
- virDrvStreamAbort, [327](#)
- virDrvStreamEventAddCallback, [327](#)
- virDrvStreamEventRemoveCallback, [327](#)
- virDrvStreamEventUpdateCallback, [327](#)
- virDrvStreamFinish, [327](#)
- virDrvStreamRecv, [327](#)
- virDrvStreamSend, [328](#)
- virInterfaceDriver, [328](#)
- virInterfaceDriverPtr, [328](#)
- virNWFilterDriver, [328](#)
- virNWFilterDriverPtr, [328](#)
- virNetworkDriver, [328](#)
- virNetworkDriverPtr, [328](#)
- virRegisterDeviceMonitor, [329](#)
- virRegisterDriver, [329](#)
- virRegisterInterfaceDriver, [329](#)
- virRegisterNWFilterDriver, [329](#)
- virRegisterNetworkDriver, [329](#)
- virRegisterSecretDriver, [329](#)
- virRegisterStorageDriver, [330](#)
- virSecretDriver, [328](#)
- virSecretDriverPtr, [328](#)
- virStorageDriver, [328](#)
- virStorageDriverPtr, [328](#)
- virStreamDriver, [328](#)
- virStreamDriverPtr, [328](#)
- driverName
  - \_virDomainDiskDef, [29](#)
- driverState
  - bridge\_driver.c, [428](#)
- driverType
  - \_virDomainDiskDef, [29](#)
- dst
  - \_virDomainDiskDef, [29](#)
  - \_virDomainFSDef, [33](#)

- dummy
  - [\\_virDomainHostdevDef, 37](#)
- dump\_core
  - [\\_virDomainDef, 20](#)
- EDIT\_DEFINE
  - [virsh-network.c, 450](#)
- EDIT\_FREE
  - [virsh-network.c, 450](#)
- EDIT\_GET\_XML
  - [virsh-network.c, 450](#)
- EDIT\_NOT\_CHANGED
  - [virsh-network.c, 450](#)
- emulator
  - [\\_virDomainDef, 20](#)
- emulator\_period
  - [\\_virDomainDef, 20](#)
- emulator\_quota
  - [\\_virDomainDef, 20](#)
- emulatorpin
  - [\\_virDomainDef, 20](#)
- encryption
  - [\\_virDomainDiskDef, 29](#)
- end
  - [\\_virDomainBlockJobInfo, 13](#)
  - [\\_virNetworkDHCPRangeDef, 70](#)
- enforcing
  - [\\_virSecurityLabel, 82](#)
- error
  - [\\_virDomainDiskError, 31](#)
  - [virDomainListData, 91](#)
- error\_policy
  - [\\_virDomainDiskDef, 29](#)
- errs
  - [\\_virDomainBlockStats, 13](#)
- ethernet
  - [\\_virDomainNetDef, 45](#)
- event\_idx
  - [\\_virDomainDiskDef, 29](#)
  - [\\_virDomainNetDef, 45](#)
- expires
  - [\\_virDomainGraphicsAuthDef, 34](#)
- family
  - [\\_virDomainEventGraphicsAddress, 32](#)
  - [\\_virNetworkIpDef, 75](#)
- features
  - [\\_virDomainDef, 20](#)
- field
  - [\\_virNodeCPUStats, 77](#)
  - [\\_virNodeMemoryStats, 78](#)
  - [\\_virTypedParameter, 88](#)
- file
  - [\\_virDomainChrSourceDef, 15](#)
  - [\\_virDomainSmartcardDef, 51](#)
- fileProcessed
  - [\\_virDomainJobInfo, 41](#)
- fileRemaining
  - [\\_virDomainJobInfo, 41](#)
- fileTotal
  - [\\_virDomainJobInfo, 41](#)
- filter
  - [\\_virDomainNetDef, 45](#)
- filterparams
  - [\\_virDomainNetDef, 45](#)
- findPoolSources
  - [\\_virStorageDriver, 84](#)
- flags
  - [virDomainListData, 91](#)
- forwardIfs
  - [\\_virNetworkDef, 68](#)
- forwardPFs
  - [\\_virNetworkDef, 68](#)
- forwardType
  - [\\_virNetworkDef, 68](#)
- frequency
  - [\\_virDomainTimerDef, 53](#)
- fs
  - [\\_virDomainDeviceDef, 24](#)
- fsdriver
  - [\\_virDomainFSDef, 33](#)
- fss
  - [\\_virDomainDef, 20](#)
- fullscreen
  - [\\_virDomainGraphicsDef, 35](#)
- geometry
  - [\\_virDomainDiskDef, 29](#)
- getCapabilities
  - [\\_virDriver, 64](#)
- getHostname
  - [\\_virDriver, 64](#)
- getMaxVcpus
  - [\\_virDriver, 64](#)
- getSysinfo
  - [\\_virDriver, 64](#)
- getValue
  - [\\_virSecretDriver, 80](#)
- getXMLDesc
  - [\\_virNWFilterDriver, 79](#)
  - [\\_virSecretDriver, 81](#)
- graphics
  - [\\_virDomainDef, 20](#)
  - [\\_virDomainDeviceDef, 24](#)
- hard\_limit
  - [\\_virDomainDef, 20](#)
- has\_reg
  - [\\_virDomainDeviceSpaprVioAddress, 26](#)
- hasManagedSave
  - [\\_virDomainObj, 47](#)
- heads
  - [\\_virDomainDiskDef, 29](#)
  - [\\_virDomainVideoDef, 55](#)
- host
  - [\\_virDomainChrSourceDef, 15](#)
- hostdev
  - [\\_virDomainActualNetDef, 10](#)

- [\\_virDomainDeviceDef](#), 24
  - [\\_virDomainNetDef](#), 45
- hostdevs
  - [\\_virDomainDef](#), 20
- hosts
  - [\\_virDomainDiskDef](#), 29
  - [\\_virNetworkDNSDef](#), 70
  - [\\_virNetworkIpfDef](#), 75
- hub
  - [\\_virDomainDeviceDef](#), 24
- hubs
  - [\\_virDomainDef](#), 20
- hugepage\_backed
  - [\\_virDomainDef](#), 20
- i
  - [\\_virTypedParameter](#), 88
- IS\_USB2\_CONTROLLER
  - [qemu\\_command.c](#), 432
- id
  - [\\_virDomainDef](#), 20
- identities
  - [\\_virDomainEventGraphicsSubject](#), 32
- ids
  - [virDomainIDDData](#), 90
- idx
  - [\\_virDomainControllerDef](#), 17
- ifname
  - [\\_virDomainNetDef](#), 45
  - [virnetdevopenvswitch.h](#), 445
- image
  - [\\_virDomainGraphicsDef](#), 36
- imagelabel
  - [\\_virSecurityLabelDef](#), 82
- implicit
  - [\\_virSecurityLabelDef](#), 82
- include/libvirt/libvirt.h, 93
- info
  - [\\_virDomainChrDef](#), 14
  - [\\_virDomainControllerDef](#), 17
  - [\\_virDomainDiskDef](#), 29
  - [\\_virDomainFSDef](#), 33
  - [\\_virDomainHostdevDef](#), 37
  - [\\_virDomainHubDef](#), 39
  - [\\_virDomainInputDef](#), 40
  - [\\_virDomainMemballoonDef](#), 43
  - [\\_virDomainNetDef](#), 45
  - [\\_virDomainRedirdevDef](#), 49
  - [\\_virDomainSmartcardDef](#), 51
  - [\\_virDomainSoundDef](#), 52
  - [\\_virDomainVideoDef](#), 55
  - [\\_virDomainWatchdogDef](#), 55
- info\_network\_autostart
  - [virsh-network.c](#), 451
- info\_network\_create
  - [virsh-network.c](#), 451
- info\_network\_define
  - [virsh-network.c](#), 451
- info\_network\_destroy
  - [virsh-network.c](#), 452
- info\_network\_dumpxml
  - [virsh-network.c](#), 452
- info\_network\_edit
  - [virsh-network.c](#), 452
- info\_network\_info
  - [virsh-network.c](#), 452
- info\_network\_list
  - [virsh-network.c](#), 452
- info\_network\_name
  - [virsh-network.c](#), 452
- info\_network\_start
  - [virsh-network.c](#), 453
- info\_network\_undefine
  - [virsh-network.c](#), 453
- info\_network\_update
  - [virsh-network.c](#), 453
- info\_network\_uuid
  - [virsh-network.c](#), 453
- init
  - [\\_virDomainOSDef](#), 48
- initargv
  - [\\_virDomainOSDef](#), 49
- initialized
  - [libvirt.c](#), 419
- initrd
  - [\\_virDomainOSDef](#), 49
- input
  - [\\_virDomainDeviceDef](#), 24
- inputs
  - [\\_virDomainDef](#), 20
- interfaceChangeBegin
  - [\\_virInterfaceDriver](#), 66
- interfaceChangeCommit
  - [\\_virInterfaceDriver](#), 66
- interfaceChangeRollback
  - [\\_virInterfaceDriver](#), 66
- interfaceCreate
  - [\\_virInterfaceDriver](#), 67
- interfaceDefineXML
  - [\\_virInterfaceDriver](#), 67
- interfaceDestroy
  - [\\_virInterfaceDriver](#), 67
- interfaceGetXMLDesc
  - [\\_virInterfaceDriver](#), 67
- interfacelsActive
  - [\\_virInterfaceDriver](#), 67
- interfaceLookupByMACString
  - [\\_virInterfaceDriver](#), 67
- interfaceLookupByName
  - [\\_virInterfaceDriver](#), 67
- interfaceUndefine
  - [\\_virInterfaceDriver](#), 67
- internal
  - [\\_virDomainNetDef](#), 45
- ioeventfd
  - [\\_virDomainDiskDef](#), 29
  - [\\_virDomainNetDef](#), 45



- iomode
  - [\\_virDomainDiskDef, 29](#)
- ip
  - [\\_virNetworkDHCPHostDef, 69](#)
  - [\\_virNetworkDNSHostsDef, 71](#)
- ipaddr
  - [\\_virDomainNetDef, 45](#)
- ips
  - [\\_virNetworkDef, 68](#)
- iptables
  - [network\\_driver, 90](#)
- isAlive
  - [\\_virDriver, 64](#)
- isDefault
  - [\\_virPortGroupDef, 80](#)
- isEncrypted
  - [\\_virDriver, 64](#)
- isSecure
  - [\\_virDriver, 65](#)
- jpeg
  - [\\_virDomainGraphicsDef, 36](#)
- kernel
  - [\\_virDomainOSDef, 49](#)
- key
  - [\\_virDomainLeaseDef, 42](#)
- keymap
  - [\\_virDomainGraphicsDef, 36](#)
- l
  - [\\_virTypedParameter, 88](#)
- label
  - [\\_virSecurityDeviceLabelDef, 81](#)
  - [\\_virSecurityLabel, 82](#)
  - [\\_virSecurityLabelDef, 82](#)
- lease
  - [\\_virDomainDeviceDef, 24](#)
- leases
  - [\\_virDomainDef, 20](#)
- libvirt.h
  - [VIR\\_CONNECT\\_CLOSE\\_REASON\\_CLIENT, 132](#)
  - [VIR\\_CONNECT\\_CLOSE\\_REASON\\_EOF, 132](#)
  - [VIR\\_CONNECT\\_CLOSE\\_REASON\\_ERROR, 132](#)
  - [VIR\\_CONNECT\\_CLOSE\\_REASON\\_KEEPAIVE, 132](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_ACTIVE, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_AUTOSTART, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_HAS\\_SNAPSHOT, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_INACTIVE, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_MANAGEDSAVE, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_NO\\_AUTOSTART, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_NO\\_MANAGEDSAVE, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_NO\\_SNAPSHOT, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_OTHER, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_PAUSED, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_PERSISTENT, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_RUNNING, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_SHUTOFF, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_TRANSIENT, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_INTERFACES\\_ACTIVE, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_INTERFACES\\_INACTIVE, 133](#)
  - [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_ACTIVE, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_AUTOSTART, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_INACTIVE, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_NO\\_AUTOSTART, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_PERSISTENT, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_TRANSIENT, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_NET, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_PCI\\_DEV, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_SCSI, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_SCSI\\_HOST, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_SCSI\\_TARGET, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_STORAGE, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_SYSTEM, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_USB\\_DEV, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_USB\\_INTERFACE, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_SECRETS\\_EPHEMERAL, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_SECRETS\\_NO\\_EPHEMERAL, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_SECRETS\\_NO\\_PRIVATE, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_SECRETS\\_PRIVATE, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_ACTIVE, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_AUTOSTART, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_DIR, 134](#)
  - [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_DISK, 134](#)



- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_FS, [134](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_INACTIVE, [134](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_ISCSI, [135](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_LOGICAL, [134](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_MPATH, [135](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_NETFS, [134](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_NO\_AUTOSTART, [134](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_PERSISTENT, [134](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_RBD, [135](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_SCSI, [135](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_SHEPDOG, [135](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_TRANSIENT, [134](#)
- VIR\_CONNECT\_NO\_ALIASES, [133](#)
- VIR\_CONNECT\_RO, [133](#)
- VIR\_CPU\_COMPARE\_ERROR, [135](#)
- VIR\_CPU\_COMPARE\_IDENTICAL, [135](#)
- VIR\_CPU\_COMPARE\_INCOMPATIBLE, [135](#)
- VIR\_CPU\_COMPARE\_SUPERSET, [135](#)
- VIR\_CRED\_AUTHNAME, [132](#)
- VIR\_CRED\_CNONCE, [132](#)
- VIR\_CRED\_ECHOPROMPT, [132](#)
- VIR\_CRED\_EXTERNAL, [132](#)
- VIR\_CRED\_LANGUAGE, [132](#)
- VIR\_CRED\_NOECHOPROMPT, [132](#)
- VIR\_CRED\_PASSPHRASE, [132](#)
- VIR\_CRED\_REALM, [132](#)
- VIR\_CRED\_USERNAME, [132](#)
- VIR\_DOMAIN\_AFFECT\_CONFIG, [143](#)
- VIR\_DOMAIN\_AFFECT\_CURRENT, [143](#)
- VIR\_DOMAIN\_AFFECT\_LIVE, [143](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_BOOLEAN, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_DOUBLE, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_INT, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_LLONG, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_UINT, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_ULLONG, [132](#)
- VIR\_DOMAIN\_BLOCK\_COMMIT\_DELETE, [135](#)
- VIR\_DOMAIN\_BLOCK\_COMMIT\_SHALLOW, [135](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_ASYNC, [135](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_PIVOT, [135](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_CANCELED, [132](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_COMPLETED, [132](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_FAILED, [132](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_COMMIT, [136](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_COPY, [136](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_PULL, [136](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_UNKNOWN, [136](#)
- VIR\_DOMAIN\_BLOCK\_REBASE\_COPY, [136](#)
- VIR\_DOMAIN\_BLOCK\_REBASE\_COPY\_RAW, [136](#)
- VIR\_DOMAIN\_BLOCK\_REBASE\_REUSE\_EXT, [136](#)
- VIR\_DOMAIN\_BLOCK\_REBASE\_SHALLOW, [136](#)
- VIR\_DOMAIN\_BLOCK\_RESIZE\_BYTES, [136](#)
- VIR\_DOMAIN\_BLOCKED, [146](#)
- VIR\_DOMAIN\_BLOCKED\_UNKNOWN, [135](#)
- VIR\_DOMAIN\_CONSOLE\_FORCE, [136](#)
- VIR\_DOMAIN\_CONSOLE\_SAFE, [136](#)
- VIR\_DOMAIN\_CONTROL\_ERROR, [136](#)
- VIR\_DOMAIN\_CONTROL\_JOB, [136](#)
- VIR\_DOMAIN\_CONTROL\_OCCUPIED, [136](#)
- VIR\_DOMAIN\_CONTROL\_OK, [136](#)
- VIR\_DOMAIN\_CRASHED, [146](#)
- VIR\_DOMAIN\_CRASHED\_UNKNOWN, [137](#)
- VIR\_DOMAIN\_DESTROY\_DEFAULT, [137](#)
- VIR\_DOMAIN\_DESTROY\_GRACEFUL, [137](#)
- VIR\_DOMAIN\_DEVICE\_MODIFY\_CONFIG, [137](#)
- VIR\_DOMAIN\_DEVICE\_MODIFY\_CURRENT, [137](#)
- VIR\_DOMAIN\_DEVICE\_MODIFY\_FORCE, [137](#)
- VIR\_DOMAIN\_DEVICE\_MODIFY\_LIVE, [137](#)
- VIR\_DOMAIN\_DISK\_ERROR\_NO\_SPACE, [138](#)
- VIR\_DOMAIN\_DISK\_ERROR\_NONE, [138](#)
- VIR\_DOMAIN\_DISK\_ERROR\_UNSPEC, [138](#)
- VIR\_DOMAIN\_EVENT\_DEFINED, [141](#)
- VIR\_DOMAIN\_EVENT\_DEFINED\_ADDED, [138](#)
- VIR\_DOMAIN\_EVENT\_DEFINED\_UPDATED, [138](#)
- VIR\_DOMAIN\_EVENT\_DISK\_CHANGE\_MISSING\_ON\_START, [133](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_IPV4, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_IPV6, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_UNIX, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_CONNECT, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_DISCONNECT, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_INITIALIZE, [138](#)
- VIR\_DOMAIN\_EVENT\_ID\_BALLOON\_CHANGE, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_BLOCK\_JOB, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_CONTROL\_ERROR, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_DISK\_CHANGE, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_GRAPHICS, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_IO\_ERROR, [138](#)
- VIR\_DOMAIN\_EVENT\_ID\_IO\_ERROR\_REASON,

- 139
- VIR\_DOMAIN\_EVENT\_ID\_LIFECYCLE, 138
- VIR\_DOMAIN\_EVENT\_ID\_PMSUSPEND, 139
- VIR\_DOMAIN\_EVENT\_ID\_PMWAKEUP, 139
- VIR\_DOMAIN\_EVENT\_ID\_REBOOT, 138
- VIR\_DOMAIN\_EVENT\_ID\_RTC\_CHANGE, 138
- VIR\_DOMAIN\_EVENT\_ID\_TRAY\_CHANGE, 139
- VIR\_DOMAIN\_EVENT\_ID\_WATCHDOG, 138
- VIR\_DOMAIN\_EVENT\_IO\_ERROR\_NONE, 139
- VIR\_DOMAIN\_EVENT\_IO\_ERROR\_PAUSE, 139
- VIR\_DOMAIN\_EVENT\_IO\_ERROR\_REPORT, 139
- VIR\_DOMAIN\_EVENT\_PMSUSPENDED, 141
- VIR\_DOMAIN\_EVENT\_PMSUSPENDED\_MEMORY, 139
- VIR\_DOMAIN\_EVENT\_RESUMED, 141
- VIR\_DOMAIN\_EVENT\_RESUMED\_FROM\_SNAPSHOT, 139
- VIR\_DOMAIN\_EVENT\_RESUMED\_MIGRATED, 139
- VIR\_DOMAIN\_EVENT\_RESUMED\_UNPAUSED, 139
- VIR\_DOMAIN\_EVENT\_SHUTDOWN, 141
- VIR\_DOMAIN\_EVENT\_SHUTDOWN\_FINISHED, 139
- VIR\_DOMAIN\_EVENT\_STARTED, 141
- VIR\_DOMAIN\_EVENT\_STARTED\_BOOTED, 140
- VIR\_DOMAIN\_EVENT\_STARTED\_FROM\_SNAPSHOT, 140
- VIR\_DOMAIN\_EVENT\_STARTED\_MIGRATED, 140
- VIR\_DOMAIN\_EVENT\_STARTED\_RESTORED, 140
- VIR\_DOMAIN\_EVENT\_STARTED\_WAKEUP, 140
- VIR\_DOMAIN\_EVENT\_STOPPED, 141
- VIR\_DOMAIN\_EVENT\_STOPPED\_CRASHED, 140
- VIR\_DOMAIN\_EVENT\_STOPPED\_DESTROYED, 140
- VIR\_DOMAIN\_EVENT\_STOPPED\_FAILED, 140
- VIR\_DOMAIN\_EVENT\_STOPPED\_FROM\_SNAPSHOT, 140
- VIR\_DOMAIN\_EVENT\_STOPPED\_MIGRATED, 140
- VIR\_DOMAIN\_EVENT\_STOPPED\_SAVED, 140
- VIR\_DOMAIN\_EVENT\_STOPPED\_SHUTDOWN, 140
- VIR\_DOMAIN\_EVENT\_SUSPENDED, 141
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_FROM\_SNAPSHOT, 140
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_IOERROR, 140
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_MIGRATED, 140
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_PAUSED, 140
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_RESTORED, 140
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_WATCHDOG, 140
- VIR\_DOMAIN\_EVENT\_TRAY\_CHANGE\_CLOSE, 140
- VIR\_DOMAIN\_EVENT\_TRAY\_CHANGE\_OPEN, 140
- VIR\_DOMAIN\_EVENT\_UNDEFINED, 141
- VIR\_DOMAIN\_EVENT\_UNDEFINED\_REMOVED, 141
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_DEBUG, 141
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_NONE, 141
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_PAUSE, 141
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_POWEROFF, 141
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_RESET, 141
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_SHUTDOWN, 141
- VIR\_DOMAIN\_JOB\_BOUNDED, 141
- VIR\_DOMAIN\_JOB\_CANCELLED, 141
- VIR\_DOMAIN\_JOB\_COMPLETED, 141
- VIR\_DOMAIN\_JOB\_FAILED, 141
- VIR\_DOMAIN\_JOB\_NONE, 141
- VIR\_DOMAIN\_JOB\_UNBOUNDED, 141
- VIR\_DOMAIN\_MEM\_CONFIG, 142
- VIR\_DOMAIN\_MEM\_CURRENT, 142
- VIR\_DOMAIN\_MEM\_LIVE, 142
- VIR\_DOMAIN\_MEM\_MAXIMUM, 142
- VIR\_DOMAIN\_MEMORY\_PARAM\_BOOLEAN, 148
- VIR\_DOMAIN\_MEMORY\_PARAM\_DOUBLE, 148
- VIR\_DOMAIN\_MEMORY\_PARAM\_INT, 148
- VIR\_DOMAIN\_MEMORY\_PARAM\_LLONG, 148
- VIR\_DOMAIN\_MEMORY\_PARAM\_UINT, 148
- VIR\_DOMAIN\_MEMORY\_PARAM\_ULLONG, 148
- VIR\_DOMAIN\_MEMORY\_STAT\_ACTUAL\_BALLOON, 142
- VIR\_DOMAIN\_MEMORY\_STAT\_AVAILABLE, 142
- VIR\_DOMAIN\_MEMORY\_STAT\_MAJOR\_FAULT, 142
- VIR\_DOMAIN\_MEMORY\_STAT\_MINOR\_FAULT, 142
- VIR\_DOMAIN\_MEMORY\_STAT\_NR, 142
- VIR\_DOMAIN\_MEMORY\_STAT\_RSS, 142
- VIR\_DOMAIN\_MEMORY\_STAT\_SWAP\_IN, 142
- VIR\_DOMAIN\_MEMORY\_STAT\_SWAP\_OUT, 142
- VIR\_DOMAIN\_MEMORY\_STAT\_UNUSED, 142
- VIR\_DOMAIN\_METADATA\_DESCRIPTION, 142
- VIR\_DOMAIN\_METADATA\_ELEMENT, 142
- VIR\_DOMAIN\_METADATA\_TITLE, 142
- VIR\_DOMAIN\_NONE, 137
- VIR\_DOMAIN\_NOSTATE, 146
- VIR\_DOMAIN\_NOSTATE\_UNKNOWN, 143
- VIR\_DOMAIN\_NUMATUNE\_MEM\_INTERLEAVE, 143

- VIR\_DOMAIN\_NUMATUNE\_MEM\_PREFERRED, [143](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_STRICT, [143](#)
- VIR\_DOMAIN\_OPEN\_GRAPHICS\_SKIPAUTH, [143](#)
- VIR\_DOMAIN\_PAUSED, [146](#)
- VIR\_DOMAIN\_PAUSED\_DUMP, [144](#)
- VIR\_DOMAIN\_PAUSED\_FROM\_SNAPSHOT, [144](#)
- VIR\_DOMAIN\_PAUSED\_IOERROR, [144](#)
- VIR\_DOMAIN\_PAUSED\_MIGRATION, [143](#)
- VIR\_DOMAIN\_PAUSED\_SAVE, [144](#)
- VIR\_DOMAIN\_PAUSED\_SHUTTING\_DOWN, [144](#)
- VIR\_DOMAIN\_PAUSED\_UNKNOWN, [143](#)
- VIR\_DOMAIN\_PAUSED\_USER, [143](#)
- VIR\_DOMAIN\_PAUSED\_WATCHDOG, [144](#)
- VIR\_DOMAIN\_PMSUSPENDED, [146](#)
- VIR\_DOMAIN\_PMSUSPENDED\_UNKNOWN, [144](#)
- VIR\_DOMAIN\_REBOOT\_ACPI\_POWER\_BTN, [144](#)
- VIR\_DOMAIN\_REBOOT\_DEFAULT, [144](#)
- VIR\_DOMAIN\_REBOOT\_GUEST\_AGENT, [144](#)
- VIR\_DOMAIN\_RUNNING, [146](#)
- VIR\_DOMAIN\_RUNNING\_BOOTED, [144](#)
- VIR\_DOMAIN\_RUNNING\_FROM\_SNAPSHOT, [144](#)
- VIR\_DOMAIN\_RUNNING\_MIGRATED, [144](#)
- VIR\_DOMAIN\_RUNNING\_MIGRATION\_CANCELED, [144](#)
- VIR\_DOMAIN\_RUNNING\_RESTORED, [144](#)
- VIR\_DOMAIN\_RUNNING\_SAVE\_CANCELED, [144](#)
- VIR\_DOMAIN\_RUNNING\_UNKNOWN, [144](#)
- VIR\_DOMAIN\_RUNNING\_UNPAUSED, [144](#)
- VIR\_DOMAIN\_RUNNING\_WAKEUP, [144](#)
- VIR\_DOMAIN\_SAVE\_BYPASS\_CACHE, [144](#)
- VIR\_DOMAIN\_SAVE\_PAUSED, [144](#)
- VIR\_DOMAIN\_SAVE\_RUNNING, [144](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_BOOLEAN, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_DOUBLE, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_INT, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_LLONG, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_UINT, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_ULLONG, [150](#)
- VIR\_DOMAIN\_SHUTDOWN, [146](#)
- VIR\_DOMAIN\_SHUTDOWN\_ACPI\_POWER\_BTN, [145](#)
- VIR\_DOMAIN\_SHUTDOWN\_DEFAULT, [145](#)
- VIR\_DOMAIN\_SHUTDOWN\_GUEST\_AGENT, [145](#)
- VIR\_DOMAIN\_SHUTDOWN\_UNKNOWN, [145](#)
- VIR\_DOMAIN\_SHUTDOWN\_USER, [145](#)
- VIR\_DOMAIN\_SHUTOFF, [146](#)
- VIR\_DOMAIN\_SHUTOFF\_CRASHED, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_DESTROYED, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_FAILED, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_FROM\_SNAPSHOT, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_MIGRATED, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_SAVED, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_SHUTDOWN, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_UNKNOWN, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_ATOMIC, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_CURRENT, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_DISK\_ONLY, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_HALT, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_NO\_METADATA, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE QUIESCE, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_REDEFINE, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_REUSE\_EXT, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_DELETE\_CHILDREN, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_DELETE\_CHILDREN\_ONLY, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_DELETE\_METADATA\_ONLY, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_DESCENDANTS, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_LEAVES, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_METADATA, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_NO\_LEAVES, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_NO\_METADATA, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_ROOTS, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_REVERT\_FORCE, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_REVERT\_PAUSED, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_REVERT\_RUNNING, [146](#)
- VIR\_DOMAIN\_START\_AUTODESTROY, [137](#)
- VIR\_DOMAIN\_START\_BYPASS\_CACHE, [137](#)
- VIR\_DOMAIN\_START\_FORCE\_BOOT, [137](#)
- VIR\_DOMAIN\_START\_PAUSED, [137](#)
- VIR\_DOMAIN\_UNDEFINE\_MANAGED\_SAVE, [146](#)
- VIR\_DOMAIN\_UNDEFINE\_SNAPSHOTS\_METADATA, [146](#)
- VIR\_DOMAIN\_VCPU\_CONFIG, [147](#)
- VIR\_DOMAIN\_VCPU\_CURRENT, [147](#)
- VIR\_DOMAIN\_VCPU\_LIVE, [147](#)
- VIR\_DOMAIN\_VCPU\_MAXIMUM, [147](#)
- VIR\_DOMAIN\_XML\_INACTIVE, [147](#)
- VIR\_DOMAIN\_XML\_SECURE, [147](#)
- VIR\_DOMAIN\_XML\_UPDATE\_CPU, [147](#)
- VIR\_DUMP\_BYPASS\_CACHE, [137](#)
- VIR\_DUMP\_CRASH, [137](#)

- VIR\_DUMP\_LIVE, [137](#)
- VIR\_DUMP\_MEMORY\_ONLY, [137](#)
- VIR\_DUMP\_RESET, [137](#)
- VIR\_EVENT\_HANDLE\_ERROR, [147](#)
- VIR\_EVENT\_HANDLE\_HANGUP, [147](#)
- VIR\_EVENT\_HANDLE\_READABLE, [147](#)
- VIR\_EVENT\_HANDLE\_WRITABLE, [147](#)
- VIR\_INTERFACE\_XML\_INACTIVE, [147](#)
- VIR\_KEYCODE\_SET\_ATSET1, [147](#)
- VIR\_KEYCODE\_SET\_ATSET2, [147](#)
- VIR\_KEYCODE\_SET\_ATSET3, [147](#)
- VIR\_KEYCODE\_SET\_LINUX, [147](#)
- VIR\_KEYCODE\_SET\_OSX, [148](#)
- VIR\_KEYCODE\_SET\_RFB, [148](#)
- VIR\_KEYCODE\_SET\_USB, [148](#)
- VIR\_KEYCODE\_SET\_WIN32, [148](#)
- VIR\_KEYCODE\_SET\_XT, [147](#)
- VIR\_KEYCODE\_SET\_XT\_KBD, [148](#)
- VIR\_MEMORY\_PHYSICAL, [142](#)
- VIR\_MEMORY\_VIRTUAL, [142](#)
- VIR\_MIGRATE\_CHANGE\_PROTECTION, [143](#)
- VIR\_MIGRATE\_LIVE, [142](#)
- VIR\_MIGRATE\_NON\_SHARED\_DISK, [142](#)
- VIR\_MIGRATE\_NON\_SHARED\_INC, [143](#)
- VIR\_MIGRATE\_PAUSED, [142](#)
- VIR\_MIGRATE\_PEER2PEER, [142](#)
- VIR\_MIGRATE\_PERSIST\_DEST, [142](#)
- VIR\_MIGRATE\_TUNNELLED, [142](#)
- VIR\_MIGRATE\_UNDEFINE\_SOURCE, [142](#)
- VIR\_MIGRATE\_UNSAFE, [143](#)
- VIR\_NETWORK\_SECTION\_BRIDGE, [149](#)
- VIR\_NETWORK\_SECTION\_DNS\_HOST, [149](#)
- VIR\_NETWORK\_SECTION\_DNS\_SRV, [149](#)
- VIR\_NETWORK\_SECTION\_DNS\_TXT, [149](#)
- VIR\_NETWORK\_SECTION\_DOMAIN, [149](#)
- VIR\_NETWORK\_SECTION\_FORWARD, [149](#)
- VIR\_NETWORK\_SECTION\_FORWARD\_INTERFACE, [149](#)
- VIR\_NETWORK\_SECTION\_FORWARD\_PF, [149](#)
- VIR\_NETWORK\_SECTION\_IP, [149](#)
- VIR\_NETWORK\_SECTION\_IP\_DHCP\_HOST, [149](#)
- VIR\_NETWORK\_SECTION\_IP\_DHCP\_RANGE, [149](#)
- VIR\_NETWORK\_SECTION\_NONE, [149](#)
- VIR\_NETWORK\_SECTION\_PORTGROUP, [149](#)
- VIR\_NETWORK\_SECTION\_TUNNEL, [149](#)
- VIR\_NETWORK\_UPDATE\_AFFECT\_CONFIG, [148](#)
- VIR\_NETWORK\_UPDATE\_AFFECT\_CURRENT, [148](#)
- VIR\_NETWORK\_UPDATE\_AFFECT\_LIVE, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_ADD\_FIRST, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_ADD\_LAST, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_DELETE, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_MODIFY, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_NONE, [148](#)
- VIR\_NETWORK\_XML\_INACTIVE, [149](#)
- VIR\_NODE\_CPU\_STATS\_ALL\_CPUS, [149](#)
- VIR\_NODE\_MEMORY\_STATS\_ALL\_CELLS, [149](#)
- VIR\_NODE\_SUSPEND\_TARGET\_DISK, [150](#)
- VIR\_NODE\_SUSPEND\_TARGET\_HYBRID, [150](#)
- VIR\_NODE\_SUSPEND\_TARGET\_MEM, [150](#)
- VIR\_SECRET\_USAGE\_TYPE\_CEPH, [150](#)
- VIR\_SECRET\_USAGE\_TYPE\_NONE, [150](#)
- VIR\_SECRET\_USAGE\_TYPE\_VOLUME, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_NEW, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_NO\_OVERWRITE, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_OVERWRITE, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_REPAIR, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_RESIZE, [150](#)
- VIR\_STORAGE\_POOL\_BUILDING, [151](#)
- VIR\_STORAGE\_POOL\_DEGRADED, [151](#)
- VIR\_STORAGE\_POOL\_DELETE\_NORMAL, [150](#)
- VIR\_STORAGE\_POOL\_DELETE\_ZEROED, [150](#)
- VIR\_STORAGE\_POOL\_INACCESSIBLE, [151](#)
- VIR\_STORAGE\_POOL\_INACTIVE, [151](#)
- VIR\_STORAGE\_POOL\_RUNNING, [151](#)
- VIR\_STORAGE\_VOL\_BLOCK, [151](#)
- VIR\_STORAGE\_VOL\_DELETE\_NORMAL, [151](#)
- VIR\_STORAGE\_VOL\_DELETE\_ZEROED, [151](#)
- VIR\_STORAGE\_VOL\_DIR, [151](#)
- VIR\_STORAGE\_VOL\_FILE, [151](#)
- VIR\_STORAGE\_VOL\_NETWORK, [151](#)
- VIR\_STORAGE\_VOL\_RESIZE\_ALLOCATE, [151](#)
- VIR\_STORAGE\_VOL\_RESIZE\_DELTA, [151](#)
- VIR\_STORAGE\_VOL\_RESIZE\_SHRINK, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_BSI, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_DOD, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_GUTMANN, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_NNSA, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_PFITZNER33, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_PFITZNER7, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_RANDOM, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_SCHNEIER, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_ZERO, [151](#)
- VIR\_STORAGE\_XML\_INACTIVE, [152](#)
- VIR\_STREAM\_EVENT\_ERROR, [152](#)
- VIR\_STREAM\_EVENT\_HANGUP, [152](#)
- VIR\_STREAM\_EVENT\_READABLE, [152](#)
- VIR\_STREAM\_EVENT\_WRITABLE, [152](#)
- VIR\_STREAM\_NONBLOCK, [152](#)
- VIR\_TYPED\_PARAM\_BOOLEAN, [152](#)
- VIR\_TYPED\_PARAM\_DOUBLE, [152](#)
- VIR\_TYPED\_PARAM\_INT, [152](#)

- VIR\_TYPED\_PARAM\_LLONG, [152](#)
- VIR\_TYPED\_PARAM\_STRING, [152](#)
- VIR\_TYPED\_PARAM\_STRING\_OKAY, [152](#)
- VIR\_TYPED\_PARAM\_UINT, [152](#)
- VIR\_TYPED\_PARAM\_ULLONG, [152](#)
- VIR\_VCPU\_BLOCKED, [153](#)
- VIR\_VCPU\_OFFLINE, [153](#)
- VIR\_VCPU\_RUNNING, [153](#)
- libvirt.c
  - do\_open, [343](#)
  - initialized, [419](#)
  - MAX\_DRIVERS, [340](#)
  - URI\_ALIAS\_CHARS, [340](#)
  - VIR\_ARG15, [340](#)
  - VIR\_DOMAIN\_DEBUG, [340](#)
  - VIR\_DOMAIN\_DEBUG\_0, [341](#)
  - VIR\_DOMAIN\_DEBUG\_1, [341](#)
  - VIR\_DOMAIN\_DEBUG\_2, [341](#)
  - VIR\_FROM\_THIS, [341](#)
  - VIR\_HAS\_COMMA, [341](#)
  - VIR\_UUID\_DEBUG, [341](#)
  - virConnectAuthCallbackDefault, [343](#)
  - virConnectAuthDefault, [419](#)
  - virConnectAuthPtrDefault, [419](#)
  - virConnectBaselineCPU, [343](#)
  - virConnectClose, [343](#)
  - virConnectCompareCPU, [343](#)
  - virConnectCredTypeDefault, [419](#)
  - virConnectDomainEventDeregister, [343](#)
  - virConnectDomainEventDeregisterAny, [344](#)
  - virConnectDomainEventRegister, [344](#)
  - virConnectDomainEventRegisterAny, [344](#)
  - virConnectDomainXMLFromNative, [344](#)
  - virConnectDomainXMLToNative, [345](#)
  - virConnectFindStoragePoolSources, [345](#)
  - virConnectGetCapabilities, [345](#)
  - virConnectGetConfigFile, [345](#)
  - virConnectGetConfigFilePath, [345](#)
  - virConnectGetDefaultURI, [345](#)
  - virConnectGetHostname, [345](#)
  - virConnectGetLibVersion, [346](#)
  - virConnectGetMaxVcpus, [346](#)
  - virConnectGetSysinfo, [346](#)
  - virConnectGetType, [346](#)
  - virConnectGetURI, [346](#)
  - virConnectGetVersion, [346](#)
  - virConnectIsAlive, [347](#)
  - virConnectIsEncrypted, [347](#)
  - virConnectIsSecure, [347](#)
  - virConnectListAllDomains, [347](#)
  - virConnectListAllInterfaces, [348](#)
  - virConnectListAllNWFilters, [349](#)
  - virConnectListAllNetworks, [348](#)
  - virConnectListAllNodeDevices, [349](#)
  - virConnectListAllSecrets, [349](#)
  - virConnectListAllStoragePools, [350](#)
  - virConnectListDefinedDomains, [350](#)
  - virConnectListDefinedInterfaces, [350](#)
  - virConnectListDefinedNetworks, [351](#)
  - virConnectListDefinedStoragePools, [351](#)
  - virConnectListDomains, [351](#)
  - virConnectListInterfaces, [351](#)
  - virConnectListNWFilters, [352](#)
  - virConnectListNetworks, [352](#)
  - virConnectListSecrets, [352](#)
  - virConnectListStoragePools, [352](#)
  - virConnectNumOfDefinedDomains, [352](#)
  - virConnectNumOfDefinedInterfaces, [352](#)
  - virConnectNumOfDefinedNetworks, [353](#)
  - virConnectNumOfDefinedStoragePools, [353](#)
  - virConnectNumOfDomains, [353](#)
  - virConnectNumOfInterfaces, [353](#)
  - virConnectNumOfNWFilters, [353](#)
  - virConnectNumOfNetworks, [353](#)
  - virConnectNumOfSecrets, [353](#)
  - virConnectNumOfStoragePools, [353](#)
  - virConnectOpen, [354](#)
  - virConnectOpenAuth, [354](#)
  - virConnectOpenFindURIAliasMatch, [354](#)
  - virConnectOpenReadOnly, [354](#)
  - virConnectOpenResolveURIAlias, [354](#)
  - virConnectRef, [354](#)
  - virConnectRegisterCloseCallback, [354](#)
  - virConnectSetKeepAlive, [354](#)
  - virConnectUnregisterCloseCallback, [355](#)
  - virDeviceMonitorTab, [420](#)
  - virDeviceMonitorTabCount, [420](#)
  - virDomainAbortJob, [355](#)
  - virDomainAttachDevice, [355](#)
  - virDomainAttachDeviceFlags, [355](#)
  - virDomainBlockCommit, [355](#)
  - virDomainBlockJobAbort, [356](#)
  - virDomainBlockJobSetSpeed, [357](#)
  - virDomainBlockPeek, [357](#)
  - virDomainBlockPull, [357](#)
  - virDomainBlockRebase, [358](#)
  - virDomainBlockResize, [359](#)
  - virDomainBlockStats, [359](#)
  - virDomainBlockStatsFlags, [359](#)
  - virDomainCoreDump, [360](#)
  - virDomainCreate, [360](#)
  - virDomainCreateLinux, [360](#)
  - virDomainCreateWithFlags, [360](#)
  - virDomainCreateXML, [361](#)
  - virDomainDefineXML, [361](#)
  - virDomainDestroy, [361](#)
  - virDomainDestroyFlags, [361](#)
  - virDomainDetachDevice, [362](#)
  - virDomainDetachDeviceFlags, [362](#)
  - virDomainFree, [362](#)
  - virDomainGetAutostart, [362](#)
  - virDomainGetBkioParameters, [363](#)
  - virDomainGetBlockInfo, [363](#)
  - virDomainGetBlockIoTune, [363](#)
  - virDomainGetBlockJobInfo, [363](#)
  - virDomainGetCPUStats, [364](#)



- virDomainGetConnect, 364
- virDomainGetControllInfo, 364
- virDomainGetDiskErrors, 365
- virDomainGetEmulatorPinInfo, 365
- virDomainGetHostname, 365
- virDomainGetID, 365
- virDomainGetInfo, 366
- virDomainGetInterfaceParameters, 366
- virDomainGetJobInfo, 366
- virDomainGetMaxMemory, 366
- virDomainGetMaxVcpus, 366
- virDomainGetMemoryParameters, 366
- virDomainGetMetadata, 367
- virDomainGetName, 367
- virDomainGetNumaParameters, 367
- virDomainGetOSType, 368
- virDomainGetSchedulerParameters, 368
- virDomainGetSchedulerParametersFlags, 368
- virDomainGetSchedulerType, 368
- virDomainGetSecurityLabel, 368
- virDomainGetSecurityLabelList, 369
- virDomainGetState, 369
- virDomainGetUUID, 369
- virDomainGetUUIDString, 369
- virDomainGetVcpuPinInfo, 369
- virDomainGetVcpus, 369
- virDomainGetVcpusFlags, 370
- virDomainGetXMLDesc, 370
- virDomainHasCurrentSnapshot, 370
- virDomainHasManagedSaveImage, 370
- virDomainInjectNMI, 370
- virDomainInterfaceStats, 371
- virDomainIsActive, 371
- virDomainIsPersistent, 371
- virDomainIsUpdated, 371
- virDomainListAllSnapshots, 371
- virDomainLookupByID, 372
- virDomainLookupByName, 372
- virDomainLookupByUUID, 372
- virDomainLookupByUUIDString, 372
- virDomainManagedSave, 372
- virDomainManagedSaveRemove, 373
- virDomainMemoryPeek, 373
- virDomainMemoryStats, 373
- virDomainMigrate, 373
- virDomainMigrate2, 374
- virDomainMigrateBegin3, 375
- virDomainMigrateConfirm3, 375
- virDomainMigrateDirect, 375
- virDomainMigrateFinish, 375
- virDomainMigrateFinish2, 375
- virDomainMigrateFinish3, 375
- virDomainMigrateGetMaxSpeed, 375
- virDomainMigratePeer2Peer, 376
- virDomainMigratePerform, 376
- virDomainMigratePerform3, 376
- virDomainMigratePrepare, 376
- virDomainMigratePrepare2, 376
- virDomainMigratePrepare3, 376
- virDomainMigratePrepareTunnel, 376
- virDomainMigratePrepareTunnel3, 376
- virDomainMigrateSetMaxDowntime, 376
- virDomainMigrateSetMaxSpeed, 376
- virDomainMigrateToURI, 376
- virDomainMigrateToURI2, 377
- virDomainMigrateVersion1, 378
- virDomainMigrateVersion2, 378
- virDomainMigrateVersion3, 378
- virDomainOpenConsole, 378
- virDomainOpenGraphics, 378
- virDomainPMSuspendForDuration, 380
- virDomainPMWakeup, 380
- virDomainPinEmulator, 379
- virDomainPinVcpu, 379
- virDomainPinVcpuFlags, 379
- virDomainReboot, 380
- virDomainRef, 380
- virDomainReset, 380
- virDomainRestore, 381
- virDomainRestoreFlags, 381
- virDomainResume, 381
- virDomainRevertToSnapshot, 381
- virDomainSave, 382
- virDomainSaveFlags, 382
- virDomainSaveImageDefineXML, 382
- virDomainSaveImageGetXMLDesc, 383
- virDomainScreenshot, 383
- virDomainSendKey, 383
- virDomainSetAutostart, 383
- virDomainSetBlkioParameters, 383
- virDomainSetBlockioTune, 383
- virDomainSetInterfaceParameters, 384
- virDomainSetMaxMemory, 384
- virDomainSetMemory, 384
- virDomainSetMemoryFlags, 384
- virDomainSetMemoryParameters, 385
- virDomainSetMetadata, 385
- virDomainSetNumaParameters, 385
- virDomainSetSchedulerParameters, 385
- virDomainSetSchedulerParametersFlags, 385
- virDomainSetVcpus, 386
- virDomainSetVcpusFlags, 386
- virDomainShutdown, 386
- virDomainShutdownFlags, 387
- virDomainSnapshotCreateXML, 387
- virDomainSnapshotCurrent, 388
- virDomainSnapshotDelete, 388
- virDomainSnapshotFree, 388
- virDomainSnapshotGetConnect, 388
- virDomainSnapshotGetDomain, 388
- virDomainSnapshotGetName, 389
- virDomainSnapshotGetParent, 389
- virDomainSnapshotGetXMLDesc, 389
- virDomainSnapshotHasMetadata, 389
- virDomainSnapshotIsCurrent, 389
- virDomainSnapshotListAllChildren, 389

virDomainSnapshotListChildrenNames, 390  
virDomainSnapshotListNames, 390  
virDomainSnapshotLookupByName, 391  
virDomainSnapshotNum, 391  
virDomainSnapshotNumChildren, 391  
virDomainSnapshotRef, 392  
virDomainSuspend, 392  
virDomainUndefine, 392  
virDomainUndefineFlags, 392  
virDomainUpdateDeviceFlags, 393  
virDriverTab, 420  
virDriverTabCount, 420  
virDrvSupportsFeature, 393  
virGetVersion, 393  
virInitialize, 393  
virInterfaceChangeBegin, 393  
virInterfaceChangeCommit, 394  
virInterfaceChangeRollback, 394  
virInterfaceCreate, 394  
virInterfaceDefineXML, 394  
virInterfaceDestroy, 394  
virInterfaceDriverTab, 420  
virInterfaceDriverTabCount, 420  
virInterfaceFree, 395  
virInterfaceGetConnect, 395  
virInterfaceGetMACString, 395  
virInterfaceGetName, 395  
virInterfaceGetXMLDesc, 395  
virInterfacelsActive, 395  
virInterfaceLookupByMACString, 396  
virInterfaceLookupByName, 396  
virInterfaceRef, 396  
virInterfaceUndefine, 396  
virLibConnError, 341  
virLibDomainError, 341  
virLibDomainSnapshotError, 341  
virLibInterfaceError, 342  
virLibNWFilterError, 342  
virLibNetworkError, 342  
virLibNodeDeviceError, 342  
virLibSecretError, 342  
virLibStoragePoolError, 342  
virLibStorageVolError, 342  
virLibStreamError, 343  
virNWFilterDefineXML, 404  
virNWFilterDriverTab, 420  
virNWFilterDriverTabCount, 420  
virNWFilterFree, 404  
virNWFilterGetName, 404  
virNWFilterGetUUID, 404  
virNWFilterGetUUIDString, 405  
virNWFilterGetXMLDesc, 405  
virNWFilterLookupByName, 405  
virNWFilterLookupByUUID, 405  
virNWFilterLookupByUUIDString, 405  
virNWFilterRef, 405  
virNWFilterUndefine, 405  
virNetworkCreate, 396  
virNetworkCreateXML, 396  
virNetworkDefineXML, 397  
virNetworkDestroy, 397  
virNetworkDriverTab, 420  
virNetworkDriverTabCount, 420  
virNetworkFree, 397  
virNetworkGetAutostart, 397  
virNetworkGetBridgeName, 397  
virNetworkGetBridgeType, 397  
virNetworkGetConnect, 397  
virNetworkGetName, 398  
virNetworkGetUUID, 398  
virNetworkGetUUIDString, 398  
virNetworkGetXMLDesc, 398  
virNetworkIsActive, 398  
virNetworkIsPersistent, 398  
virNetworkLookupByName, 399  
virNetworkLookupByUUID, 399  
virNetworkLookupByUUIDString, 399  
virNetworkRef, 399  
virNetworkSetAutostart, 399  
virNetworkUndefine, 399  
virNetworkUpdate, 399  
virNodeDeviceCreateXML, 399  
virNodeDeviceDestroy, 400  
virNodeDeviceDetach, 400  
virNodeDeviceFree, 400  
virNodeDeviceGetName, 400  
virNodeDeviceGetParent, 400  
virNodeDeviceGetXMLDesc, 400  
virNodeDeviceListCaps, 401  
virNodeDeviceLookupByName, 401  
virNodeDeviceNumOfCaps, 401  
virNodeDeviceReAttach, 401  
virNodeDeviceRef, 401  
virNodeDeviceReset, 401  
virNodeGetCPUStats, 402  
virNodeGetCellsFreeMemory, 401  
virNodeGetFreeMemory, 402  
virNodeGetInfo, 402  
virNodeGetMemoryParameters, 403  
virNodeGetMemoryStats, 403  
virNodeGetSecurityModel, 403  
virNodeListDevices, 403  
virNodeNumOfDevices, 403  
virNodeSetMemoryParameters, 404  
virNodeSuspendForDuration, 404  
virRegisterDeviceMonitor, 405  
virRegisterDriver, 406  
virRegisterInterfaceDriver, 406  
virRegisterNWFilterDriver, 406  
virRegisterNetworkDriver, 406  
virRegisterSecretDriver, 406  
virRegisterStorageDriver, 406  
virSecretDefineXML, 406  
virSecretDriverTab, 420  
virSecretDriverTabCount, 420  
virSecretFree, 407

- virSecretGetConnect, [407](#)
- virSecretGetUUID, [407](#)
- virSecretGetUUIDString, [407](#)
- virSecretGetUsageID, [407](#)
- virSecretGetUsageType, [407](#)
- virSecretGetValue, [408](#)
- virSecretGetXMLDesc, [408](#)
- virSecretLookupByUUID, [408](#)
- virSecretLookupByUUIDString, [408](#)
- virSecretLookupByUsage, [408](#)
- virSecretRef, [408](#)
- virSecretSetValue, [409](#)
- virSecretUndefine, [409](#)
- virStorageDriverTab, [420](#)
- virStorageDriverTabCount, [420](#)
- virStoragePoolBuild, [409](#)
- virStoragePoolCreate, [409](#)
- virStoragePoolCreateXML, [409](#)
- virStoragePoolDefineXML, [409](#)
- virStoragePoolDelete, [409](#)
- virStoragePoolDestroy, [410](#)
- virStoragePoolFree, [410](#)
- virStoragePoolGetAutostart, [410](#)
- virStoragePoolGetConnect, [410](#)
- virStoragePoolGetInfo, [410](#)
- virStoragePoolGetName, [410](#)
- virStoragePoolGetUUID, [410](#)
- virStoragePoolGetUUIDString, [411](#)
- virStoragePoolGetXMLDesc, [411](#)
- virStoragePoolsActive, [411](#)
- virStoragePoolsPersistent, [411](#)
- virStoragePoolListAllVolumes, [411](#)
- virStoragePoolListVolumes, [411](#)
- virStoragePoolLookupByName, [412](#)
- virStoragePoolLookupByUUID, [412](#)
- virStoragePoolLookupByUUIDString, [412](#)
- virStoragePoolLookupByVolume, [412](#)
- virStoragePoolNumOfVolumes, [412](#)
- virStoragePoolRef, [412](#)
- virStoragePoolRefresh, [412](#)
- virStoragePoolSetAutostart, [412](#)
- virStoragePoolUndefine, [413](#)
- virStorageVolCreateXML, [413](#)
- virStorageVolCreateXMLFrom, [413](#)
- virStorageVolDelete, [413](#)
- virStorageVolDownload, [413](#)
- virStorageVolFree, [413](#)
- virStorageVolGetConnect, [414](#)
- virStorageVolGetInfo, [414](#)
- virStorageVolGetKey, [414](#)
- virStorageVolGetName, [414](#)
- virStorageVolGetPath, [414](#)
- virStorageVolGetXMLDesc, [414](#)
- virStorageVolLookupByKey, [414](#)
- virStorageVolLookupByName, [415](#)
- virStorageVolLookupByPath, [415](#)
- virStorageVolRef, [415](#)
- virStorageVolResize, [415](#)
- virStorageVolUpload, [415](#)
- virStorageVolWipe, [416](#)
- virStorageVolWipePattern, [416](#)
- virStreamAbort, [416](#)
- virStreamEventAddCallback, [416](#)
- virStreamEventRemoveCallback, [416](#)
- virStreamEventUpdateCallback, [416](#)
- virStreamFinish, [416](#)
- virStreamFree, [417](#)
- virStreamNew, [417](#)
- virStreamRecv, [417](#)
- virStreamRecvAll, [417](#)
- virStreamRef, [418](#)
- virStreamSend, [418](#)
- virStreamSendAll, [418](#)
- virTLSMutexDestroy, [419](#)
- virTLSMutexInit, [419](#)
- virTLSMutexLock, [419](#)
- virTLSMutexUnlock, [419](#)
- virTLSThreadImpl, [420](#)
- virTypedParameterValidateSet, [419](#)
- libvirt.h
  - \_virBlkioParameter, [111](#)
  - \_virMemoryParameter, [111](#)
  - \_virSchedParameter, [111](#)
  - VIR\_COPY\_CPUMAP, [111](#)
  - VIR\_CPU\_MAPLEN, [111](#)
  - VIR\_CPU\_USABLE, [111](#)
  - VIR\_DEPRECATED, [112](#)
  - VIR\_EXPORT\_VAR, [117](#)
  - VIR\_GET\_CPUMAP, [117](#)
  - VIR\_UNUSE\_CPU, [120](#)
  - VIR\_USE\_CPU, [120](#)
  - VIR\_UUID\_BUFLen, [120](#)
  - virBlkioParameter, [120](#)
  - virBlkioParameterPtr, [120](#)
  - virBlkioParameterType, [132](#)
  - virCPUCompareResult, [135](#)
  - virConnect, [120](#)
  - virConnectAuth, [120](#)
  - virConnectAuthCallbackPtr, [121](#)
  - virConnectAuthPtr, [121](#)
  - virConnectAuthPtrDefault, [227](#)
  - virConnectBaselineCPU, [153](#)
  - virConnectClose, [153](#)
  - virConnectCloseFunc, [121](#)
  - virConnectCloseReason, [132](#)
  - virConnectCompareCPU, [153](#)
  - virConnectCredential, [121](#)
  - virConnectCredentialPtr, [121](#)
  - virConnectCredentialType, [132](#)
  - virConnectDomainEventBalloonChangeCallback, [121](#)
  - virConnectDomainEventBlockJobCallback, [121](#)
  - virConnectDomainEventBlockJobStatus, [132](#)
  - virConnectDomainEventCallback, [121](#)
  - virConnectDomainEventDeregister, [154](#)
  - virConnectDomainEventDeregisterAny, [154](#)



- virConnectDomainEventDiskChangeCallback, 121
- virConnectDomainEventDiskChangeReason, 132
- virConnectDomainEventGenericCallback, 122
- virConnectDomainEventGraphicsCallback, 122
- virConnectDomainEventIOErrorCallback, 122
- virConnectDomainEventIOErrorReasonCallback, 122
- virConnectDomainEventPMSuspendCallback, 122
- virConnectDomainEventPMWakeupCallback, 122
- virConnectDomainEventRTCChangeCallback, 122
- virConnectDomainEventRegister, 154
- virConnectDomainEventRegisterAny, 154
- virConnectDomainEventTrayChangeCallback, 123
- virConnectDomainEventWatchdogCallback, 123
- virConnectDomainXMLFromNative, 155
- virConnectDomainXMLToNative, 155
- virConnectFindStoragePoolSources, 155
- virConnectFlags, 133
- virConnectGetCapabilities, 155
- virConnectGetHostname, 155
- virConnectGetLibVersion, 156
- virConnectGetMaxVcpus, 156
- virConnectGetSysinfo, 156
- virConnectGetType, 156
- virConnectGetURI, 156
- virConnectGetVersion, 156
- virConnectIsAlive, 157
- virConnectIsEncrypted, 157
- virConnectIsSecure, 157
- virConnectListAllDomains, 157
- virConnectListAllDomainsFlags, 133
- virConnectListAllInterfaces, 158
- virConnectListAllInterfacesFlags, 133
- virConnectListAllNWFilters, 159
- virConnectListAllNetworks, 158
- virConnectListAllNetworksFlags, 133
- virConnectListAllNodeDeviceFlags, 134
- virConnectListAllNodeDevices, 159
- virConnectListAllSecrets, 159
- virConnectListAllSecretsFlags, 134
- virConnectListAllStoragePools, 160
- virConnectListAllStoragePoolsFlags, 134
- virConnectListDefinedDomains, 160
- virConnectListDefinedInterfaces, 160
- virConnectListDefinedNetworks, 161
- virConnectListDefinedStoragePools, 161
- virConnectListDomains, 161
- virConnectListInterfaces, 161
- virConnectListNWFilters, 162
- virConnectListNetworks, 161
- virConnectListSecrets, 162
- virConnectListStoragePools, 162
- virConnectNumOfDefinedDomains, 162
- virConnectNumOfDefinedInterfaces, 162
- virConnectNumOfDefinedNetworks, 162
- virConnectNumOfDefinedStoragePools, 163
- virConnectNumOfDomains, 163
- virConnectNumOfInterfaces, 163
- virConnectNumOfNWFilters, 163
- virConnectNumOfNetworks, 163
- virConnectNumOfSecrets, 163
- virConnectNumOfStoragePools, 163
- virConnectOpen, 164
- virConnectOpenAuth, 164
- virConnectOpenReadOnly, 164
- virConnectPtr, 123
- virConnectRef, 164
- virConnectRegisterCloseCallback, 164
- virConnectSetKeepAlive, 164
- virConnectUnregisterCloseCallback, 164
- virDomain, 123
- virDomainAbortJob, 165
- virDomainAttachDevice, 165
- virDomainAttachDeviceFlags, 165
- virDomainBlockCommit, 165
- virDomainBlockCommitFlags, 135
- virDomainBlockInfo, 123
- virDomainBlockInfoPtr, 124
- virDomainBlockJobAbort, 166
- virDomainBlockJobAbortFlags, 135
- virDomainBlockJobCursor, 124
- virDomainBlockJobInfo, 124
- virDomainBlockJobInfoPtr, 124
- virDomainBlockJobSetSpeed, 166
- virDomainBlockJobType, 135
- virDomainBlockPeek, 167
- virDomainBlockPull, 167
- virDomainBlockRebase, 168
- virDomainBlockRebaseFlags, 136
- virDomainBlockResize, 168
- virDomainBlockResizeFlags, 136
- virDomainBlockStats, 169
- virDomainBlockStatsFlags, 169
- virDomainBlockStatsPtr, 124
- virDomainBlockStatsStruct, 124
- virDomainBlockedReason, 135
- virDomainConsoleFlags, 136
- virDomainControlInfo, 124
- virDomainControlInfoPtr, 124
- virDomainControlState, 136
- virDomainCoreDump, 169
- virDomainCoreDumpFlags, 136
- virDomainCrashedReason, 137
- virDomainCreate, 170
- virDomainCreateFlags, 137
- virDomainCreateLinux, 170
- virDomainCreateWithFlags, 170
- virDomainCreateXML, 171
- virDomainDefineXML, 171
- virDomainDestroy, 171
- virDomainDestroyFlags, 171
- virDomainDestroyFlagsValues, 137
- virDomainDetachDevice, 172
- virDomainDetachDeviceFlags, 172
- virDomainDeviceModifyFlags, 137
- virDomainDiskError, 124

- virDomainDiskErrorCode, 137
- virDomainDiskErrorPtr, 124
- virDomainEventDefinedDetailType, 138
- virDomainEventGraphicsAddress, 125
- virDomainEventGraphicsAddressPtr, 125
- virDomainEventGraphicsAddressType, 138
- virDomainEventGraphicsPhase, 138
- virDomainEventGraphicsSubject, 125
- virDomainEventGraphicsSubjectIdentity, 125
- virDomainEventGraphicsSubjectIdentityPtr, 125
- virDomainEventGraphicsSubjectPtr, 125
- virDomainEventID, 138
- virDomainEventIOErrorAction, 139
- virDomainEventPMSuspendedDetailType, 139
- virDomainEventResumedDetailType, 139
- virDomainEventShutdownDetailType, 139
- virDomainEventStartedDetailType, 139
- virDomainEventStoppedDetailType, 140
- virDomainEventSuspendedDetailType, 140
- virDomainEventTrayChangeReason, 140
- virDomainEventType, 140
- virDomainEventUndefinedDetailType, 141
- virDomainEventWatchdogAction, 141
- virDomainFree, 172
- virDomainGetAutostart, 172
- virDomainGetBlkioParameters, 172
- virDomainGetBlockInfo, 173
- virDomainGetBlockIoTune, 173
- virDomainGetBlockJobInfo, 173
- virDomainGetCPUStats, 174
- virDomainGetConnect, 174
- virDomainGetControllInfo, 174
- virDomainGetDiskErrors, 175
- virDomainGetEmulatorPinInfo, 175
- virDomainGetHostname, 175
- virDomainGetID, 175
- virDomainGetInfo, 175
- virDomainGetInterfaceParameters, 176
- virDomainGetJobInfo, 176
- virDomainGetMaxMemory, 176
- virDomainGetMaxVcpus, 176
- virDomainGetMemoryParameters, 176
- virDomainGetMetadata, 177
- virDomainGetName, 177
- virDomainGetNumaParameters, 177
- virDomainGetOSType, 177
- virDomainGetSchedulerParameters, 177
- virDomainGetSchedulerParametersFlags, 178
- virDomainGetSchedulerType, 178
- virDomainGetSecurityLabel, 178
- virDomainGetSecurityLabelList, 178
- virDomainGetState, 178
- virDomainGetUUID, 179
- virDomainGetUUIDString, 179
- virDomainGetVcpuPinInfo, 179
- virDomainGetVcpus, 179
- virDomainGetVcpusFlags, 179
- virDomainGetXMLEDesc, 180
- virDomainHasCurrentSnapshot, 180
- virDomainHasManagedSaveImage, 180
- virDomainInfo, 125
- virDomainInfoPtr, 125
- virDomainInjectNMI, 180
- virDomainInterfaceStats, 180
- virDomainInterfaceStatsPtr, 125
- virDomainInterfaceStatsStruct, 125
- virDomainIsActive, 181
- virDomainIsPersistent, 181
- virDomainIsUpdated, 181
- virDomainJobInfo, 125
- virDomainJobInfoPtr, 125
- virDomainJobType, 141
- virDomainListAllSnapshots, 181
- virDomainLookupByID, 182
- virDomainLookupByName, 182
- virDomainLookupByUUID, 182
- virDomainLookupByUUIDString, 182
- virDomainManagedSave, 182
- virDomainManagedSaveRemove, 182
- virDomainMemoryFlags, 141
- virDomainMemoryModFlags, 142
- virDomainMemoryPeek, 183
- virDomainMemoryStatPtr, 125
- virDomainMemoryStatStruct, 125
- virDomainMemoryStatTags, 142
- virDomainMemoryStats, 183
- virDomainMetadataType, 142
- virDomainMigrate, 183
- virDomainMigrate2, 184
- virDomainMigrateFlags, 142
- virDomainMigrateGetMaxSpeed, 185
- virDomainMigrateSetMaxDowntime, 185
- virDomainMigrateSetMaxSpeed, 185
- virDomainMigrateToURI, 186
- virDomainMigrateToURI2, 186
- virDomainModificationImpact, 143
- virDomainNostateReason, 143
- virDomainNumatuneMemMode, 143
- virDomainOpenConsole, 187
- virDomainOpenGraphics, 187
- virDomainOpenGraphicsFlags, 143
- virDomainPMSuspendForDuration, 188
- virDomainPMSuspendedReason, 144
- virDomainPMWakeup, 189
- virDomainPausedReason, 143
- virDomainPinEmulator, 188
- virDomainPinVcpu, 188
- virDomainPinVcpuFlags, 188
- virDomainPtr, 125
- virDomainReboot, 189
- virDomainRebootFlagValues, 144
- virDomainRef, 189
- virDomainReset, 189
- virDomainRestore, 189
- virDomainRestoreFlags, 190
- virDomainResume, 190

- [virDomainRevertToSnapshot, 190](#)
- [virDomainRunningReason, 144](#)
- [virDomainSave, 191](#)
- [virDomainSaveFlags, 191](#)
- [virDomainSaveImageDefineXML, 191](#)
- [virDomainSaveImageGetXMLDesc, 191](#)
- [virDomainSaveRestoreFlags, 144](#)
- [virDomainScreenshot, 191](#)
- [virDomainSendKey, 192](#)
- [virDomainSetAutostart, 192](#)
- [virDomainSetBlkioParameters, 192](#)
- [virDomainSetBlockioTune, 192](#)
- [virDomainSetInterfaceParameters, 192](#)
- [virDomainSetMaxMemory, 193](#)
- [virDomainSetMemory, 193](#)
- [virDomainSetMemoryFlags, 193](#)
- [virDomainSetMemoryParameters, 193](#)
- [virDomainSetMetadata, 194](#)
- [virDomainSetNumaParameters, 194](#)
- [virDomainSetSchedulerParameters, 194](#)
- [virDomainSetSchedulerParametersFlags, 194](#)
- [virDomainSetVcpus, 194](#)
- [virDomainSetVcpusFlags, 195](#)
- [virDomainShutdown, 195](#)
- [virDomainShutdownFlagValues, 144](#)
- [virDomainShutdownFlags, 195](#)
- [virDomainShutdownReason, 145](#)
- [virDomainShutoffReason, 145](#)
- [virDomainSnapshot, 125](#)
- [virDomainSnapshotCreateFlags, 145](#)
- [virDomainSnapshotCreateXML, 196](#)
- [virDomainSnapshotCurrent, 197](#)
- [virDomainSnapshotDelete, 197](#)
- [virDomainSnapshotDeleteFlags, 145](#)
- [virDomainSnapshotFree, 197](#)
- [virDomainSnapshotGetConnect, 197](#)
- [virDomainSnapshotGetDomain, 197](#)
- [virDomainSnapshotGetName, 197](#)
- [virDomainSnapshotGetParent, 198](#)
- [virDomainSnapshotGetXMLDesc, 198](#)
- [virDomainSnapshotHasMetadata, 198](#)
- [virDomainSnapshotIsCurrent, 198](#)
- [virDomainSnapshotListAllChildren, 198](#)
- [virDomainSnapshotListChildrenNames, 199](#)
- [virDomainSnapshotListFlags, 145](#)
- [virDomainSnapshotListNames, 199](#)
- [virDomainSnapshotLookupByName, 200](#)
- [virDomainSnapshotNum, 200](#)
- [virDomainSnapshotNumChildren, 200](#)
- [virDomainSnapshotPtr, 126](#)
- [virDomainSnapshotRef, 201](#)
- [virDomainSnapshotRevertFlags, 146](#)
- [virDomainState, 146](#)
- [virDomainSuspend, 201](#)
- [virDomainUndefine, 201](#)
- [virDomainUndefineFlags, 201](#)
- [virDomainUndefineFlagsValues, 146](#)
- [virDomainUpdateDeviceFlags, 201](#)
- [virDomainVcpuFlags, 146](#)
- [virDomainXMLFlags, 147](#)
- [virEventAddHandle, 202](#)
- [virEventAddHandleFunc, 126](#)
- [virEventAddTimeout, 202](#)
- [virEventAddTimeoutFunc, 126](#)
- [virEventHandleCallback, 126](#)
- [virEventHandleType, 147](#)
- [virEventRegisterDefaultImpl, 202](#)
- [virEventRegisterImpl, 202](#)
- [virEventRemoveHandle, 202](#)
- [virEventRemoveHandleFunc, 126](#)
- [virEventRemoveTimeout, 202](#)
- [virEventRemoveTimeoutFunc, 126](#)
- [virEventRunDefaultImpl, 202](#)
- [virEventTimeoutCallback, 127](#)
- [virEventUpdateHandle, 202](#)
- [virEventUpdateHandleFunc, 127](#)
- [virEventUpdateTimeout, 202](#)
- [virEventUpdateTimeoutFunc, 127](#)
- [virFreeCallback, 127](#)
- [virGetVersion, 202](#)
- [virInitialize, 202](#)
- [virInterface, 127](#)
- [virInterfaceChangeBegin, 202](#)
- [virInterfaceChangeCommit, 203](#)
- [virInterfaceChangeRollback, 203](#)
- [virInterfaceCreate, 203](#)
- [virInterfaceDefineXML, 203](#)
- [virInterfaceDestroy, 203](#)
- [virInterfaceFree, 204](#)
- [virInterfaceGetConnect, 204](#)
- [virInterfaceGetMACString, 204](#)
- [virInterfaceGetName, 204](#)
- [virInterfaceGetXMLDesc, 204](#)
- [virInterfaceIsActive, 204](#)
- [virInterfaceLookupByMACString, 205](#)
- [virInterfaceLookupByName, 205](#)
- [virInterfacePtr, 127](#)
- [virInterfaceRef, 205](#)
- [virInterfaceUndefine, 205](#)
- [virInterfaceXMLFlags, 147](#)
- [virKeyCodeSet, 147](#)
- [virMemoryParameter, 127](#)
- [virMemoryParameterPtr, 127](#)
- [virMemoryParameterType, 148](#)
- [virNWFilter, 129](#)
- [virNWFilterDefineXML, 213](#)
- [virNWFilterFree, 213](#)
- [virNWFilterGetName, 213](#)
- [virNWFilterGetUUID, 213](#)
- [virNWFilterGetUUIDString, 214](#)
- [virNWFilterGetXMLDesc, 214](#)
- [virNWFilterLookupByName, 214](#)
- [virNWFilterLookupByUUID, 214](#)
- [virNWFilterLookupByUUIDString, 214](#)
- [virNWFilterPtr, 129](#)
- [virNWFilterRef, 214](#)

[virNWFilterUndefine](#), 214  
[virNetwork](#), 127  
[virNetworkCreate](#), 205  
[virNetworkCreateXML](#), 205  
[virNetworkDefineXML](#), 206  
[virNetworkDestroy](#), 206  
[virNetworkFree](#), 206  
[virNetworkGetAutostart](#), 206  
[virNetworkGetBridgeName](#), 206  
[virNetworkGetBridgeType](#), 206  
[virNetworkGetConnect](#), 206  
[virNetworkGetName](#), 207  
[virNetworkGetUUID](#), 207  
[virNetworkGetUUIDString](#), 207  
[virNetworkGetXMLDesc](#), 207  
[virNetworkIsActive](#), 207  
[virNetworkIsPersistent](#), 207  
[virNetworkLookupByName](#), 208  
[virNetworkLookupByUUID](#), 208  
[virNetworkLookupByUUIDString](#), 208  
[virNetworkPtr](#), 128  
[virNetworkRef](#), 208  
[virNetworkSetAutostart](#), 208  
[virNetworkUndefine](#), 208  
[virNetworkUpdate](#), 208  
[virNetworkUpdateCommand](#), 148  
[virNetworkUpdateFlags](#), 148  
[virNetworkUpdateSection](#), 148  
[virNetworkXMLFlags](#), 149  
[virNodeCPUStats](#), 128  
[virNodeCPUStatsPtr](#), 128  
[virNodeDevice](#), 128  
[virNodeDeviceCreateXML](#), 208  
[virNodeDeviceDestroy](#), 209  
[virNodeDeviceDetach](#), 209  
[virNodeDeviceFree](#), 209  
[virNodeDeviceGetName](#), 209  
[virNodeDeviceGetParent](#), 209  
[virNodeDeviceGetXMLDesc](#), 209  
[virNodeDeviceListCaps](#), 210  
[virNodeDeviceLookupByName](#), 210  
[virNodeDeviceNumOfCaps](#), 210  
[virNodeDevicePtr](#), 128  
[virNodeDeviceReAttach](#), 210  
[virNodeDeviceRef](#), 210  
[virNodeDeviceReset](#), 210  
[virNodeGetCPUStats](#), 211  
[virNodeGetCPUStatsAllCPUs](#), 149  
[virNodeGetCellsFreeMemory](#), 210  
[virNodeGetFreeMemory](#), 211  
[virNodeGetInfo](#), 211  
[virNodeGetMemoryParameters](#), 212  
[virNodeGetMemoryStats](#), 212  
[virNodeGetMemoryStatsAllCells](#), 149  
[virNodeGetSecurityModel](#), 212  
[virNodeInfo](#), 128  
[virNodeInfoPtr](#), 128  
[virNodeListDevices](#), 212  
[virNodeMemoryStats](#), 128  
[virNodeMemoryStatsPtr](#), 128  
[virNodeNumOfDevices](#), 212  
[virNodeSetMemoryParameters](#), 213  
[virNodeSuspendForDuration](#), 213  
[virNodeSuspendTarget](#), 149  
[virSchedParameter](#), 129  
[virSchedParameterPtr](#), 129  
[virSchedParameterType](#), 150  
[virSecret](#), 129  
[virSecretDefineXML](#), 214  
[virSecretFree](#), 215  
[virSecretGetConnect](#), 215  
[virSecretGetUUID](#), 215  
[virSecretGetUUIDString](#), 215  
[virSecretGetUsageID](#), 215  
[virSecretGetUsageType](#), 215  
[virSecretGetValue](#), 216  
[virSecretGetXMLDesc](#), 216  
[virSecretLookupByUUID](#), 216  
[virSecretLookupByUUIDString](#), 216  
[virSecretLookupByUsage](#), 216  
[virSecretPtr](#), 129  
[virSecretRef](#), 216  
[virSecretSetValue](#), 217  
[virSecretUndefine](#), 217  
[virSecretUsageType](#), 150  
[virSecurityLabel](#), 129  
[virSecurityLabelPtr](#), 129  
[virSecurityModel](#), 129  
[virSecurityModelPtr](#), 129  
[virStoragePool](#), 130  
[virStoragePoolBuild](#), 217  
[virStoragePoolBuildFlags](#), 150  
[virStoragePoolCreate](#), 217  
[virStoragePoolCreateXML](#), 217  
[virStoragePoolDefineXML](#), 217  
[virStoragePoolDelete](#), 218  
[virStoragePoolDeleteFlags](#), 150  
[virStoragePoolDestroy](#), 218  
[virStoragePoolFree](#), 218  
[virStoragePoolGetAutostart](#), 218  
[virStoragePoolGetConnect](#), 218  
[virStoragePoolGetInfo](#), 218  
[virStoragePoolGetName](#), 218  
[virStoragePoolGetUUID](#), 219  
[virStoragePoolGetUUIDString](#), 219  
[virStoragePoolGetXMLDesc](#), 219  
[virStoragePoolInfo](#), 130  
[virStoragePoolInfoPtr](#), 130  
[virStoragePoolsActive](#), 219  
[virStoragePoolsPersistent](#), 219  
[virStoragePoolListAllVolumes](#), 219  
[virStoragePoolListVolumes](#), 219  
[virStoragePoolLookupByName](#), 220  
[virStoragePoolLookupByUUID](#), 220  
[virStoragePoolLookupByUUIDString](#), 220  
[virStoragePoolLookupByVolume](#), 220

- virStoragePoolNumOfVolumes, [220](#)
- virStoragePoolPtr, [130](#)
- virStoragePoolRef, [220](#)
- virStoragePoolRefresh, [220](#)
- virStoragePoolSetAutostart, [221](#)
- virStoragePoolState, [150](#)
- virStoragePoolUndefine, [221](#)
- virStorageVol, [130](#)
- virStorageVolCreateXML, [221](#)
- virStorageVolCreateXMLFrom, [221](#)
- virStorageVolDelete, [221](#)
- virStorageVolDeleteFlags, [151](#)
- virStorageVolDownload, [221](#)
- virStorageVolFree, [222](#)
- virStorageVolGetConnect, [222](#)
- virStorageVolGetInfo, [222](#)
- virStorageVolGetKey, [222](#)
- virStorageVolGetName, [222](#)
- virStorageVolGetPath, [222](#)
- virStorageVolGetXMLDesc, [222](#)
- virStorageVolInfo, [130](#)
- virStorageVolInfoPtr, [130](#)
- virStorageVolLookupByKey, [223](#)
- virStorageVolLookupByName, [223](#)
- virStorageVolLookupByPath, [223](#)
- virStorageVolPtr, [130](#)
- virStorageVolRef, [223](#)
- virStorageVolResize, [223](#)
- virStorageVolResizeFlags, [151](#)
- virStorageVolType, [151](#)
- virStorageVolUpload, [223](#)
- virStorageVolWipe, [224](#)
- virStorageVolWipeAlgorithm, [151](#)
- virStorageVolWipePattern, [224](#)
- virStorageXMLFlags, [151](#)
- virStream, [130](#)
- virStreamAbort, [224](#)
- virStreamEventAddCallback, [224](#)
- virStreamEventCallback, [130](#)
- virStreamEventRemoveCallback, [224](#)
- virStreamEventType, [152](#)
- virStreamEventUpdateCallback, [224](#)
- virStreamFinish, [225](#)
- virStreamFlags, [152](#)
- virStreamFree, [225](#)
- virStreamNew, [225](#)
- virStreamPtr, [130](#)
- virStreamRecv, [225](#)
- virStreamRecvAll, [226](#)
- virStreamRef, [226](#)
- virStreamSend, [226](#)
- virStreamSendAll, [227](#)
- virStreamSinkFunc, [131](#)
- virStreamSourceFunc, [131](#)
- virTypedParameter, [131](#)
- virTypedParameterFlags, [152](#)
- virTypedParameterPtr, [131](#)
- virTypedParameterType, [152](#)
- virVcpuInfo, [131](#)
- virVcpuInfoPtr, [131](#)
- virVcpuState, [152](#)
- libvirtVersion
  - \_virDriver, [65](#)
- limit
  - \_virDomainTimerCatchupDef, [53](#)
- linkdev
  - \_virDomainActualNetDef, [11](#)
  - \_virDomainNetDef, [45](#)
- linkstate
  - \_virDomainNetDef, [45](#)
- listAllDomains
  - \_virDriver, [65](#)
- listAllInterfaces
  - \_virInterfaceDriver, [67](#)
- listAllNWFilters
  - \_virNWFilterDriver, [79](#)
- listAllNetworks
  - \_virNetworkDriver, [73](#)
- listAllNodeDevices
  - \_virDeviceMonitor, [9](#)
- listAllPools
  - \_virStorageDriver, [84](#)
- listAllSecrets
  - \_virSecretDriver, [81](#)
- listDefinedDomains
  - \_virDriver, [65](#)
- listDefinedInterfaces
  - \_virInterfaceDriver, [67](#)
- listDefinedNetworks
  - \_virNetworkDriver, [73](#)
- listDefinedPools
  - \_virStorageDriver, [84](#)
- listDevices
  - \_virDeviceMonitor, [9](#)
- listDomains
  - \_virDriver, [65](#)
- listInterfaces
  - \_virInterfaceDriver, [67](#)
- listNWFilters
  - \_virNWFilterDriver, [79](#)
- listNetworks
  - \_virNetworkDriver, [73](#)
- listPools
  - \_virStorageDriver, [84](#)
- listSecrets
  - \_virSecretDriver, [81](#)
- listen
  - \_virDomainChrSourceDef, [15](#)
- listens
  - \_virDomainGraphicsDef, [36](#)
- loader
  - \_virDomainOSDef, [49](#)
- lock
  - \_virDomainObj, [47](#)
  - \_virNetworkObj, [76](#)
  - network\_driver, [90](#)

- lockspace
  - \_virDomainLeaseDef, 42
- logDir
  - network\_driver, 90
- logical\_block\_size
  - \_virDomainDiskDef, 29
- lookupByUUID
  - \_virSecretDriver, 81
- lookupByUsage
  - \_virSecretDriver, 81
- MATCH
  - domain\_conf.c, 235
  - network\_conf.c, 295
- MAX\_BRIDGE\_ID
  - network\_conf.c, 295
- MAX\_DRIVERS
  - libvirt.c, 340
- mac
  - \_virDomainNetDef, 45
  - \_virNetworkDHCPHostDef, 69
  - \_virNetworkDef, 68
- mac\_specified
  - \_virNetworkDef, 68
- macaddr
  - virnetdevopenvswitch.h, 445
- machine
  - \_virDomainOSDef, 49
- managed
  - \_virDomainHostdevDef, 37
  - \_virNetworkDef, 68
- master
  - \_virDomainDeviceInfo, 25
- mastertype
  - \_virDomainDeviceInfo, 25
- max\_balloon
  - \_virDomainDef, 20
- maxMem
  - \_virDomainInfo, 40
- maxids
  - virDomainIDDData, 90
- maxnames
  - virDomainNameData, 91
- maxvcpus
  - \_virDomainDef, 20
- mem
  - \_virDomainDef, 20
- memProcessed
  - \_virDomainJobInfo, 42
- memRemaining
  - \_virDomainJobInfo, 42
- memTotal
  - \_virDomainJobInfo, 42
- memballoon
  - \_virDomainDef, 20
  - \_virDomainDeviceDef, 24
- memory
  - \_virDomainInfo, 40
  - \_virDomainNumatuneDef, 46
- \_virNodeInfo, 77
- metadata
  - \_virDomainDef, 20
- mhz
  - \_virNodeInfo, 77
- min\_guarantee
  - \_virDomainDef, 20
- mirror
  - \_virDomainDiskDef, 29
- mirrorFormat
  - \_virDomainDiskDef, 29
- mirroring
  - \_virDomainDiskDef, 29
- mode
  - \_virDomainActualNetDef, 11
  - \_virDomainHostdevDef, 37
  - \_virDomainNetDef, 45
  - \_virDomainNumatuneDef, 46
  - \_virDomainTimerDef, 53
- model
  - \_virDomainControllerDef, 17
  - \_virDomainMemballoonDef, 43
  - \_virDomainNetDef, 45
  - \_virDomainSoundDef, 52
  - \_virDomainWatchdogDef, 55
  - \_virNodeInfo, 77
  - \_virSecurityDeviceLabelDef, 81
  - \_virSecurityLabelDef, 82
  - \_virSecurityModel, 83
- mousemode
  - \_virDomainGraphicsDef, 36
- multiUser
  - \_virDomainGraphicsDef, 36
- nBootDevs
  - \_virDomainOSDef, 49
- NET\_MODEL\_CHARS
  - domain\_conf.c, 236
- NETWORK\_PID\_DIR
  - bridge\_driver.c, 424
- NETWORK\_STATE\_DIR
  - bridge\_driver.c, 424
- nForwardIfs
  - \_virNetworkDef, 68
- nForwardPfs
  - \_virNetworkDef, 68
- nListens
  - \_virDomainGraphicsDef, 36
- nPortGroups
  - \_virNetworkDef, 68
- nTunnels
  - \_virNetworkDef, 69
- name
  - \_virDeviceMonitor, 10
  - \_virDomainChrDef, 14
  - \_virDomainDef, 21
  - \_virDomainDiskHostDef, 31
  - \_virDomainEventGraphicsSubjectIdentity, 33
  - \_virDomainNetDef, 45



- [\\_virDomainTimerDef, 53](#)
- [\\_virDriver, 65](#)
- [\\_virInterfaceDriver, 67](#)
- [\\_virNWFilterDriver, 79](#)
- [\\_virNetworkDHCPHostDef, 69](#)
- [\\_virNetworkDNSTxtRecordsDef, 72](#)
- [\\_virNetworkDef, 68](#)
- [\\_virNetworkDriver, 73](#)
- [\\_virPortGroupDef, 80](#)
- [\\_virSecretDriver, 81](#)
- [\\_virStorageDriver, 84](#)
- names
  - [\\_virNetworkDNSHostsDef, 71](#)
  - [virDomainNameData, 91](#)
- namespaceData
  - [\\_virDomainDef, 21](#)
- nchannels
  - [\\_virDomainDef, 21](#)
- ncodecs
  - [\\_virDomainSoundDef, 52](#)
- nconsoles
  - [\\_virDomainDef, 21](#)
- ncontrollers
  - [\\_virDomainDef, 21](#)
- ncredtype
  - [\\_virConnectAuth, 8](#)
- ndevices
  - [\\_virDomainDef, 21](#)
- ndisks
  - [\\_virDomainDef, 21](#)
- ndomains
  - [virDomainListData, 91](#)
- net
  - [\\_virDomainDeviceDef, 24](#)
- netmask
  - [\\_virNetworkIpfDef, 75](#)
- nets
  - [\\_virDomainDef, 21](#)
  - [vshNetworkList, 91](#)
- network
  - [\\_virDomainGraphicsListenDef, 37](#)
  - [\\_virDomainNetDef, 45](#)
- network\_conf.h
  - [VIR\\_NETWORK\\_FORWARD\\_BRIDGE, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_HOSTDEV, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_HOSTDEV\\_DEVICE\\_LAST, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_HOSTDEV\\_DEVICE\\_NETDEV, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_HOSTDEV\\_DEVICE\\_NONE, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_HOSTDEV\\_DEVICE\\_PCI, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_LAST, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_NAT, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_NONE, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_HOSTDEV\\_DEVICE\\_PCI, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_LAST, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_NAT, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_NONE, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_PASSTHROUGH, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_PRIVATE, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_ROUTE, 305](#)
  - [VIR\\_NETWORK\\_FORWARD\\_VEPA, 305](#)
- network\_conf.c
  - [MATCH, 295](#)
  - [MAX\\_BRIDGE\\_ID, 295](#)
  - [VIR\\_ENUM\\_IMPL, 295](#)
  - [VIR\\_FROM\\_THIS, 295](#)
  - [virNetworkAllocateBridge, 295](#)
  - [virNetworkAssignDef, 295](#)
  - [virNetworkBridgeInUse, 295](#)
  - [virNetworkConfigChangeSetup, 295](#)
  - [virNetworkConfigFile, 296](#)
  - [virNetworkDHCPDefParse, 298](#)
  - [virNetworkDHCPHostDefClear, 298](#)
  - [virNetworkDHCPHostDefParse, 298](#)
  - [virNetworkDHCPRangeDefParse, 298](#)
  - [virNetworkDNSDefFormat, 298](#)
  - [virNetworkDNSDefFree, 298](#)
  - [virNetworkDNSDefParseXML, 298](#)
  - [virNetworkDNSHostsDefParseXML, 298](#)
  - [virNetworkDNSSrvDefParseXML, 298](#)
  - [virNetworkDefCopy, 296](#)
  - [virNetworkDefFormat, 296](#)
  - [virNetworkDefFree, 296](#)
  - [virNetworkDefGetIpByIndex, 296](#)
  - [virNetworkDefParse, 296](#)
  - [virNetworkDefParseFile, 296](#)
  - [virNetworkDefParseNode, 296](#)
  - [virNetworkDefParseString, 296](#)
  - [virNetworkDefParseXML, 296](#)
  - [virNetworkDefUpdateBridge, 297](#)
  - [virNetworkDefUpdateCheckElementName, 297](#)
  - [virNetworkDefUpdateDNSHost, 297](#)
  - [virNetworkDefUpdateDNSSrv, 297](#)
  - [virNetworkDefUpdateDNSTxt, 297](#)
  - [virNetworkDefUpdateDomain, 297](#)
  - [virNetworkDefUpdateForward, 297](#)
  - [virNetworkDefUpdateForwardInterface, 297](#)
  - [virNetworkDefUpdateForwardPF, 297](#)
  - [virNetworkDefUpdateIP, 297](#)
  - [virNetworkDefUpdateIPDHCPHost, 297](#)
  - [virNetworkDefUpdateIPDHCPRange, 297](#)
  - [virNetworkDefUpdateNoSupport, 297](#)
  - [virNetworkDefUpdatePortGroup, 297](#)
  - [virNetworkDefUpdateSection, 297](#)
  - [virNetworkDefUpdateTunnel, 298](#)
  - [virNetworkDefUpdateUnknownCommand, 298](#)
  - [virNetworkDeleteConfig, 298](#)
  - [virNetworkFindByName, 298](#)
  - [virNetworkForwardIfDefClear, 299](#)
  - [virNetworkForwardPfDefClear, 299](#)
  - [virNetworkIppParseXML, 299](#)
  - [virNetworkIpfDefByIndex, 299](#)
  - [virNetworkIpfDefClear, 299](#)
  - [virNetworkIpfDefFormat, 299](#)
  - [virNetworkIpfDefNetmask, 299](#)
  - [virNetworkIpfDefPrefix, 299](#)

- virNetworkList, 299
- virNetworkLoadAllConfigs, 299
- virNetworkLoadConfig, 299
- virNetworkMatch, 299
- virNetworkObjAssignDef, 299
- virNetworkObjFree, 299
- virNetworkObjGetPersistentDef, 299
- virNetworkObjsDuplicate, 299
- virNetworkObjListFree, 299
- virNetworkObjLock, 299
- virNetworkObjReplacePersistentDef, 299
- virNetworkObjSetDefTransient, 299
- virNetworkObjUnlock, 299
- virNetworkObjUnsetDefTransient, 299
- virNetworkObjUpdate, 299
- virNetworkPortGroupParseXML, 299
- virNetworkRemoveInactive, 299
- virNetworkSaveConfig, 299
- virNetworkSaveStatus, 300
- virNetworkSaveXML, 300
- virNetworkSetBridgeMacAddr, 300
- virNetworkSetBridgeName, 300
- virNetworkTunnelParseXML, 300
- virPortGroupDefClear, 300
- virPortGroupDefFormat, 300
- virPortGroupFindByName, 300
- virTunnelDefFormat, 300
- virTunnelDefFree, 300
- network\_conf.h
  - virNetworkAllocateBridge, 305
  - virNetworkAssignDef, 305
  - virNetworkBridgeInUse, 305
  - virNetworkConfigChangeSetup, 305
  - virNetworkConfigFile, 305
  - virNetworkDHCPHostDef, 304
  - virNetworkDHCPHostDefPtr, 304
  - virNetworkDHCPRangeDef, 304
  - virNetworkDHCPRangeDefPtr, 304
  - virNetworkDNSDefPtr, 304
  - virNetworkDNSHostsDefPtr, 304
  - virNetworkDNSSrvRecordsDef, 304
  - virNetworkDNSSrvRecordsDefPtr, 304
  - virNetworkDNSTxtRecordsDef, 304
  - virNetworkDNSTxtRecordsDefPtr, 304
  - virNetworkDef, 304
  - virNetworkDefCopy, 305
  - virNetworkDefFormat, 305
  - virNetworkDefForwardIf, 306
  - virNetworkDefFree, 306
  - virNetworkDefGetIpByIndex, 306
  - virNetworkDefParseFile, 306
  - virNetworkDefParseNode, 306
  - virNetworkDefParseString, 306
  - virNetworkDefPtr, 304
  - virNetworkDeleteConfig, 306
  - virNetworkFindByName, 306
  - virNetworkFindByUUID, 306
  - virNetworkForwardHostdevDeviceType, 305
  - virNetworkForwardIfDef, 304
  - virNetworkForwardIfDefPtr, 304
  - virNetworkForwardPfDef, 304
  - virNetworkForwardPfDefPtr, 304
  - virNetworkForwardType, 305
  - virNetworkIpDef, 304
  - virNetworkIpDefNetmask, 306
  - virNetworkIpDefPrefix, 306
  - virNetworkIpDefPtr, 304
  - virNetworkList, 306
  - virNetworkLoadAllConfigs, 306
  - virNetworkLoadConfig, 306
  - virNetworkObj, 304
  - virNetworkObjAssignDef, 306
  - virNetworkObjFree, 306
  - virNetworkObjGetPersistentDef, 306
  - virNetworkObjsActive, 306
  - virNetworkObjsDuplicate, 306
  - virNetworkObjList, 304
  - virNetworkObjListFree, 306
  - virNetworkObjListPtr, 304
  - virNetworkObjLock, 307
  - virNetworkObjPtr, 304
  - virNetworkObjReplacePersistentDef, 307
  - virNetworkObjSetDefTransient, 307
  - virNetworkObjUnlock, 307
  - virNetworkObjUnsetDefTransient, 307
  - virNetworkObjUpdate, 307
  - virNetworkRemoveInactive, 307
  - virNetworkSaveConfig, 307
  - virNetworkSaveStatus, 307
  - virNetworkSaveXML, 307
  - virNetworkSetBridgeMacAddr, 307
  - virNetworkSetBridgeName, 307
  - virPortGroupDef, 304
  - virPortGroupDefPtr, 304
  - virPortGroupFindByName, 307
  - virTunnelDef, 304
  - virTunnelDefPtr, 305
- network\_driver, 89
  - dnsmasqCaps, 90
  - iptables, 90
  - lock, 90
  - logDir, 90
  - networkAutostartDir, 90
  - networkConfigDir, 90
  - networks, 90
- networkActive
  - bridge\_driver.c, 424
- networkAddAddrToBridge
  - bridge\_driver.c, 424
- networkAddGeneralIp6tablesRules
  - bridge\_driver.c, 424
- networkAddGeneralIptablesRules
  - bridge\_driver.c, 424
- networkAddIpSpecificIptablesRules
  - bridge\_driver.c, 424
- networkAddIptablesRules



- bridge\_driver.c, [424](#)
- networkAddMasqueradingIptablesRules
  - bridge\_driver.c, [424](#)
- networkAddRoutingIptablesRules
  - bridge\_driver.c, [424](#)
- networkAllocateActualDevice
  - bridge\_driver.c, [424](#)
- networkAutostartConfigs
  - bridge\_driver.c, [424](#)
- networkAutostartDir
  - network\_driver, [90](#)
- networkBridgeDummyNicName
  - bridge\_driver.c, [424](#)
- networkBuildDhcpDaemonCommandLine
  - bridge\_driver.c, [424](#)
- networkBuildDnsmasqArgv
  - bridge\_driver.c, [424](#)
- networkBuildDnsmasqHostsfile
  - bridge\_driver.c, [424](#)
- networkCheckRouteCollision
  - bridge\_driver.c, [425](#)
- networkCloseNetwork
  - bridge\_driver.c, [425](#)
- networkCmds
  - virsh-network.c, [453](#)
- networkConfigDir
  - network\_driver, [90](#)
- networkCreate
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkCreateInterfacePool
  - bridge\_driver.c, [425](#)
- networkCreateXML
  - \_virNetworkDriver, [73](#)
- networkDefine
  - bridge\_driver.c, [425](#)
- networkDefineXML
  - \_virNetworkDriver, [73](#)
- networkDestroy
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkDnsmasqLeaseFileName
  - bridge\_driver.c, [428](#)
- networkDnsmasqLeaseFileNameDefault
  - bridge\_driver.c, [425](#)
- networkDriver
  - bridge\_driver.c, [428](#)
- networkDriverLock
  - bridge\_driver.c, [425](#)
- networkDriverUnlock
  - bridge\_driver.c, [425](#)
- networkEnableIpForwarding
  - bridge\_driver.c, [425](#)
- networkFindActiveConfigs
  - bridge\_driver.c, [425](#)
- networkGetAutostart
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkGetBridgeName
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkGetBridgeType
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkGetNetworkAddress
  - bridge\_driver.c, [425](#)
- networkGetXMLDesc
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkIsActive
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkIsPersistent
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkKillDaemon
  - bridge\_driver.c, [425](#)
- networkListAllNetworks
  - bridge\_driver.c, [425](#)
- networkListDefinedNetworks
  - bridge\_driver.c, [425](#)
- networkListNetworks
  - bridge\_driver.c, [425](#)
- networkLookupByName
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [425](#)
- networkLookupByUUID
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [426](#)
- networkNotifyActualDevice
  - bridge\_driver.c, [426](#)
- networkNumDefinedNetworks
  - bridge\_driver.c, [426](#)
- networkNumNetworks
  - bridge\_driver.c, [426](#)
- networkOpenNetwork
  - bridge\_driver.c, [426](#)
- networkRadvdConfContents
  - bridge\_driver.c, [426](#)
- networkRadvdConfWrite
  - bridge\_driver.c, [426](#)
- networkRadvdConfigFileName
  - bridge\_driver.c, [426](#)
- networkRadvdPidfileBasename
  - bridge\_driver.c, [426](#)
- networkRefreshDaemons
  - bridge\_driver.c, [426](#)
- networkRefreshDhcpDaemon
  - bridge\_driver.c, [426](#)
- networkRefreshRadvd
  - bridge\_driver.c, [426](#)
- networkRegister
  - bridge\_driver.c, [426](#)
- networkReleaseActualDevice
  - bridge\_driver.c, [426](#)
- networkReload

- bridge\_driver.c, [426](#)
- networkReloadIptablesRules
  - bridge\_driver.c, [426](#)
- networkRemoveGeneralIptablesRules
  - bridge\_driver.c, [426](#)
- networkRemoveGeneralIptablesRules
  - bridge\_driver.c, [426](#)
- networkRemoveInactive
  - bridge\_driver.c, [426](#)
- networkRemoveIptablesRules
  - bridge\_driver.c, [426](#)
- networkRemoveMasqueradingIptablesRules
  - bridge\_driver.c, [426](#)
- networkRemoveRoutingIptablesRules
  - bridge\_driver.c, [427](#)
- networkRestartDhcpDaemon
  - bridge\_driver.c, [427](#)
- networkSetAutostart
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [427](#)
- networkSetIPv6Sysctls
  - bridge\_driver.c, [427](#)
- networkShutdown
  - bridge\_driver.c, [427](#)
- networkShutdownNetwork
  - bridge\_driver.c, [427](#)
- networkShutdownNetworkExternal
  - bridge\_driver.c, [427](#)
- networkShutdownNetworkVirtual
  - bridge\_driver.c, [427](#)
- networkStart
  - bridge\_driver.c, [427](#)
- networkStartDhcpDaemon
  - bridge\_driver.c, [427](#)
- networkStartNetwork
  - bridge\_driver.c, [427](#)
- networkStartNetworkExternal
  - bridge\_driver.c, [427](#)
- networkStartNetworkVirtual
  - bridge\_driver.c, [427](#)
- networkStartRadvd
  - bridge\_driver.c, [428](#)
- networkStartup
  - bridge\_driver.c, [428](#)
- networkStateDriver
  - bridge\_driver.c, [429](#)
- networkUndefine
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [428](#)
- networkUpdate
  - \_virNetworkDriver, [73](#)
  - bridge\_driver.c, [428](#)
- networkValidate
  - bridge\_driver.c, [428](#)
- networks
  - network\_driver, [90](#)
- newDef
  - \_virDomainObj, [47](#)
  - \_virNetworkObj, [76](#)
- nextslot
  - \_qemuDomainPCIAddressSet, [7](#)
- nfss
  - \_virDomainDef, [21](#)
- ngraphics
  - \_virDomainDef, [21](#)
- nhostdevs
  - \_virDomainDef, [21](#)
- nhosts
  - \_virDomainDiskDef, [29](#)
  - \_virNetworkDNSDef, [70](#)
  - \_virNetworkIpDef, [75](#)
- nhubs
  - \_virDomainDef, [21](#)
- nidentity
  - \_virDomainEventGraphicsSubject, [32](#)
- ninputs
  - \_virDomainDef, [21](#)
- nips
  - \_virNetworkDef, [68](#)
- nix
  - \_virDomainChrSourceDef, [15](#)
- nleases
  - \_virDomainDef, [21](#)
- nnames
  - \_virNetworkDNSHostsDef, [71](#)
- nnets
  - \_virDomainDef, [21](#)
  - vshNetworkList, [91](#)
- no
  - \_virDriver, [65](#)
- node
  - \_virDomainEventGraphicsAddress, [32](#)
- nodeDeviceDetach
  - \_virDriver, [65](#)
- nodeDeviceReAttach
  - \_virDriver, [65](#)
- nodeDeviceReset
  - \_virDriver, [65](#)
- nodeGetCPUStats
  - \_virDriver, [65](#)
- nodeGetCellsFreeMemory
  - \_virDriver, [65](#)
- nodeGetFreeMemory
  - \_virDriver, [65](#)
- nodeGetInfo
  - \_virDriver, [65](#)
- nodeGetMemoryParameters
  - \_virDriver, [65](#)
- nodeGetMemoryStats
  - \_virDriver, [65](#)
- nodeGetSecurityModel
  - \_virDriver, [65](#)
- nodeSetMemoryParameters
  - \_virDriver, [65](#)

- nodeSuspendForDuration
  - \_virDriver, 65
- nodemask
  - \_virDomainNumatuneDef, 46
- nodes
  - \_virNodeInfo, 78
- norelabel
  - \_virSecurityDeviceLabelDef, 81
  - \_virSecurityLabelDef, 82
- nparallels
  - \_virDomainDef, 21
- nrVirtCpu
  - \_virDomainInfo, 40
- nranges
  - \_virNetworkIpDef, 75
- nredirdevs
  - \_virDomainDef, 21
- ns
  - \_virDomainDef, 21
- nseclabels
  - \_virDomainChrDef, 14
  - \_virDomainDef, 21
  - \_virDomainDiskDef, 29
- nserials
  - \_virDomainDef, 21
- nsmartcards
  - \_virDomainDef, 21
- nsounds
  - \_virDomainDef, 21
- nsrvrecords
  - \_virNetworkDNSDef, 70
- ntimers
  - \_virDomainClockDef, 16
- ntxtrecords
  - \_virNetworkDNSDef, 70
- numOfDefinedDomains
  - \_virDriver, 65
- numOfDefinedInterfaces
  - \_virInterfaceDriver, 67
- numOfDefinedNetworks
  - \_virNetworkDriver, 73
- numOfDefinedPools
  - \_virStorageDriver, 84
- numOfDevices
  - \_virDeviceMonitor, 10
- numOfDomains
  - \_virDriver, 65
- numOfInterfaces
  - \_virInterfaceDriver, 67
- numOfNWFilters
  - \_virNWFilterDriver, 79
- numOfNetworks
  - \_virNetworkDriver, 73
- numOfPools
  - \_virStorageDriver, 84
- numOfSecrets
  - \_virSecretDriver, 81
- numatune
  - \_virDomainDef, 21
- number
  - \_virVcpuInfo, 89
- numids
  - virDomainIDData, 90
- numnames
  - virDomainNameData, 91
- nusbdevs
  - \_virDomainRedirFilterDef, 50
- nvcupin
  - \_virDomainDef, 21
- nvideos
  - \_virDomainDef, 21
- nwfilterLookupByName
  - \_virNWFilterDriver, 79
- nwfilterLookupByUUID
  - \_virNWFilterDriver, 79
- object
  - \_virDomainObj, 47
- objs
  - \_virDomainObjList, 48
  - \_virNetworkObjList, 76
- offset
  - \_virDomainClockDef, 16
  - \_virDomainLeaseDef, 42
- onCrash
  - \_virDomainDef, 21
- onPoweroff
  - \_virDomainDef, 21
- onReboot
  - \_virDomainDef, 21
- oom
  - virDomainNameData, 91
- open
  - \_virDeviceMonitor, 10
  - \_virDriver, 65
  - \_virInterfaceDriver, 67
  - \_virNWFilterDriver, 79
  - \_virNetworkDriver, 73
  - \_virSecretDriver, 81
  - \_virStorageDriver, 85
- opts
  - \_virDomainControllerDef, 17
- opts\_network\_autostart
  - virsh-network.c, 454
- opts\_network\_create
  - virsh-network.c, 454
- opts\_network\_define
  - virsh-network.c, 454
- opts\_network\_destroy
  - virsh-network.c, 454
- opts\_network\_dumpxml
  - virsh-network.c, 454
- opts\_network\_edit
  - virsh-network.c, 455
- opts\_network\_info
  - virsh-network.c, 455
- opts\_network\_list

- virsh-network.c, 455
- opts\_network\_name
  - virsh-network.c, 455
- opts\_network\_start
  - virsh-network.c, 455
- opts\_network\_undefine
  - virsh-network.c, 455
- opts\_network\_update
  - virsh-network.c, 456
- opts\_network\_uuid
  - virsh-network.c, 456
- origstates
  - \_virDomainHostdevDef, 37
- os
  - \_virDomainDef, 22
- ovsport
  - virnetdevopenvswitch.h, 445
- PROC\_NET\_ROUTE
  - bridge\_driver.c, 424
- parallels
  - \_virDomainDef, 22
- parent
  - \_virDomainHostdevDef, 37
- passthru
  - \_virDomainSmartcardDef, 51
- passwd
  - \_virDomainGraphicsAuthDef, 34
- path
  - \_virBlkioDeviceWeight, 7
  - \_virDomainChrSourceDef, 15
  - \_virDomainLeaseDef, 42
- pci
  - \_virDomainDeviceInfo, 25
  - \_virDomainHostdevOrigStates, 38
  - \_virDomainHostdevSubsys, 39
  - \_virNetworkForwardIfDef, 74
- period
  - \_virDomainDef, 22
- persistent
  - \_virDomainObj, 47
  - \_virNetworkObj, 76
- physical
  - \_virDomainBlockInfo, 12
- physical\_block\_size
  - \_virDomainDiskDef, 29
- pid
  - \_virDomainObj, 47
- placement\_mode
  - \_virDomainDef, 22
  - \_virDomainNumatuneDef, 46
- playback
  - \_virDomainGraphicsDef, 36
- pm
  - \_virDomainDef, 22
- poolBuild
  - \_virStorageDriver, 85
- poolCreate
  - \_virStorageDriver, 85
- poolCreateXML
  - \_virStorageDriver, 85
- poolDefineXML
  - \_virStorageDriver, 85
- poolDelete
  - \_virStorageDriver, 85
- poolDestroy
  - \_virStorageDriver, 85
- poolGetAutostart
  - \_virStorageDriver, 85
- poolGetInfo
  - \_virStorageDriver, 85
- poolGetXMLDesc
  - \_virStorageDriver, 85
- poolIsActive
  - \_virStorageDriver, 85
- poolIsPersistent
  - \_virStorageDriver, 85
- poolListAllVolumes
  - \_virStorageDriver, 85
- poolListVolumes
  - \_virStorageDriver, 85
- poolLookupByName
  - \_virStorageDriver, 85
- poolLookupByUUID
  - \_virStorageDriver, 85
- poolLookupByVolume
  - \_virStorageDriver, 85
- poolNumOfVolumes
  - \_virStorageDriver, 85
- poolRefresh
  - \_virStorageDriver, 85
- poolSetAutostart
  - \_virStorageDriver, 85
- poolUndefine
  - \_virStorageDriver, 85
- port
  - \_virDomainChrDef, 14
  - \_virDomainDeviceUSBAddress, 26
  - \_virDomainDeviceVirtioSerialAddress, 27
  - \_virDomainDiskHostDef, 31
  - \_virDomainGraphicsDef, 36
  - \_virDomainNetDef, 45
  - \_virNetworkDNSSrvRecordsDef, 71
- portGroups
  - \_virNetworkDef, 69
- portgroup
  - \_virDomainNetDef, 45
- ports
  - \_virDomainVirtioSerialOpts, 55
- prefix
  - \_virNetworkIpDef, 75
- present
  - \_virDomainTimerDef, 53
- priority
  - \_virNetworkDNSSrvRecordsDef, 71
- privateData
  - \_virDomainObj, 47

- privateDataFreeFunc
  - \_virDomainObj, [47](#)
- product
  - \_virDomainHostdevSubsys, [39](#)
  - \_virDomainRedirFilterUsbDevDef, [50](#)
- prompt
  - \_virConnectCredential, [8](#)
- protocol
  - \_virDomainChrSourceDef, [15](#)
  - \_virDomainDiskDef, [29](#)
  - \_virNetworkDNSSrvRecordsDef, [71](#)
- qemu\_command.c
  - IS\_USB2\_CONTROLLER, [432](#)
  - qemuAddRBDHost, [433](#)
  - qemuAssignDeviceAliases, [433](#)
  - qemuAssignDeviceControllerAlias, [433](#)
  - qemuAssignDeviceDiskAlias, [433](#)
  - qemuAssignDeviceDiskAliasCustom, [433](#)
  - qemuAssignDeviceDiskAliasFixed, [433](#)
  - qemuAssignDeviceDiskAliasLegacy, [433](#)
  - qemuAssignDeviceHostdevAlias, [433](#)
  - qemuAssignDeviceNetAlias, [433](#)
  - qemuAssignDevicePCISlots, [433](#)
  - qemuAssignDeviceRedirdevAlias, [433](#)
  - qemuAssignSpaprVIOAddress, [433](#)
  - qemuBuildChrArgStr, [433](#)
  - qemuBuildChrChardevStr, [433](#)
  - qemuBuildChrDeviceStr, [433](#)
  - qemuBuildClockArgStr, [433](#)
  - qemuBuildCommandLine, [433](#)
  - qemuBuildControllerDevStr, [434](#)
  - qemuBuildCpuArgStr, [434](#)
  - qemuBuildDeviceAddressStr, [434](#)
  - qemuBuildDriveDevStr, [434](#)
  - qemuBuildDriveStr, [434](#)
  - qemuBuildFSDDevStr, [434](#)
  - qemuBuildFSStr, [434](#)
  - qemuBuildHostNetStr, [434](#)
  - qemuBuildHubDevStr, [434](#)
  - qemuBuildIoEventFdStr, [434](#)
  - qemuBuildMachineArgStr, [434](#)
  - qemuBuildMemballoonDevStr, [434](#)
  - qemuBuildNicDevStr, [434](#)
  - qemuBuildNicStr, [434](#)
  - qemuBuildNumaArgStr, [434](#)
  - qemuBuildPCIHostdevDevStr, [434](#)
  - qemuBuildPCIHostdevPCIDevStr, [434](#)
  - qemuBuildRBDString, [434](#)
  - qemuBuildRedirdevDevStr, [434](#)
  - qemuBuildRomStr, [434](#)
  - qemuBuildSmbiosBiosStr, [434](#)
  - qemuBuildSmbiosSystemStr, [434](#)
  - qemuBuildSmpArgStr, [435](#)
  - qemuBuildSoundCodecStr, [435](#)
  - qemuBuildSoundDevStr, [435](#)
  - qemuBuildUSBControllerDevStr, [435](#)
  - qemuBuildUSBHostdevDevStr, [435](#)
  - qemuBuildUSBHostdevUsbDevStr, [435](#)
  - qemuBuildUSBInputDevStr, [435](#)
  - qemuBuildVideoDevStr, [435](#)
  - qemuBuildVirtioSerialPortDevStr, [435](#)
  - qemuBuildWatchdogDevStr, [435](#)
  - qemuCollectPCIAddress, [435](#)
  - qemuControllerModelUSBToCaps, [435](#)
  - qemuDeviceDriveHostAlias, [435](#)
  - qemuDomainAssignAddresses, [435](#)
  - qemuDomainAssignPCIAddresses, [435](#)
  - qemuDomainAssignS390Addresses, [435](#)
  - qemuDomainAssignSpaprVIOAddresses, [435](#)
  - qemuDomainDeviceAliasIndex, [435](#)
  - qemuDomainNetVLAN, [435](#)
  - qemuDomainPCIAddressCheckSlot, [435](#)
  - qemuDomainPCIAddressEnsureAddr, [435](#)
  - qemuDomainPCIAddressGetNextSlot, [435](#)
  - qemuDomainPCIAddressReleaseAddr, [435](#)
  - qemuDomainPCIAddressReleaseFunction, [435](#)
  - qemuDomainPCIAddressReleaseSlot, [435](#)
  - qemuDomainPCIAddressReserveAddr, [435](#)
  - qemuDomainPCIAddressReserveFunction, [436](#)
  - qemuDomainPCIAddressReserveSlot, [436](#)
  - qemuDomainPCIAddressSetCreate, [436](#)
  - qemuDomainPCIAddressSetFree, [436](#)
  - qemuDomainPCIAddressSetFreeEntry, [436](#)
  - qemuDomainPCIAddressSetNextAddr, [436](#)
  - qemuDomainPrimeS390VirtioDevices, [436](#)
  - qemuFindEnv, [436](#)
  - qemuFindNICForVLAN, [436](#)
  - qemuInitGuestCPU, [436](#)
  - qemuNetworkIfaceConnect, [436](#)
  - qemuOpenPCIConfig, [436](#)
  - qemuOpenVhostNet, [436](#)
  - qemuPCIAddressAsString, [437](#)
  - qemuParseCommandLine, [436](#)
  - qemuParseCommandLineBootDevs, [436](#)
  - qemuParseCommandLineCPU, [436](#)
  - qemuParseCommandLineChr, [436](#)
  - qemuParseCommandLineDisk, [436](#)
  - qemuParseCommandLineNet, [437](#)
  - qemuParseCommandLinePCI, [437](#)
  - qemuParseCommandLinePid, [437](#)
  - qemuParseCommandLineSmp, [437](#)
  - qemuParseCommandLineString, [437](#)
  - qemuParseCommandLineUSB, [437](#)
  - qemuParseKeywords, [437](#)
  - qemuParseProcFileStrings, [437](#)
  - qemuParseRBDString, [437](#)
  - qemuPhysIfaceConnect, [437](#)
  - qemuSafeSerialParamValue, [437](#)
  - qemuSetScsiControllerModel, [437](#)
  - qemuSoundCodecTypeToCaps, [437](#)
  - qemuSpaprVIOFindByReg, [437](#)
  - qemuStringToArgvEnv, [437](#)
  - qemuUsbId, [437](#)
  - VIR\_ENUM\_IMPL, [437](#)
  - VIR\_FROM\_THIS, [433](#)
  - WANT\_VALUE, [433](#)

- qemuAddRBDHost
  - qemu\_command.c, [433](#)
- qemuAssignDeviceAliases
  - qemu\_command.c, [433](#)
- qemuAssignDeviceControllerAlias
  - qemu\_command.c, [433](#)
- qemuAssignDeviceDiskAlias
  - qemu\_command.c, [433](#)
- qemuAssignDeviceDiskAliasCustom
  - qemu\_command.c, [433](#)
- qemuAssignDeviceDiskAliasFixed
  - qemu\_command.c, [433](#)
- qemuAssignDeviceDiskAliasLegacy
  - qemu\_command.c, [433](#)
- qemuAssignDeviceHostdevAlias
  - qemu\_command.c, [433](#)
- qemuAssignDeviceNetAlias
  - qemu\_command.c, [433](#)
- qemuAssignDevicePCISlots
  - qemu\_command.c, [433](#)
- qemuAssignDeviceRedirdevAlias
  - qemu\_command.c, [433](#)
- qemuAssignSpaprVIOAddress
  - qemu\_command.c, [433](#)
- qemuBuildChrArgStr
  - qemu\_command.c, [433](#)
- qemuBuildChrChardevStr
  - qemu\_command.c, [433](#)
- qemuBuildChrDeviceStr
  - qemu\_command.c, [433](#)
- qemuBuildClockArgStr
  - qemu\_command.c, [433](#)
- qemuBuildCommandLine
  - qemu\_command.c, [433](#)
- qemuBuildControllerDevStr
  - qemu\_command.c, [434](#)
- qemuBuildCpuArgStr
  - qemu\_command.c, [434](#)
- qemuBuildDeviceAddressStr
  - qemu\_command.c, [434](#)
- qemuBuildDriveDevStr
  - qemu\_command.c, [434](#)
- qemuBuildDriveStr
  - qemu\_command.c, [434](#)
- qemuBuildFSDevStr
  - qemu\_command.c, [434](#)
- qemuBuildFSStr
  - qemu\_command.c, [434](#)
- qemuBuildHostNetStr
  - qemu\_command.c, [434](#)
- qemuBuildHubDevStr
  - qemu\_command.c, [434](#)
- qemuBuildIoEventFdStr
  - qemu\_command.c, [434](#)
- qemuBuildMachineArgStr
  - qemu\_command.c, [434](#)
- qemuBuildMemballoonDevStr
  - qemu\_command.c, [434](#)
- qemuBuildNicDevStr
  - qemu\_command.c, [434](#)
- qemuBuildNicStr
  - qemu\_command.c, [434](#)
- qemuBuildNumaArgStr
  - qemu\_command.c, [434](#)
- qemuBuildPCIHostdevDevStr
  - qemu\_command.c, [434](#)
- qemuBuildPCIHostdevPCIDevStr
  - qemu\_command.c, [434](#)
- qemuBuildRBDString
  - qemu\_command.c, [434](#)
- qemuBuildRedirdevDevStr
  - qemu\_command.c, [434](#)
- qemuBuildRomStr
  - qemu\_command.c, [434](#)
- qemuBuildSmbiosBiosStr
  - qemu\_command.c, [434](#)
- qemuBuildSmbiosSystemStr
  - qemu\_command.c, [434](#)
- qemuBuildSmpArgStr
  - qemu\_command.c, [435](#)
- qemuBuildSoundCodecStr
  - qemu\_command.c, [435](#)
- qemuBuildSoundDevStr
  - qemu\_command.c, [435](#)
- qemuBuildUSBControllerDevStr
  - qemu\_command.c, [435](#)
- qemuBuildUSBHostdevDevStr
  - qemu\_command.c, [435](#)
- qemuBuildUSBHostdevUsbDevStr
  - qemu\_command.c, [435](#)
- qemuBuildUSBInputDevStr
  - qemu\_command.c, [435](#)
- qemuBuildVideoDevStr
  - qemu\_command.c, [435](#)
- qemuBuildVirtioSerialPortDevStr
  - qemu\_command.c, [435](#)
- qemuBuildWatchdogDevStr
  - qemu\_command.c, [435](#)
- qemuCollectPCIAddress
  - qemu\_command.c, [435](#)
- qemuControllerModelUSBToCaps
  - qemu\_command.c, [435](#)
- qemuDeviceDriveHostAlias
  - qemu\_command.c, [435](#)
- qemuDomainArbitraryAgentCommand
  - \_virDriver, [65](#)
- qemuDomainAssignAddresses
  - qemu\_command.c, [435](#)
- qemuDomainAssignPCIAddresses
  - qemu\_command.c, [435](#)
- qemuDomainAssignS390Addresses
  - qemu\_command.c, [435](#)
- qemuDomainAssignSpaprVIOAddresses
  - qemu\_command.c, [435](#)
- qemuDomainAttach
  - \_virDriver, [65](#)



- qemuDomainDeviceAliasIndex
  - qemu\_command.c, [435](#)
- qemuDomainMonitorCommand
  - \_virDriver, [65](#)
- qemuDomainNetVLAN
  - qemu\_command.c, [435](#)
- qemuDomainPCIAddressCheckSlot
  - qemu\_command.c, [435](#)
- qemuDomainPCIAddressEnsureAddr
  - qemu\_command.c, [435](#)
- qemuDomainPCIAddressGetNextSlot
  - qemu\_command.c, [435](#)
- qemuDomainPCIAddressReleaseAddr
  - qemu\_command.c, [435](#)
- qemuDomainPCIAddressReleaseFunction
  - qemu\_command.c, [435](#)
- qemuDomainPCIAddressReleaseSlot
  - qemu\_command.c, [435](#)
- qemuDomainPCIAddressReserveAddr
  - qemu\_command.c, [435](#)
- qemuDomainPCIAddressReserveFunction
  - qemu\_command.c, [436](#)
- qemuDomainPCIAddressReserveSlot
  - qemu\_command.c, [436](#)
- qemuDomainPCIAddressSetCreate
  - qemu\_command.c, [436](#)
- qemuDomainPCIAddressSetFree
  - qemu\_command.c, [436](#)
- qemuDomainPCIAddressSetFreeEntry
  - qemu\_command.c, [436](#)
- qemuDomainPCIAddressSetNextAddr
  - qemu\_command.c, [436](#)
- qemuDomainPrimeS390VirtioDevices
  - qemu\_command.c, [436](#)
- qemuFindEnv
  - qemu\_command.c, [436](#)
- qemuFindNICForVLAN
  - qemu\_command.c, [436](#)
- qemuInitGuestCPU
  - qemu\_command.c, [436](#)
- qemuNetworkInterfaceConnect
  - qemu\_command.c, [436](#)
- qemuOpenPCIConfig
  - qemu\_command.c, [436](#)
- qemuOpenVhostNet
  - qemu\_command.c, [436](#)
- qemuPCIAddressAsString
  - qemu\_command.c, [437](#)
- qemuParseCommandLine
  - qemu\_command.c, [436](#)
- qemuParseCommandLineBootDevs
  - qemu\_command.c, [436](#)
- qemuParseCommandLineCPU
  - qemu\_command.c, [436](#)
- qemuParseCommandLineChr
  - qemu\_command.c, [436](#)
- qemuParseCommandLineDisk
  - qemu\_command.c, [436](#)
- qemuParseCommandLineNet
  - qemu\_command.c, [437](#)
- qemuParseCommandLinePCI
  - qemu\_command.c, [437](#)
- qemuParseCommandLinePid
  - qemu\_command.c, [437](#)
- qemuParseCommandLineSmp
  - qemu\_command.c, [437](#)
- qemuParseCommandLineString
  - qemu\_command.c, [437](#)
- qemuParseCommandLineUSB
  - qemu\_command.c, [437](#)
- qemuParseKeywords
  - qemu\_command.c, [437](#)
- qemuParseProcFileStrings
  - qemu\_command.c, [437](#)
- qemuParseRBDString
  - qemu\_command.c, [437](#)
- qemuPhysIfaceConnect
  - qemu\_command.c, [437](#)
- qemuSafeSerialParamValue
  - qemu\_command.c, [437](#)
- qemuSetScsiControllerModel
  - qemu\_command.c, [437](#)
- qemuSoundCodecTypeToCaps
  - qemu\_command.c, [437](#)
- qemuSpaprVIOFindByReg
  - qemu\_command.c, [437](#)
- qemuStringToArgvEnv
  - qemu\_command.c, [437](#)
- qemuUsbId
  - qemu\_command.c, [437](#)
- quota
  - \_virDomainDef, [22](#)
- RADVD\_STATE\_DIR
  - bridge\_driver.c, [424](#)
- radvdPid
  - \_virNetworkObj, [76](#)
- ranges
  - \_virNetworkIpDef, [75](#)
- rawio
  - \_virDomainDiskDef, [30](#)
- rawio\_specified
  - \_virDomainDiskDef, [30](#)
- rd\_bytes
  - \_virDomainBlockStats, [13](#)
- rd\_req
  - \_virDomainBlockStats, [13](#)
- rdp
  - \_virDomainGraphicsDef, [36](#)
- read\_bytes\_sec
  - \_virDomainBlockIoTuneInfo, [12](#)
- read\_iops\_sec
  - \_virDomainBlockIoTuneInfo, [12](#)
- readonly
  - \_virDomainDiskDef, [30](#)
  - \_virDomainFSDef, [33](#)
- reason

- [\\_virDomainStateReason, 52](#)
- redirdev
  - [\\_virDomainDeviceDef, 24](#)
- redirdevs
  - [\\_virDomainDef, 22](#)
- redirfilter
  - [\\_virDomainDef, 22](#)
- reg
  - [\\_virDomainDeviceSpaprVioAddress, 26](#)
- remote\_ip
  - [\\_virTunnelDef, 88](#)
- remove\_slot
  - [\\_virDomainHostdevOrigStates, 38](#)
- replaceUser
  - [\\_virDomainGraphicsDef, 36](#)
- reprobe
  - [\\_virDomainHostdevOrigStates, 38](#)
- error\_policy
  - [\\_virDomainDiskDef, 30](#)
- result
  - [\\_virConnectCredential, 8](#)
- resultlen
  - [\\_virConnectCredential, 8](#)
- rombar
  - [\\_virDomainDeviceInfo, 25](#)
- romfile
  - [\\_virDomainDeviceInfo, 25](#)
- root
  - [\\_virDomainOSDef, 49](#)
- rt\_delay
  - [\\_virDomainBIOSDef, 11](#)
- rt\_set
  - [\\_virDomainBIOSDef, 11](#)
- rx\_bytes
  - [\\_virDomainInterfaceStats, 41](#)
- rx\_drop
  - [\\_virDomainInterfaceStats, 41](#)
- rx\_errs
  - [\\_virDomainInterfaceStats, 41](#)
- rx\_packets
  - [\\_virDomainInterfaceStats, 41](#)
- s
  - [\\_virTypedParameter, 89](#)
- s3
  - [\\_virDomainDef, 22](#)
- s4
  - [\\_virDomainDef, 22](#)
- SYSCTL\_PATH
  - [bridge\\_driver.c, 424](#)
- script
  - [\\_virDomainNetDef, 45](#)
- sdl
  - [\\_virDomainGraphicsDef, 36](#)
- seclabels
  - [\\_virDomainChrDef, 14](#)
  - [\\_virDomainDef, 22](#)
  - [\\_virDomainDiskDef, 30](#)
- secret
  - [\\_virDomainDiskDef, 30](#)
- secretType
  - [\\_virDomainDiskDef, 30](#)
- sectors
  - [\\_virDomainDiskDef, 30](#)
- serial
  - [\\_virDomainDiskDef, 30](#)
- serials
  - [\\_virDomainDef, 22](#)
- service
  - [\\_virDomainChrSourceDef, 15](#)
  - [\\_virDomainEventGraphicsAddress, 32](#)
  - [\\_virNetworkDNSSrvRecordsDef, 71](#)
- setKeepAlive
  - [\\_virDriver, 65](#)
- setValue
  - [\\_virSecretDriver, 81](#)
- shared
  - [\\_virDomainDiskDef, 30](#)
- shares
  - [\\_virDomainDef, 22](#)
- slew
  - [\\_virDomainTimerCatchupDef, 53](#)
- slot
  - [\\_virDomainDeviceCcidAddress, 23](#)
- smartcard
  - [\\_virDomainDeviceDef, 24](#)
- smartcards
  - [\\_virDomainDef, 22](#)
- smbios\_mode
  - [\\_virDomainOSDef, 49](#)
- snapshot
  - [\\_virDomainDiskDef, 30](#)
- snapshots
  - [\\_virDomainObj, 47](#)
- sndbuf
  - [\\_virDomainNetDef, 45](#)
- sndbuf\_specified
  - [\\_virDomainNetDef, 45](#)
- socket
  - [\\_virDomainGraphicsDef, 36](#)
  - [\\_virDomainNetDef, 45](#)
- sockets
  - [\\_virNodeInfo, 78](#)
- soft\_limit
  - [\\_virDomainDef, 22](#)
- sound
  - [\\_virDomainDeviceDef, 24](#)
- sounds
  - [\\_virDomainDef, 22](#)
- source
  - [\\_virDomainChrDef, 14](#)
  - [\\_virDomainHostdevDef, 37](#)
  - [\\_virDomainRedirdevDef, 49](#)
- source\_bridge
  - [\\_virNetworkDef, 69](#)
- space\_hard\_limit
  - [\\_virDomainFSDef, 33](#)



- space\_soft\_limit
  - \_virDomainFSDef, [33](#)
- spaprpio
  - \_virDomainDeviceInfo, [26](#)
- spice
  - \_virDomainGraphicsDef, [36](#)
- spicevmc
  - \_virDomainChrSourceDef, [15](#)
- src
  - \_virDomainDiskDef, [30](#)
  - \_virDomainFSDef, [33](#)
  - src/conf/domain\_conf.c, [227](#)
  - src/conf/domain\_conf.h, [251](#)
  - src/conf/network\_conf.c, [293](#)
  - src/conf/network\_conf.h, [301](#)
  - src/driver.h, [307](#)
  - src/libvirt.c, [330](#)
  - src/network/bridge\_driver.c, [420](#)
  - src/qemu/qemu\_command.c, [429](#)
  - src/uml/uml\_conf.c, [438](#)
  - src/util/virnetdevbridge.c, [440](#)
  - src/util/virnetdevopenvswitch.c, [441](#)
  - src/util/virnetdevopenvswitch.h, [443](#)
  - src/util/virnetdevtap.c, [445](#)
  - src/util/virnetdevtap.h, [447](#)
- srvrecords
  - \_virNetworkDNSDef, [70](#)
- start
  - \_virNetworkDHCPRangeDef, [70](#)
- startport
  - \_virDomainDeviceUSBMaster, [27](#)
- startupPolicy
  - \_virDomainDiskDef, [30](#)
- state
  - \_virDomainControllInfo, [17](#)
  - \_virDomainInfo, [40](#)
  - \_virDomainObj, [47](#)
  - \_virDomainStateReason, [52](#)
  - \_virStoragePoolInfo, [86](#)
  - \_virVcpuInfo, [89](#)
- stateTime
  - \_virDomainControllInfo, [17](#)
- states
  - \_virDomainHostdevOrigStates, [38](#)
- stp
  - \_virNetworkDef, [69](#)
- streamAbort
  - \_virStreamDriver, [87](#)
- streamAddCallback
  - \_virStreamDriver, [87](#)
- streamFinish
  - \_virStreamDriver, [87](#)
- streamRecv
  - \_virStreamDriver, [87](#)
- streamRemoveCallback
  - \_virStreamDriver, [87](#)
- streamSend
  - \_virStreamDriver, [87](#)
- streamUpdateCallback
  - \_virStreamDriver, [87](#)
- streaming
  - \_virDomainGraphicsDef, [36](#)
- subsys
  - \_virDomainHostdevDef, [37](#)
- support2d
  - \_virDomainVideoAccelDef, [54](#)
- support3d
  - \_virDomainVideoAccelDef, [54](#)
- supports\_feature
  - \_virDriver, [65](#)
- swap\_hard\_limit
  - \_virDomainDef, [22](#)
- sysinfo
  - \_virDomainDef, [22](#)
- tag
  - \_virDomainMemoryStat, [43](#)
- taint
  - \_virDomainObj, [47](#)
- target
  - \_virDomainChrDef, [14](#)
  - \_virDomainDeviceDriveAddress, [24](#)
  - \_virNetworkDNSSrvRecordsDef, [71](#)
- targetType
  - \_virDomainChrDef, [14](#)
- tcp
  - \_virDomainChrSourceDef, [15](#)
- tftpport
  - \_virNetworkIpDef, [75](#)
- threads
  - \_virNodeInfo, [78](#)
- threshold
  - \_virDomainTimerCatchupDef, [53](#)
- tickpolicy
  - \_virDomainTimerDef, [53](#)
- timeElapsed
  - \_virDomainJobInfo, [42](#)
- timeRemaining
  - \_virDomainJobInfo, [42](#)
- timers
  - \_virDomainClockDef, [16](#)
- timezone
  - \_virDomainClockDef, [16](#)
- title
  - \_virDomainDef, [22](#)
- tlsPort
  - \_virDomainGraphicsDef, [36](#)
- tools/virsh-network.c, [448](#)
- total\_bytes\_sec
  - \_virDomainBlockIoTuneInfo, [12](#)
- total\_iops\_sec
  - \_virDomainBlockIoTuneInfo, [12](#)
- track
  - \_virDomainTimerDef, [53](#)
- trans
  - \_virDomainDiskDef, [30](#)
- transient

- [\\_virDomainDiskDef, 30](#)
- [tray\\_status](#)
  - [\\_virDomainDiskDef, 30](#)
- [trunks](#)
  - [\\_virTunnelDef, 88](#)
- [tune](#)
  - [\\_virDomainNetDef, 46](#)
- [tunnels](#)
  - [\\_virNetworkDef, 69](#)
- [tx\\_bytes](#)
  - [\\_virDomainInterfaceStats, 41](#)
- [tx\\_drop](#)
  - [\\_virDomainInterfaceStats, 41](#)
- [tx\\_errs](#)
  - [\\_virDomainInterfaceStats, 41](#)
- [tx\\_packets](#)
  - [\\_virDomainInterfaceStats, 41](#)
- [txmode](#)
  - [\\_virDomainNetDef, 46](#)
- [txtrecords](#)
  - [\\_virNetworkDNSDef, 70](#)
- [type](#)
  - [\\_virConnectCredential, 8](#)
  - [\\_virDomainActualNetDef, 11](#)
  - [\\_virDomainBlockJobInfo, 13](#)
  - [\\_virDomainChrSourceDef, 15](#)
  - [\\_virDomainControllerDef, 17](#)
  - [\\_virDomainDeviceDef, 24](#)
  - [\\_virDomainDeviceInfo, 26](#)
  - [\\_virDomainDiskDef, 30](#)
  - [\\_virDomainEventGraphicsSubjectIdentity, 33](#)
  - [\\_virDomainFSDef, 34](#)
  - [\\_virDomainGraphicsDef, 36](#)
  - [\\_virDomainGraphicsListenDef, 37](#)
  - [\\_virDomainHostdevSubsys, 39](#)
  - [\\_virDomainHubDef, 39](#)
  - [\\_virDomainInputDef, 40](#)
  - [\\_virDomainJobInfo, 42](#)
  - [\\_virDomainNetDef, 46](#)
  - [\\_virDomainOSDef, 49](#)
  - [\\_virDomainSmartcardDef, 51](#)
  - [\\_virDomainSoundCodecDef, 51](#)
  - [\\_virDomainVideoDef, 55](#)
  - [\\_virDriver, 65](#)
  - [\\_virNetworkForwardIfDef, 74](#)
  - [\\_virSecurityLabelDef, 82](#)
  - [\\_virStorageVolInfo, 87](#)
  - [\\_virTypedParameter, 89](#)
- [u](#)
  - [\\_virDomainHostdevSubsys, 39](#)
- [URI\\_ALIAS\\_CHARS](#)
  - [libvirt.c, 340](#)
- [udp](#)
  - [\\_virDomainChrSourceDef, 15](#)
- [ui](#)
  - [\\_virTypedParameter, 89](#)
- [ul](#)
  - [\\_virTypedParameter, 89](#)
- [uml\\_conf.c](#)
  - [umlBuildCommandLine, 438](#)
  - [umlBuildCommandLineChr, 438](#)
  - [umlBuildCommandLineNet, 439](#)
  - [umlCapsInit, 439](#)
  - [umlConnectTapDevice, 439](#)
  - [umlDefaultConsoleType, 439](#)
  - [umlLog, 438](#)
  - [umlNextArg, 439](#)
  - [VIR\\_FROM\\_THIS, 438](#)
- [umlBuildCommandLine](#)
  - [uml\\_conf.c, 438](#)
- [umlBuildCommandLineChr](#)
  - [uml\\_conf.c, 438](#)
- [umlBuildCommandLineNet](#)
  - [uml\\_conf.c, 439](#)
- [umlCapsInit](#)
  - [uml\\_conf.c, 439](#)
- [umlConnectTapDevice](#)
  - [uml\\_conf.c, 439](#)
- [umlDefaultConsoleType](#)
  - [uml\\_conf.c, 439](#)
- [umlLog](#)
  - [uml\\_conf.c, 438](#)
- [umlNextArg](#)
  - [uml\\_conf.c, 439](#)
- [unbind\\_from\\_stub](#)
  - [\\_virDomainHostdevOrigStates, 38](#)
- [undefine](#)
  - [\\_virNWFilterDriver, 79](#)
  - [\\_virSecretDriver, 81](#)
- [unit](#)
  - [\\_virDomainDeviceDriveAddress, 25](#)
- [updated](#)
  - [\\_virDomainObj, 47](#)
- [usage](#)
  - [\\_virDomainDiskDef, 30](#)
  - [\\_virDomainFSDef, 34](#)
- [usb](#)
  - [\\_virDomainDeviceInfo, 26](#)
  - [\\_virDomainHostdevSubsys, 39](#)
- [usbClass](#)
  - [\\_virDomainRedirFilterUsbDevDef, 50](#)
- [usbdevs](#)
  - [\\_virDomainRedirFilterDef, 50](#)
- [used](#)
  - [\\_qemuDomainPCIAddressSet, 7](#)
- [username](#)
  - [\\_virDomainDiskDef, 30](#)
- [useserial](#)
  - [\\_virDomainBIOSDef, 11](#)
- [utc\\_reset](#)
  - [\\_virDomainClockDef, 16](#)
- [uuid](#)
  - [\\_virDomainDef, 22](#)
  - [\\_virDomainDiskDef, 30](#)
  - [\\_virNetworkDef, 69](#)
- [uuid\\_specified](#)

- [\\_virNetworkDef](#), 69
- [VIR\\_CONNECT\\_CLOSE\\_REASON\\_CLIENT](#)  
[libvirt.h](#), 132
- [VIR\\_CONNECT\\_CLOSE\\_REASON\\_EOF](#)  
[libvirt.h](#), 132
- [VIR\\_CONNECT\\_CLOSE\\_REASON\\_ERROR](#)  
[libvirt.h](#), 132
- [VIR\\_CONNECT\\_CLOSE\\_REASON\\_KEEPAIVE](#)  
[libvirt.h](#), 132
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_ACTIVE](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_AUTOSTART](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_HAS\\_SNAPSHOT](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_INACTIVE](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_MANAGEDSAVE](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_NO\\_AUTOSTART](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_NO\\_MANAGEDSAVE](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_NO\\_SNAPSHOT](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_OTHER](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_PAUSED](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_PERSISTENT](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_RUNNING](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_SHUTOFF](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_DOMAINS\\_TRANSIENT](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_INTERFACES\\_ACTIVE](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_INTERFACES\\_INACTIVE](#)  
[libvirt.h](#), 133
- [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_ACTIVE](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_AUTOSTART](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_INACTIVE](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_NO\\_AUTOSTART](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_PERSISTENT](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NETWORKS\\_TRANSIENT](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_NET](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_PCI\\_DEV](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_SCSI](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_SCSI\\_HOST](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_SCSI\\_TARGET](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_STORAGE](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_SYSTEM](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_USB\\_DEV](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_NODE\\_DEVICES\\_CAP\\_USB\\_INTERFACE](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_SECRETS\\_EPHEMERAL](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_SECRETS\\_NO\\_EPHEMERAL](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_SECRETS\\_NO\\_PRIVATE](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_SECRETS\\_PRIVATE](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_ACTIVE](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_AUTOSTART](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_DIR](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_DISK](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_FS](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_INACTIVE](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_ISCSI](#)  
[libvirt.h](#), 135
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_LOGICAL](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_MPATH](#)  
[libvirt.h](#), 135
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_NETFS](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_NO\\_AUTOSTART](#)  
[libvirt.h](#), 134
- [VIR\\_CONNECT\\_LIST\\_STORAGE\\_POOLS\\_PERSISTENT](#)  
[libvirt.h](#), 134

- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_RBD  
libvirt.h, [135](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_SCSI  
libvirt.h, [135](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_SHEEPDOG  
libvirt.h, [135](#)
- VIR\_CONNECT\_LIST\_STORAGE\_POOLS\_TRANSIENT  
libvirt.h, [134](#)
- VIR\_CONNECT\_NO\_ALIASES  
libvirt.h, [133](#)
- VIR\_CONNECT\_RO  
libvirt.h, [133](#)
- VIR\_CPU\_COMPARE\_ERROR  
libvirt.h, [135](#)
- VIR\_CPU\_COMPARE\_IDENTICAL  
libvirt.h, [135](#)
- VIR\_CPU\_COMPARE\_INCOMPATIBLE  
libvirt.h, [135](#)
- VIR\_CPU\_COMPARE\_SUPERSET  
libvirt.h, [135](#)
- VIR\_CRED\_AUTHNAME  
libvirt.h, [132](#)
- VIR\_CRED\_CNONCE  
libvirt.h, [132](#)
- VIR\_CRED\_ECHOPROMPT  
libvirt.h, [132](#)
- VIR\_CRED\_EXTERNAL  
libvirt.h, [132](#)
- VIR\_CRED\_LANGUAGE  
libvirt.h, [132](#)
- VIR\_CRED\_NOECHOPROMPT  
libvirt.h, [132](#)
- VIR\_CRED\_PASSPHRASE  
libvirt.h, [132](#)
- VIR\_CRED\_REALM  
libvirt.h, [132](#)
- VIR\_CRED\_USERNAME  
libvirt.h, [132](#)
- VIR\_DOMAIN\_AFFECT\_CONFIG  
libvirt.h, [143](#)
- VIR\_DOMAIN\_AFFECT\_CURRENT  
libvirt.h, [143](#)
- VIR\_DOMAIN\_AFFECT\_LIVE  
libvirt.h, [143](#)
- VIR\_DOMAIN\_APIC\_EOI\_DEFAULT  
domain\_conf.h, [269](#)
- VIR\_DOMAIN\_APIC\_EOI\_LAST  
domain\_conf.h, [269](#)
- VIR\_DOMAIN\_APIC\_EOI\_OFF  
domain\_conf.h, [269](#)
- VIR\_DOMAIN\_APIC\_EOI\_ON  
domain\_conf.h, [269](#)
- VIR\_DOMAIN\_BIOS\_USESERIAL\_DEFAULT  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BIOS\_USESERIAL\_NO  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BIOS\_USESERIAL\_YES  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_BOOLEAN  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_DOUBLE  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_INT  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_LLONG  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_UINT  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLKIO\_PARAM\_ULLONG  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLOCK\_COMMIT\_DELETE  
libvirt.h, [135](#)
- VIR\_DOMAIN\_BLOCK\_COMMIT\_SHALLOW  
libvirt.h, [135](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_ASYNC  
libvirt.h, [135](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_ABORT\_PIVOT  
libvirt.h, [135](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_CANCELED  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_COMPLETED  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_FAILED  
libvirt.h, [132](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_COMMIT  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_COPY  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_PULL  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCK\_JOB\_TYPE\_UNKNOWN  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCK\_REBASE\_COPY  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCK\_REBASE\_COPY\_RAW  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCK\_REBASE\_REUSE\_EXT  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCK\_REBASE\_SHALLOW  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCK\_RESIZE\_BYTES  
libvirt.h, [136](#)
- VIR\_DOMAIN\_BLOCKED  
libvirt.h, [146](#)
- VIR\_DOMAIN\_BLOCKED\_UNKNOWN  
libvirt.h, [135](#)
- VIR\_DOMAIN\_BOOT\_CDROM  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BOOT\_DISK  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BOOT\_FLOPPY  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BOOT\_LAST  
domain\_conf.h, [270](#)

- VIR\_DOMAIN\_BOOT\_MENU\_DEFAULT  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BOOT\_MENU\_DISABLED  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BOOT\_MENU\_ENABLED  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BOOT\_MENU\_LAST  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_BOOT\_NET  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE\_GUESTFWD  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE\_LAST  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE\_NONE  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CHANNEL\_TARGET\_TYPE\_VIRTIO  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_LAST  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_LXC  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_OPENVZ  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_SERIAL  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_UML  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_VIRTIO  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_CONSOLE\_TARGET\_TYPE\_XEN  
domain\_conf.h, [270](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_CHANNEL  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_CONSOLE  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_LAST  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_PARALLEL  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_DEVICE\_TYPE\_SERIAL  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_SPICEVMC\_LAST  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_SPICEVMC\_SMARTCARD  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_SPICEVMC\_USBREDIR  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_SPICEVMC\_VDAGENT  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_LAST  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_RAW  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TELNET  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TELNETS  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TCP\_PROTOCOL\_TLS  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_DEV  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_FILE  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_LAST  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_NULL  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_PIPE  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_PTY  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_SPICEVMC  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_STDIO  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_TCP  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_UDP  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_UNIX  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CHR\_TYPE\_VC  
domain\_conf.h, [271](#)
- VIR\_DOMAIN\_CLOCK\_BASIS\_LAST  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CLOCK\_BASIS\_LOCALTIME  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CLOCK\_BASIS\_UTC  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_LAST  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_LOCALTIME  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_TIMEZONE  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_UTC  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CLOCK\_OFFSET\_VARIABLE  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONSOLE\_FORCE  
libvirt.h, [136](#)
- VIR\_DOMAIN\_CONSOLE\_SAFE  
libvirt.h, [136](#)
- VIR\_DOMAIN\_CONTROL\_ERROR  
libvirt.h, [136](#)

- VIR\_DOMAIN\_CONTROL\_JOB  
libvirt.h, [136](#)
- VIR\_DOMAIN\_CONTROL\_OCCUPIED  
libvirt.h, [136](#)
- VIR\_DOMAIN\_CONTROL\_OK  
libvirt.h, [136](#)
- VIR\_DOMAIN\_CONTROLLER\_MASTER\_LAST  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MASTER\_NONE  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MASTER\_USB  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_AUTO  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_BUSLOGIC  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_IBMVSCSI  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_LAST  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_LSILOGIC  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_LSI5068  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_VIRTIO\_SCSI  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_SCSI\_VMPVSCSI  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_EHCI  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_ICH9\_EHCI1  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_ICH9\_UHCI1  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_ICH9\_UHCI2  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_ICH9\_UHCI3  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_LAST  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_NEC\_XHCI  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_NONE  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_PCI\_OHCI  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_PIIX3\_UHCI  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_PIIX4\_UHCI  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_MODEL\_USB\_VT82C686B\_UHCI  
domain\_conf.h, [272](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_CCID  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_FDC  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_IDE  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_LAST  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_SATA  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_SCSI  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_USB  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CONTROLLER\_TYPE\_VIRTIO\_SERIAL  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_AUTO  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_LAST  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CPU\_PLACEMENT\_MODE\_STATIC  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_CRASHED  
libvirt.h, [146](#)
- VIR\_DOMAIN\_CRASHED\_UNKNOWN  
libvirt.h, [137](#)
- VIR\_DOMAIN\_DESTROY\_DEFAULT  
libvirt.h, [137](#)
- VIR\_DOMAIN\_DESTROY\_GRACEFUL  
libvirt.h, [137](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_CCID  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_DRIVE  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_LAST  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_NONE  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_PCI  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_SPAPR\_VIO  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_USB  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_VIRTIO\_S390  
domain\_conf.h, [273](#)

- VIR\_DOMAIN\_DEVICE\_ADDRESS\_TYPE\_VIRTIO\_S-  
ERIAL  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_CHR  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_CONTROLLER  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_DISK  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_FS  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_GRAPHICS  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_HOSTDEV  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_HUB  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_INPUT  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_LAST  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_LEASE  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_MEMBALLOON  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_MODIFY\_CONFIG  
libvirt.h, [137](#)
- VIR\_DOMAIN\_DEVICE\_MODIFY\_CURRENT  
libvirt.h, [137](#)
- VIR\_DOMAIN\_DEVICE\_MODIFY\_FORCE  
libvirt.h, [137](#)
- VIR\_DOMAIN\_DEVICE\_MODIFY\_LIVE  
libvirt.h, [137](#)
- VIR\_DOMAIN\_DEVICE\_NET  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_NONE  
domain\_conf.h, [273](#)
- VIR\_DOMAIN\_DEVICE\_REDIRECTED  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_SMARTCARD  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_SOUND  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_VIDEO  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DEVICE\_WATCHDOG  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_FDC  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_IDE  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_LAST  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_SATA  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_SCSI  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_UML  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_USB  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_VIRTIO  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_BUS\_XEN  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_DEFAULT  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_DIRECTSYNC  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_DISABLE  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_LAST  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_UNSAFE  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_WRITEBACK  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_CACHE\_WRITETHRU  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_DEFAULT  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_LAST  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_OFF  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_COPY\_ON\_READ\_ON  
domain\_conf.h, [274](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_CDROM  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_DISK  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_FLOPPY  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_LAST  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_DEVICE\_LUN  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_NO\_SPACE  
libvirt.h, [138](#)
- VIR\_DOMAIN\_DISK\_ERROR\_NONE  
libvirt.h, [138](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_DEFAULT  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_ENOSPACE  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_IGNORE  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_LAST  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_REPORT  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_POLICY\_STOP  
domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_ERROR\_UNSPEC  
libvirt.h, [138](#)
- VIR\_DOMAIN\_DISK\_IO\_DEFAULT



- domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_IO\_LAST
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_IO\_NATIVE
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_IO\_THREADS
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_PROTOCOL\_LAST
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_PROTOCOL\_NBD
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_PROTOCOL\_RBD
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_PROTOCOL\_SHEEPDOG
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_LAST
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_NONE
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_USAGE
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_SECRET\_TYPE\_UUID
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_TRANS\_AUTO
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_TRANS\_DEFAULT
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_TRANS\_LAST
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_TRANS\_LBA
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_TRANS\_NONE
  - domain\_conf.h, [275](#)
- VIR\_DOMAIN\_DISK\_TRAY\_CLOSED
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_TRAY\_LAST
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_TRAY\_OPEN
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_BLOCK
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_DIR
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_FILE
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_LAST
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_DISK\_TYPE\_NETWORK
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_EVENT\_DEFINED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_DEFINED\_ADDED
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_DEFINED\_UPDATED
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_DISK\_CHANGE\_MISSING\_ON\_START
  - libvirt.h, [133](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_IPV4
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_IPV6
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_ADDRESS\_UNIX
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_CONNECT
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_DISCONNECT
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_GRAPHICS\_INITIALIZE
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_ID\_BALLOON\_CHANGE
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_BLOCK\_JOB
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_CONTROL\_ERROR
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_DISK\_CHANGE
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_GRAPHICS
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_IO\_ERROR
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_ID\_IO\_ERROR\_REASON
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_LIFECYCLE
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_ID\_PMSUSPEND
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_PMWAKEUP
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_REBOOT
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_ID\_RTC\_CHANGE
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_ID\_TRAY\_CHANGE
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_ID\_WATCHDOG
  - libvirt.h, [138](#)
- VIR\_DOMAIN\_EVENT\_IO\_ERROR\_NONE
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_IO\_ERROR\_PAUSE
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_IO\_ERROR\_REPORT
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_PMSUSPENDED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_PMSUSPENDED\_MEMORY
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_RESUMED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_RESUMED\_FROM\_SNAPSHOT
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_RESUMED\_MIGRATED
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_RESUMED\_UNPAUSED



- libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_SHUTDOWN
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_SHUTDOWN\_FINISHED
  - libvirt.h, [139](#)
- VIR\_DOMAIN\_EVENT\_STARTED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_STARTED\_BOOTED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STARTED\_FROM\_SNAPSHOT
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STARTED\_MIGRATED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STARTED\_RESTORED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STARTED\_WAKEUP
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STOPPED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_STOPPED\_CRASHED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STOPPED\_DESTROYED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STOPPED\_FAILED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STOPPED\_FROM\_SNAPSHOT
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STOPPED\_MIGRATED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STOPPED\_SAVED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_STOPPED\_SHUTDOWN
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_SUSPENDED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_FROM\_SNAPSHOT
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_IOERROR
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_MIGRATED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_PAUSED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_RESTORED
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_SUSPENDED\_WATCHDOG
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_TRAY\_CHANGE\_CLOSE
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_TRAY\_CHANGE\_OPEN
  - libvirt.h, [140](#)
- VIR\_DOMAIN\_EVENT\_UNDEFINED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_UNDEFINED\_REMOVED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_DEBUG
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_NONE
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_PAUSE
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_POWEROFF
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_RESET
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_EVENT\_WATCHDOG\_SHUTDOWN
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_FEATURE\_ACPI
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FEATURE\_APIC
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FEATURE\_HAP
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FEATURE\_LAST
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FEATURE\_PAE
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FEATURE\_PRIVNET
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FEATURE\_VIRIDIAN
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FS\_ACCESSMODE\_LAST
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FS\_ACCESSMODE\_MAPPED
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FS\_ACCESSMODE\_PASSTHROUGH
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FS\_ACCESSMODE\_SQUASH
  - domain\_conf.h, [276](#)
- VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_DEFAULT
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_HANDLE
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_LAST
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_DRIVER\_TYPE\_PATH
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_BIND
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_BLOCK
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_FILE
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_LAST
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_MOUNT
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_RAM
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_TYPE\_TEMPLATE
  - domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_WRPOLICY\_DEFAULT
  - domain\_conf.h, [277](#)

- VIR\_DOMAIN\_FS\_WRPOLICY\_IMMEDIATE  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_FS\_WRPOLICY\_LAST  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_DEFAULT  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_DISCONNECT  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_FAIL  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_KEEP  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_AUTH\_CONNECTED\_LAST  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_ADDRESS  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_LAST  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_NETWORK  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_LISTEN\_TYPE\_NONE  
domain\_conf.h, [277](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_CURSOR  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_DISPLAY  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_INPUT  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_LAST  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MAIN  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_ANY  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_INSECURE  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_LAST  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_MODE\_SECURE  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_PLAYBACK  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_RECORD  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_SMARTCARD  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CHANNEL\_USBEDIR  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_DEFAULT  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_LAST  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_NO  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_CLIPBOARD\_COPYPASTE\_YES  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_AUTO\_GLZ  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_AUTO\_LZ  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_DEFAULT  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_GLZ  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_LAST  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_LZ  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_OFF  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_IMAGE\_COMPRESSION\_QUIC  
domain\_conf.h, [278](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_ALWAYS  
domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_AUTO  
domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_DEFAULT  
domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_LAST  
domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_JPEG\_COMPRESSION\_NEVER  
domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_

- CLIENT
- domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_DEFAULT
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_LAST
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_MOUSE\_MODE\_SERVER
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_DEFAULT
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_LAST
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_OFF
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_PLAYBACK\_COMPRESSION\_ON
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_ALL
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_DEFAULT
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_FILTER
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_LAST
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_STREAMING\_MODE\_OFF
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_ALWAYS
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_AUTO
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_DEFAULT
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_LAST
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_SPICE\_ZLIB\_COMPRESSION\_NEVER
  - domain\_conf.h, [279](#)
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_DESKTOP
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_LAST
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_RDP
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_SDL
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_SPICE
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_GRAPHICS\_TYPE\_VNC
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_HOSTDEV\_MODE\_CAPABILITIES
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_HOSTDEV\_MODE\_LAST
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_HOSTDEV\_MODE\_SUBSYS
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_LAST
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_PCI
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_HOSTDEV\_SUBSYS\_TYPE\_USB
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_HUB\_TYPE\_LAST
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_HUB\_TYPE\_USB
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_INPUT\_BUS\_LAST
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_INPUT\_BUS\_PS2
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_INPUT\_BUS\_USB
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_INPUT\_BUS\_XEN
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_INPUT\_TYPE\_LAST
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_INPUT\_TYPE\_MOUSE
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_INPUT\_TYPE\_TABLET
  - domain\_conf.h, [280](#)
- VIR\_DOMAIN\_IO\_EVENT\_FD\_DEFAULT
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_IO\_EVENT\_FD\_LAST
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_IO\_EVENT\_FD\_OFF
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_IO\_EVENT\_FD\_ON
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_JOB\_BOUNDED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_JOB\_CANCELLED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_JOB\_COMPLETED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_JOB\_FAILED
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_JOB\_NONE
  - libvirt.h, [141](#)
- VIR\_DOMAIN\_JOB\_UNBOUNDED
  - libvirt.h, [141](#)

- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_COREDUMP\_DESTROY
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_COREDUMP\_RESTART
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_DESTROY
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_LAST
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_PRESERVE
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_RESTART
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_CRASH\_RESTART\_RENAME
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_DESTROY
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_LAST
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_PRESERVE
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_RESTART
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_LIFECYCLE\_RESTART\_RENAME
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_MEM\_CONFIG
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEM\_CURRENT
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEM\_DUMP\_DEFAULT
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_MEM\_DUMP\_LAST
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_MEM\_DUMP\_OFF
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_MEM\_DUMP\_ON
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_MEM\_LIVE
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEM\_MAXIMUM
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMBALLOON\_MODEL\_LAST
  - domain\_conf.h, [269](#)
- VIR\_DOMAIN\_MEMBALLOON\_MODEL\_NONE
  - domain\_conf.h, [269](#)
- VIR\_DOMAIN\_MEMBALLOON\_MODEL\_VIRTIO
  - domain\_conf.h, [269](#)
- VIR\_DOMAIN\_MEMBALLOON\_MODEL\_XEN
  - domain\_conf.h, [269](#)
- VIR\_DOMAIN\_MEMORY\_PARAM\_BOOLEAN
  - libvirt.h, [148](#)
- VIR\_DOMAIN\_MEMORY\_PARAM\_DOUBLE
  - libvirt.h, [148](#)
- VIR\_DOMAIN\_MEMORY\_PARAM\_INT
  - libvirt.h, [148](#)
- VIR\_DOMAIN\_MEMORY\_PARAM\_LLONG
  - libvirt.h, [148](#)
- VIR\_DOMAIN\_MEMORY\_PARAM\_UINT
  - libvirt.h, [148](#)
- VIR\_DOMAIN\_MEMORY\_PARAM\_ULLONG
  - libvirt.h, [148](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_ACTUAL\_BALLOON
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_AVAILABLE
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_MAJOR\_FAULT
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_MINOR\_FAULT
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_NR
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_RSS
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_SWAP\_IN
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_SWAP\_OUT
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_MEMORY\_STAT\_UNUSED
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_METADATA\_DESCRIPTION
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_METADATA\_ELEMENT
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_METADATA\_TITLE
  - libvirt.h, [142](#)
- VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_DEFAULT
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_LAST
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_QEMU
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_NET\_BACKEND\_TYPE\_VHOST
  - domain\_conf.h, [281](#)
- VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_DEFAULT
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_DOWN
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_LAST
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_INTERFACE\_LINK\_STATE\_UP
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_BRIDGE
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_CLIENT
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_DIRECT
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_ETHERNET
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_HOSTDEV
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_INTERNAL

- domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_LAST
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_MCAST
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_NETWORK
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_SERVER
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_TYPE\_USER
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_DEFAULT
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_IOTHREAD
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_LAST
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NET\_VIRTIO\_TX\_MODE\_TIMER
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NONE
  - libvirt.h, [137](#)
- VIR\_DOMAIN\_NOSTATE
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_NOSTATE\_UNKNOWN
  - libvirt.h, [143](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_INTERLEAVE
  - libvirt.h, [143](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_AUTO
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_DEFAULT
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_LAST
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PLACEMENT\_MODE\_STATIC
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_PREFERRED
  - libvirt.h, [143](#)
- VIR\_DOMAIN\_NUMATUNE\_MEM\_STRICT
  - libvirt.h, [143](#)
- VIR\_DOMAIN\_OPEN\_GRAPHICS\_SKIPAUTH
  - libvirt.h, [143](#)
- VIR\_DOMAIN\_PAUSED
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_PAUSED\_DUMP
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_PAUSED\_FROM\_SNAPSHOT
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_PAUSED\_IOERROR
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_PAUSED\_MIGRATION
  - libvirt.h, [143](#)
- VIR\_DOMAIN\_PAUSED\_SAVE
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_PAUSED\_SHUTTING\_DOWN
  - libvirt.h, [144](#)
- libvirt.h, [144](#)
- VIR\_DOMAIN\_PAUSED\_UNKNOWN
  - libvirt.h, [143](#)
- VIR\_DOMAIN\_PAUSED\_USER
  - libvirt.h, [143](#)
- VIR\_DOMAIN\_PAUSED\_WATCHDOG
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_PCI\_ROMBAR\_DEFAULT
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_PCI\_ROMBAR\_LAST
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_PCI\_ROMBAR\_OFF
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_PCI\_ROMBAR\_ON
  - domain\_conf.h, [282](#)
- VIR\_DOMAIN\_PM\_STATE\_DEFAULT
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_PM\_STATE\_DISABLED
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_PM\_STATE\_ENABLED
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_PM\_STATE\_LAST
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_PMSUSPENDED
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_PMSUSPENDED\_UNKNOWN
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_REBOOT\_ACPI\_POWER\_BTN
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_REBOOT\_DEFAULT
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_REBOOT\_GUEST\_AGENT
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_REDIRDEV\_BUS\_LAST
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_REDIRDEV\_BUS\_USB
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_RUNNING
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_RUNNING\_BOOTED
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_RUNNING\_FROM\_SNAPSHOT
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_RUNNING\_MIGRATED
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_RUNNING\_MIGRATION\_CANCELED
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_RUNNING\_RESTORED
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_RUNNING\_SAVE\_CANCELED
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_RUNNING\_UNKNOWN
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_RUNNING\_UNPAUSED
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_RUNNING\_WAKEUP
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_SAVE\_BYPASS\_CACHE

- libvirt.h, [144](#)
- VIR\_DOMAIN\_SAVE\_PAUSED
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_SAVE\_RUNNING
  - libvirt.h, [144](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_BOOLEAN
  - libvirt.h, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_DOUBLE
  - libvirt.h, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_INT
  - libvirt.h, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_LLONG
  - libvirt.h, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_UINT
  - libvirt.h, [150](#)
- VIR\_DOMAIN\_SCHED\_FIELD\_ULLONG
  - libvirt.h, [150](#)
- VIR\_DOMAIN\_SECLABEL\_DEFAULT
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SECLABEL\_DYNAMIC
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SECLABEL\_LAST
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SECLABEL\_NONE
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SECLABEL\_STATIC
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SHUTDOWN
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SHUTDOWN\_ACPI\_POWER\_BTN
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTDOWN\_DEFAULT
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTDOWN\_GUEST\_AGENT
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTDOWN\_UNKNOWN
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTDOWN\_USER
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTOFF
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SHUTOFF\_CRASHED
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_DESTROYED
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_FAILED
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_FROM\_SNAPSHOT
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_MIGRATED
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_SAVED
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_SHUTDOWN
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SHUTOFF\_UNKNOWN
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SMARTCARD\_TYPE\_HOST
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SMARTCARD\_TYPE\_HOST\_CERTIFICATES
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SMARTCARD\_TYPE\_LAST
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SMARTCARD\_TYPE\_PASSTHROUGH
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SMBIOS\_EMULATE
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SMBIOS\_HOST
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SMBIOS\_LAST
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SMBIOS\_NONE
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SMBIOS\_SYSINFO
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_ATOMIC
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_CURRENT
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_DISK\_ONLY
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_HALT
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_NO\_METADATA
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_QUIESCE
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_REDEFINE
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_CREATE\_REUSE\_EXT
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_DELETE\_CHILDREN
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_DELETE\_CHILDREN\_ONLY
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_DELETE\_METADATA\_ONLY
  - libvirt.h, [145](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_DESCENDANTS
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_LEAVES
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_METADATA
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_NO\_LEAVES
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_NO\_METADATA
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_LIST\_ROOTS
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_REVERT\_FORCE
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_REVERT\_PAUSED

- libvirt.h, [146](#)
- VIR\_DOMAIN\_SNAPSHOT\_REVERT\_RUNNING
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_DUPLEX
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_LAST
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SOUND\_CODEC\_TYPE\_MICRO
  - domain\_conf.h, [283](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_AC97
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_ES1370
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_ICH6
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_LAST
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_PCSPK
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_SOUND\_MODEL\_SB16
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_START\_AUTODESTROY
  - libvirt.h, [137](#)
- VIR\_DOMAIN\_START\_BYPASS\_CACHE
  - libvirt.h, [137](#)
- VIR\_DOMAIN\_START\_FORCE\_BOOT
  - libvirt.h, [137](#)
- VIR\_DOMAIN\_START\_PAUSED
  - libvirt.h, [137](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_DEFAULT
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_LAST
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_MANDATORY
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_OPTIONAL
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_STARTUP\_POLICY\_REQUISITE
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TAINT\_CUSTOM\_ARGV
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TAINT\_CUSTOM\_MONITOR
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TAINT\_DISK\_PROBING
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TAINT\_EXTERNAL\_LAUNCH
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TAINT\_HIGH\_PRIVILEGES
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TAINT\_HOST\_CPU
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TAINT\_LAST
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TAINT\_SHELL\_SCRIPTS
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_AUTO
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_EMULATE
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_LAST
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_NATIVE
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_PARAVIRT
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TIMER\_MODE\_SMPSAFE
  - domain\_conf.h, [284](#)
- VIR\_DOMAIN\_TIMER\_NAME\_HPET
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_KVMCLOCK
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_LAST
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_PIT
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_PLATFORM
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_RTC
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_NAME\_TSC
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_CATCHUP
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_DELAY
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_DISCARD
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_LAST
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TICKPOLICY\_MERGE
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TRACK\_BOOT
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TRACK\_GUEST
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TRACK\_LAST
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_TIMER\_TRACK\_WALL
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_UNDEFINE\_MANAGED\_SAVE
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_UNDEFINE\_SNAPSHOTS\_METADATA
  - libvirt.h, [146](#)
- VIR\_DOMAIN\_VCPU\_CONFIG
  - libvirt.h, [147](#)
- VIR\_DOMAIN\_VCPU\_CURRENT
  - libvirt.h, [147](#)
- VIR\_DOMAIN\_VCPU\_LIVE
  - libvirt.h, [147](#)
- VIR\_DOMAIN\_VCPU\_MAXIMUM
  - libvirt.h, [147](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_CIRRUS
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_LAST
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_QXL



- domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_VBOX
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_VGA
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_VMVGA
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIDEO\_TYPE\_XEN
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIRT\_HYPERV
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_KQEMU
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_KVM
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_LAST
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_LXC
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_OPENVZ
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_PARALLELS
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_PHPY
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_QEMU
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_TEST
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_UML
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_VBOX
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_VMWARE
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRT\_XEN
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_VIRTIO\_EVENT\_IDX\_DEFAULT
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIRTIO\_EVENT\_IDX\_LAST
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIRTIO\_EVENT\_IDX\_OFF
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_VIRTIO\_EVENT\_IDX\_ON
  - domain\_conf.h, [285](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_DUMP
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_LAST
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_NONE
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_PAUSE
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_POWEROFF
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_RESET
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_ACTION\_SHUTDOWN
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_MODEL\_I6300ESB
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_MODEL\_IB700
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_WATCHDOG\_MODEL\_LAST
  - domain\_conf.h, [286](#)
- VIR\_DOMAIN\_XML\_INACTIVE
  - libvirt.h, [147](#)
- VIR\_DOMAIN\_XML\_INTERNAL\_ACTUAL\_NET
  - domain\_conf.c, [236](#)
- VIR\_DOMAIN\_XML\_INTERNAL\_ALLOW\_BOOT
  - domain\_conf.c, [236](#)
- VIR\_DOMAIN\_XML\_INTERNAL\_ALLOW\_ROM
  - domain\_conf.c, [236](#)
- VIR\_DOMAIN\_XML\_INTERNAL\_PCI\_ORIG\_STATES
  - domain\_conf.c, [236](#)
- VIR\_DOMAIN\_XML\_INTERNAL\_STATUS
  - domain\_conf.c, [236](#)
- VIR\_DOMAIN\_XML\_SECURE
  - libvirt.h, [147](#)
- VIR\_DOMAIN\_XML\_UPDATE\_CPU
  - libvirt.h, [147](#)
- VIR\_DRV\_ESX
  - driver.h, [328](#)
- VIR\_DRV\_HYPERV
  - driver.h, [329](#)
- VIR\_DRV\_LIBXL
  - driver.h, [328](#)
- VIR\_DRV\_LXC
  - driver.h, [328](#)
- VIR\_DRV\_ONE
  - driver.h, [328](#)
- VIR\_DRV\_OPEN\_DECLINED
  - driver.h, [329](#)
- VIR\_DRV\_OPEN\_ERROR
  - driver.h, [329](#)
- VIR\_DRV\_OPEN\_SUCCESS
  - driver.h, [329](#)
- VIR\_DRV\_OPENVZ
  - driver.h, [328](#)
- VIR\_DRV\_PARALLELS
  - driver.h, [329](#)
- VIR\_DRV\_PHPY
  - driver.h, [328](#)
- VIR\_DRV\_QEMU
  - driver.h, [328](#)
- VIR\_DRV\_REMOTE
  - driver.h, [328](#)
- VIR\_DRV\_TEST
  - driver.h, [328](#)
- VIR\_DRV\_UML
  - driver.h, [328](#)
- VIR\_DRV\_VBOX
  - driver.h, [328](#)
- VIR\_DRV\_VMWARE
  - driver.h, [328](#)
- VIR\_DRV\_XEN\_UNIFIED



- driver.h, [328](#)
- VIR\_DRV\_XENAPI
  - driver.h, [328](#)
- VIR\_DUMP\_BYPASS\_CACHE
  - libvirt.h, [137](#)
- VIR\_DUMP\_CRASH
  - libvirt.h, [137](#)
- VIR\_DUMP\_LIVE
  - libvirt.h, [137](#)
- VIR\_DUMP\_MEMORY\_ONLY
  - libvirt.h, [137](#)
- VIR\_DUMP\_RESET
  - libvirt.h, [137](#)
- VIR\_EVENT\_HANDLE\_ERROR
  - libvirt.h, [147](#)
- VIR\_EVENT\_HANDLE\_HANGUP
  - libvirt.h, [147](#)
- VIR\_EVENT\_HANDLE\_READABLE
  - libvirt.h, [147](#)
- VIR\_EVENT\_HANDLE\_WRITABLE
  - libvirt.h, [147](#)
- VIR\_INTERFACE\_XML\_INACTIVE
  - libvirt.h, [147](#)
- VIR\_KEYCODE\_SET\_ATSET1
  - libvirt.h, [147](#)
- VIR\_KEYCODE\_SET\_ATSET2
  - libvirt.h, [147](#)
- VIR\_KEYCODE\_SET\_ATSET3
  - libvirt.h, [147](#)
- VIR\_KEYCODE\_SET\_LINUX
  - libvirt.h, [147](#)
- VIR\_KEYCODE\_SET\_OSX
  - libvirt.h, [148](#)
- VIR\_KEYCODE\_SET\_RFB
  - libvirt.h, [148](#)
- VIR\_KEYCODE\_SET\_USB
  - libvirt.h, [148](#)
- VIR\_KEYCODE\_SET\_WIN32
  - libvirt.h, [148](#)
- VIR\_KEYCODE\_SET\_XT
  - libvirt.h, [147](#)
- VIR\_KEYCODE\_SET\_XT\_KBD
  - libvirt.h, [148](#)
- VIR\_MEMORY\_PHYSICAL
  - libvirt.h, [142](#)
- VIR\_MEMORY\_VIRTUAL
  - libvirt.h, [142](#)
- VIR\_MIGRATE\_CHANGE\_PROTECTION
  - libvirt.h, [143](#)
- VIR\_MIGRATE\_LIVE
  - libvirt.h, [142](#)
- VIR\_MIGRATE\_NON\_SHARED\_DISK
  - libvirt.h, [142](#)
- VIR\_MIGRATE\_NON\_SHARED\_INC
  - libvirt.h, [143](#)
- VIR\_MIGRATE\_PAUSED
  - libvirt.h, [142](#)
- VIR\_MIGRATE\_PEER2PEER
  - libvirt.h, [142](#)
- VIR\_MIGRATE\_PERSIST\_DEST
  - libvirt.h, [142](#)
- VIR\_MIGRATE\_TUNNELLED
  - libvirt.h, [142](#)
- VIR\_MIGRATE\_UNDEFINE\_SOURCE
  - libvirt.h, [142](#)
- VIR\_MIGRATE\_UNSAFE
  - libvirt.h, [143](#)
- VIR\_NETDEV\_TAP\_CREATE\_IFUP
  - virnetdevtap.h, [447](#)
- VIR\_NETDEV\_TAP\_CREATE\_NONE
  - virnetdevtap.h, [447](#)
- VIR\_NETDEV\_TAP\_CREATE\_PERSIST
  - virnetdevtap.h, [448](#)
- VIR\_NETDEV\_TAP\_CREATE\_USE\_MAC\_FOR\_BRIDGE
  - virnetdevtap.h, [448](#)
- VIR\_NETDEV\_TAP\_CREATE\_VNET\_HDR
  - virnetdevtap.h, [447](#)
- VIR\_NETWORK\_FORWARD\_BRIDGE
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_HOSTDEV
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_HOSTDEV\_DEVICE\_LAST
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_HOSTDEV\_DEVICE\_NETWORKDEV
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_HOSTDEV\_DEVICE\_NETWORKDEV\_ONE
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_HOSTDEV\_DEVICE\_PRIVATE
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_LAST
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_NAT
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_NONE
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_PASSTHROUGH
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_PRIVATE
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_ROUTE
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_FORWARD\_VEPA
  - network\_conf.h, [305](#)
- VIR\_NETWORK\_SECTION\_BRIDGE
  - libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_DNS\_HOST
  - libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_DNS\_SRV
  - libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_DNS\_TXT
  - libvirt.h, [149](#)

- VIR\_NETWORK\_SECTION\_DOMAIN  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_FORWARD  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_FORWARD\_INTERFACE  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_FORWARD\_PF  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_IP  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_IP\_DHCP\_HOST  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_IP\_DHCP\_RANGE  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_NONE  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_PORTGROUP  
libvirt.h, [149](#)
- VIR\_NETWORK\_SECTION\_TUNNEL  
libvirt.h, [149](#)
- VIR\_NETWORK\_UPDATE\_AFFECT\_CONFIG  
libvirt.h, [148](#)
- VIR\_NETWORK\_UPDATE\_AFFECT\_CURRENT  
libvirt.h, [148](#)
- VIR\_NETWORK\_UPDATE\_AFFECT\_LIVE  
libvirt.h, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_ADD\_FIRST  
libvirt.h, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_ADD\_LAST  
libvirt.h, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_DELETE  
libvirt.h, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_MODIFY  
libvirt.h, [148](#)
- VIR\_NETWORK\_UPDATE\_COMMAND\_NONE  
libvirt.h, [148](#)
- VIR\_NETWORK\_XML\_INACTIVE  
libvirt.h, [149](#)
- VIR\_NODE\_CPU\_STATS\_ALL\_CPUS  
libvirt.h, [149](#)
- VIR\_NODE\_MEMORY\_STATS\_ALL\_CELLS  
libvirt.h, [149](#)
- VIR\_NODE\_SUSPEND\_TARGET\_DISK  
libvirt.h, [150](#)
- VIR\_NODE\_SUSPEND\_TARGET\_HYBRID  
libvirt.h, [150](#)
- VIR\_NODE\_SUSPEND\_TARGET\_MEM  
libvirt.h, [150](#)
- VIR\_SECRET\_GET\_VALUE\_INTERNAL\_CALL  
driver.h, [328](#)
- VIR\_SECRET\_USAGE\_TYPE\_CEPH  
libvirt.h, [150](#)
- VIR\_SECRET\_USAGE\_TYPE\_NONE  
libvirt.h, [150](#)
- VIR\_SECRET\_USAGE\_TYPE\_VOLUME  
libvirt.h, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_NEW  
libvirt.h, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_NO\_OVERWRITE  
libvirt.h, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_OVERWRITE  
libvirt.h, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_REPAIR  
libvirt.h, [150](#)
- VIR\_STORAGE\_POOL\_BUILD\_RESIZE  
libvirt.h, [150](#)
- VIR\_STORAGE\_POOL\_BUILDING  
libvirt.h, [151](#)
- VIR\_STORAGE\_POOL\_DEGRADED  
libvirt.h, [151](#)
- VIR\_STORAGE\_POOL\_DELETE\_NORMAL  
libvirt.h, [150](#)
- VIR\_STORAGE\_POOL\_DELETE\_ZEROED  
libvirt.h, [150](#)
- VIR\_STORAGE\_POOL\_INACCESSIBLE  
libvirt.h, [151](#)
- VIR\_STORAGE\_POOL\_INACTIVE  
libvirt.h, [151](#)
- VIR\_STORAGE\_POOL\_RUNNING  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_BLOCK  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_DELETE\_NORMAL  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_DELETE\_ZEROED  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_DIR  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_FILE  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_NETWORK  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_RESIZE\_ALLOCATE  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_RESIZE\_DELTA  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_RESIZE\_SHRINK  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_BSI  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_DOD  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_GUTMANN  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_NNSA  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_PFITZNER33  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_PFITZNER7  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_RANDOM  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_SCHNEIER  
libvirt.h, [151](#)
- VIR\_STORAGE\_VOL\_WIPE\_ALG\_ZERO  
libvirt.h, [151](#)

- VIR\_STORAGE\_XML\_INACTIVE
  - libvirt.h, [152](#)
- VIR\_STREAM\_EVENT\_ERROR
  - libvirt.h, [152](#)
- VIR\_STREAM\_EVENT\_HANGUP
  - libvirt.h, [152](#)
- VIR\_STREAM\_EVENT\_READABLE
  - libvirt.h, [152](#)
- VIR\_STREAM\_EVENT\_WRITABLE
  - libvirt.h, [152](#)
- VIR\_STREAM\_NONBLOCK
  - libvirt.h, [152](#)
- VIR\_TYPED\_PARAM\_BOOLEAN
  - libvirt.h, [152](#)
- VIR\_TYPED\_PARAM\_DOUBLE
  - libvirt.h, [152](#)
- VIR\_TYPED\_PARAM\_INT
  - libvirt.h, [152](#)
- VIR\_TYPED\_PARAM\_LLONG
  - libvirt.h, [152](#)
- VIR\_TYPED\_PARAM\_STRING
  - libvirt.h, [152](#)
- VIR\_TYPED\_PARAM\_STRING\_OKAY
  - libvirt.h, [152](#)
- VIR\_TYPED\_PARAM\_UINT
  - libvirt.h, [152](#)
- VIR\_TYPED\_PARAM\_ULLONG
  - libvirt.h, [152](#)
- VIR\_VCPU\_BLOCKED
  - libvirt.h, [153](#)
- VIR\_VCPU\_OFFLINE
  - libvirt.h, [153](#)
- VIR\_VCPU\_RUNNING
  - libvirt.h, [153](#)
- VIR\_ARG15
  - libvirt.c, [340](#)
- VIR\_COPY\_CPUMAP
  - libvirt.h, [111](#)
- VIR\_CPU\_MAPLEN
  - libvirt.h, [111](#)
- VIR\_CPU\_USABLE
  - libvirt.h, [111](#)
- VIR\_DEPRECATED
  - libvirt.h, [112](#)
- VIR\_DOMAIN\_DEBUG
  - libvirt.c, [340](#)
- VIR\_DOMAIN\_DEBUG\_0
  - libvirt.c, [341](#)
- VIR\_DOMAIN\_DEBUG\_1
  - libvirt.c, [341](#)
- VIR\_DOMAIN\_DEBUG\_2
  - libvirt.c, [341](#)
- VIR\_ENUM\_IMPL
  - domain\_conf.c, [236](#)
  - network\_conf.c, [295](#)
  - qemu\_command.c, [437](#)
  - virsh-network.c, [451](#)
- VIR\_EXPORT\_VAR
  - libvirt.h, [117](#)
- VIR\_FROM\_THIS
  - bridge\_driver.c, [424](#)
  - domain\_conf.c, [236](#)
  - libvirt.c, [341](#)
  - network\_conf.c, [295](#)
  - qemu\_command.c, [433](#)
  - uml\_conf.c, [438](#)
  - virnetdevbridge.c, [440](#)
  - virnetdevopenvswitch.c, [442](#)
  - virnetdevtap.c, [446](#)
- VIR\_GET\_CPUMAP
  - libvirt.h, [117](#)
- VIR\_HAS\_COMMA
  - libvirt.c, [341](#)
- VIR\_UNUSE\_CPU
  - libvirt.h, [120](#)
- VIR\_USE\_CPU
  - libvirt.h, [120](#)
- VIR\_UUID\_BUFLen
  - libvirt.h, [120](#)
- VIR\_UUID\_DEBUG
  - libvirt.c, [341](#)
- val
  - \_virDomainMemoryStat, [43](#)
- validTo
  - \_virDomainGraphicsAuthDef, [34](#)
- value
  - \_virNetworkDNSTxtRecordsDef, [72](#)
  - \_virNodeCPUStats, [77](#)
  - \_virNodeMemoryStats, [78](#)
  - \_virTypedParameter, [89](#)
- variable
  - \_virDomainClockDef, [16](#)
- vcpuId
  - \_virDomainVcpuPinDef, [54](#)
- vcpuPin
  - \_virDomainDef, [22](#)
- vcpus
  - \_virDomainDef, [22](#)
- vectors
  - \_virDomainVirtioSerialOpts, [55](#)
- vendor
  - \_virDomainHostdevSubsys, [39](#)
  - \_virDomainRedirFilterUsbDevDef, [50](#)
- verify
  - domain\_conf.c, [236](#)
- version
  - \_virDomainRedirFilterUsbDevDef, [50](#)
  - \_virDriver, [66](#)
- video
  - \_virDomainDeviceDef, [24](#)
- videos
  - \_virDomainDef, [22](#)
- vioserial
  - \_virDomainControllerDef, [17](#)
  - \_virDomainDeviceInfo, [26](#)
- virBlkioDeviceWeight

- domain\_conf.h, 265
- virBlkioDeviceWeightArrayClear
  - domain\_conf.c, 236
  - domain\_conf.h, 286
- virBlkioDeviceWeightPtr
  - domain\_conf.h, 265
- virBlkioParameter
  - libvirt.h, 120
- virBlkioParameterPtr
  - libvirt.h, 120
- virBlkioParameterType
  - libvirt.h, 132
- virCPUCompareResult
  - libvirt.h, 135
- virConnect
  - libvirt.h, 120
- virConnectAuth
  - libvirt.h, 120
- virConnectAuthCallbackDefault
  - libvirt.c, 343
- virConnectAuthCallbackPtr
  - libvirt.h, 121
- virConnectAuthDefault
  - libvirt.c, 419
- virConnectAuthPtr
  - libvirt.h, 121
- virConnectAuthPtrDefault
  - libvirt.c, 419
  - libvirt.h, 227
- virConnectBaselineCPU
  - libvirt.c, 343
  - libvirt.h, 153
- virConnectClose
  - libvirt.c, 343
  - libvirt.h, 153
- virConnectCloseFunc
  - libvirt.h, 121
- virConnectCloseReason
  - libvirt.h, 132
- virConnectCompareCPU
  - libvirt.c, 343
  - libvirt.h, 153
- virConnectCredTypeDefault
  - libvirt.c, 419
- virConnectCredential
  - libvirt.h, 121
- virConnectCredentialPtr
  - libvirt.h, 121
- virConnectCredentialType
  - libvirt.h, 132
- virConnectDomainEventBalloonChangeCallback
  - libvirt.h, 121
- virConnectDomainEventBlockJobCallback
  - libvirt.h, 121
- virConnectDomainEventBlockJobStatus
  - libvirt.h, 132
- virConnectDomainEventCallback
  - libvirt.h, 121
- virConnectDomainEventDeregister
  - libvirt.c, 343
  - libvirt.h, 154
- virConnectDomainEventDeregisterAny
  - libvirt.c, 344
  - libvirt.h, 154
- virConnectDomainEventDiskChangeCallback
  - libvirt.h, 121
- virConnectDomainEventDiskChangeReason
  - libvirt.h, 132
- virConnectDomainEventGenericCallback
  - libvirt.h, 122
- virConnectDomainEventGraphicsCallback
  - libvirt.h, 122
- virConnectDomainEventIOErrorCallback
  - libvirt.h, 122
- virConnectDomainEventIOErrorReasonCallback
  - libvirt.h, 122
- virConnectDomainEventPMSuspendCallback
  - libvirt.h, 122
- virConnectDomainEventPMWakeupCallback
  - libvirt.h, 122
- virConnectDomainEventRTCChangeCallback
  - libvirt.h, 122
- virConnectDomainEventRegister
  - libvirt.c, 344
  - libvirt.h, 154
- virConnectDomainEventRegisterAny
  - libvirt.c, 344
  - libvirt.h, 154
- virConnectDomainEventTrayChangeCallback
  - libvirt.h, 123
- virConnectDomainEventWatchdogCallback
  - libvirt.h, 123
- virConnectDomainXMLFromNative
  - libvirt.c, 344
  - libvirt.h, 155
- virConnectDomainXMLToNative
  - libvirt.c, 345
  - libvirt.h, 155
- virConnectFindStoragePoolSources
  - libvirt.c, 345
  - libvirt.h, 155
- virConnectFlags
  - libvirt.h, 133
- virConnectGetCapabilities
  - libvirt.c, 345
  - libvirt.h, 155
- virConnectGetConfigFile
  - libvirt.c, 345
- virConnectGetConfigFilePath
  - libvirt.c, 345
- virConnectGetDefaultURI
  - libvirt.c, 345
- virConnectGetHostname
  - libvirt.c, 345
- virConnectGetLibVersion
  - libvirt.h, 155

- libvirt.c, [346](#)
- libvirt.h, [156](#)
- virConnectGetMaxVcpus
  - libvirt.c, [346](#)
  - libvirt.h, [156](#)
- virConnectGetSysinfo
  - libvirt.c, [346](#)
  - libvirt.h, [156](#)
- virConnectGetType
  - libvirt.c, [346](#)
  - libvirt.h, [156](#)
- virConnectGetURI
  - libvirt.c, [346](#)
  - libvirt.h, [156](#)
- virConnectGetVersion
  - libvirt.c, [346](#)
  - libvirt.h, [156](#)
- virConnectIsAlive
  - libvirt.c, [347](#)
  - libvirt.h, [157](#)
- virConnectIsEncrypted
  - libvirt.c, [347](#)
  - libvirt.h, [157](#)
- virConnectIsSecure
  - libvirt.c, [347](#)
  - libvirt.h, [157](#)
- virConnectListAllDomains
  - libvirt.c, [347](#)
  - libvirt.h, [157](#)
- virConnectListAllDomainsFlags
  - libvirt.h, [133](#)
- virConnectListAllInterfaces
  - libvirt.c, [348](#)
  - libvirt.h, [158](#)
- virConnectListAllInterfacesFlags
  - libvirt.h, [133](#)
- virConnectListAllNWFilters
  - libvirt.c, [349](#)
  - libvirt.h, [159](#)
- virConnectListAllNetworks
  - libvirt.c, [348](#)
  - libvirt.h, [158](#)
- virConnectListAllNetworksFlags
  - libvirt.h, [133](#)
- virConnectListAllNodeDeviceFlags
  - libvirt.h, [134](#)
- virConnectListAllNodeDevices
  - libvirt.c, [349](#)
  - libvirt.h, [159](#)
- virConnectListAllSecrets
  - libvirt.c, [349](#)
  - libvirt.h, [159](#)
- virConnectListAllSecretsFlags
  - libvirt.h, [134](#)
- virConnectListAllStoragePools
  - libvirt.c, [350](#)
  - libvirt.h, [160](#)
- virConnectListAllStoragePoolsFlags
  - libvirt.h, [134](#)
- virConnectListDefinedDomains
  - libvirt.c, [350](#)
  - libvirt.h, [160](#)
- virConnectListDefinedInterfaces
  - libvirt.c, [350](#)
  - libvirt.h, [160](#)
- virConnectListDefinedNetworks
  - libvirt.c, [351](#)
  - libvirt.h, [161](#)
- virConnectListDefinedStoragePools
  - libvirt.c, [351](#)
  - libvirt.h, [161](#)
- virConnectListDomains
  - libvirt.c, [351](#)
  - libvirt.h, [161](#)
- virConnectListInterfaces
  - libvirt.c, [351](#)
  - libvirt.h, [161](#)
- virConnectListNWFilters
  - libvirt.c, [352](#)
  - libvirt.h, [162](#)
- virConnectListNetworks
  - libvirt.c, [352](#)
  - libvirt.h, [161](#)
- virConnectListSecrets
  - libvirt.c, [352](#)
  - libvirt.h, [162](#)
- virConnectListStoragePools
  - libvirt.c, [352](#)
  - libvirt.h, [162](#)
- virConnectNumOfDefinedDomains
  - libvirt.c, [352](#)
  - libvirt.h, [162](#)
- virConnectNumOfDefinedInterfaces
  - libvirt.c, [352](#)
  - libvirt.h, [162](#)
- virConnectNumOfDefinedNetworks
  - libvirt.c, [353](#)
  - libvirt.h, [162](#)
- virConnectNumOfDefinedStoragePools
  - libvirt.c, [353](#)
  - libvirt.h, [163](#)
- virConnectNumOfDomains
  - libvirt.c, [353](#)
  - libvirt.h, [163](#)
- virConnectNumOfInterfaces
  - libvirt.c, [353](#)
  - libvirt.h, [163](#)
- virConnectNumOfNWFilters
  - libvirt.c, [353](#)
  - libvirt.h, [163](#)
- virConnectNumOfNetworks
  - libvirt.c, [353](#)
  - libvirt.h, [163](#)
- virConnectNumOfSecrets
  - libvirt.c, [353](#)
  - libvirt.h, [163](#)

- virConnectNumOfStoragePools
  - libvirt.c, [353](#)
  - libvirt.h, [163](#)
- virConnectOpen
  - libvirt.c, [354](#)
  - libvirt.h, [164](#)
- virConnectOpenAuth
  - libvirt.c, [354](#)
  - libvirt.h, [164](#)
- virConnectOpenFindURIAliasMatch
  - libvirt.c, [354](#)
- virConnectOpenReadOnly
  - libvirt.c, [354](#)
  - libvirt.h, [164](#)
- virConnectOpenResolveURIAlias
  - libvirt.c, [354](#)
- virConnectPtr
  - libvirt.h, [123](#)
- virConnectRef
  - libvirt.c, [354](#)
  - libvirt.h, [164](#)
- virConnectRegisterCloseCallback
  - libvirt.c, [354](#)
  - libvirt.h, [164](#)
- virConnectSetKeepAlive
  - libvirt.c, [354](#)
  - libvirt.h, [164](#)
- virConnectUnregisterCloseCallback
  - libvirt.c, [355](#)
  - libvirt.h, [164](#)
- virDevMonDeviceGetParent
  - driver.h, [316](#)
- virDevMonDeviceGetXMLDesc
  - driver.h, [316](#)
- virDevMonDeviceListCaps
  - driver.h, [316](#)
- virDevMonDeviceLookupByName
  - driver.h, [316](#)
- virDevMonDeviceNumOfCaps
  - driver.h, [316](#)
- virDevMonListAllNodeDevices
  - driver.h, [316](#)
- virDevMonListDevices
  - driver.h, [316](#)
- virDevMonNumOfDevices
  - driver.h, [316](#)
- virDeviceMonitor
  - driver.h, [316](#)
- virDeviceMonitorPtr
  - driver.h, [316](#)
- virDeviceMonitorTab
  - libvirt.c, [420](#)
- virDeviceMonitorTabCount
  - libvirt.c, [420](#)
- virDiskNameToBusDeviceIndex
  - domain\_conf.c, [237](#)
  - domain\_conf.h, [286](#)
- virDomain
  - libvirt.h, [123](#)
- virDomainAbortJob
  - libvirt.c, [355](#)
  - libvirt.h, [165](#)
- virDomainActualNetDef
  - domain\_conf.h, [265](#)
- virDomainActualNetDefFormat
  - domain\_conf.c, [237](#)
- virDomainActualNetDefFree
  - domain\_conf.c, [237](#)
  - domain\_conf.h, [286](#)
- virDomainActualNetDefParseXML
  - domain\_conf.c, [237](#)
- virDomainActualNetDefPtr
  - domain\_conf.h, [265](#)
- virDomainApicEoi
  - domain\_conf.h, [269](#)
- virDomainAssignDef
  - domain\_conf.c, [237](#)
  - domain\_conf.h, [287](#)
- virDomainAttachDevice
  - libvirt.c, [355](#)
  - libvirt.h, [165](#)
- virDomainAttachDeviceFlags
  - libvirt.c, [355](#)
  - libvirt.h, [165](#)
- virDomainBIOSDef
  - domain\_conf.h, [265](#)
- virDomainBIOSDefPtr
  - domain\_conf.h, [265](#)
- virDomainBIOSUseserial
  - domain\_conf.h, [269](#)
- virDomainBlkioDeviceWeightParseXML
  - domain\_conf.c, [237](#)
- virDomainBlockCommit
  - libvirt.c, [355](#)
  - libvirt.h, [165](#)
- virDomainBlockCommitFlags
  - libvirt.h, [135](#)
- virDomainBlockInfo
  - libvirt.h, [123](#)
- virDomainBlockInfoPtr
  - libvirt.h, [124](#)
- virDomainBlockIoTunelInfo
  - domain\_conf.h, [265](#)
- virDomainBlockIoTunelInfoPtr
  - domain\_conf.h, [265](#)
- virDomainBlockJobAbort
  - libvirt.c, [356](#)
  - libvirt.h, [166](#)
- virDomainBlockJobAbortFlags
  - libvirt.h, [135](#)
- virDomainBlockJobCursor
  - libvirt.h, [124](#)
- virDomainBlockJobInfo
  - libvirt.h, [124](#)
- virDomainBlockJobInfoPtr
  - libvirt.h, [124](#)

- virDomainBlockJobSetSpeed
  - libvirt.c, [357](#)
  - libvirt.h, [166](#)
- virDomainBlockJobType
  - libvirt.h, [135](#)
- virDomainBlockPeek
  - libvirt.c, [357](#)
  - libvirt.h, [167](#)
- virDomainBlockPull
  - libvirt.c, [357](#)
  - libvirt.h, [167](#)
- virDomainBlockRebase
  - libvirt.c, [358](#)
  - libvirt.h, [168](#)
- virDomainBlockRebaseFlags
  - libvirt.h, [136](#)
- virDomainBlockResize
  - libvirt.c, [359](#)
  - libvirt.h, [168](#)
- virDomainBlockResizeFlags
  - libvirt.h, [136](#)
- virDomainBlockStats
  - libvirt.c, [359](#)
  - libvirt.h, [169](#)
- virDomainBlockStatsFlags
  - libvirt.c, [359](#)
  - libvirt.h, [169](#)
- virDomainBlockStatsPtr
  - libvirt.h, [124](#)
- virDomainBlockStatsStruct
  - libvirt.h, [124](#)
- virDomainBlockedReason
  - libvirt.h, [135](#)
- virDomainBootMenu
  - domain\_conf.h, [270](#)
- virDomainBootOrder
  - domain\_conf.h, [270](#)
- virDomainChannelDefCheckABIStability
  - domain\_conf.c, [237](#)
- virDomainChrChannelTargetType
  - domain\_conf.h, [270](#)
- virDomainChrConsoleTargetType
  - domain\_conf.h, [270](#)
- virDomainChrDef
  - domain\_conf.h, [265](#)
- virDomainChrDefForeach
  - domain\_conf.c, [237](#)
  - domain\_conf.h, [287](#)
- virDomainChrDefFormat
  - domain\_conf.c, [238](#)
- virDomainChrDefFree
  - domain\_conf.c, [238](#)
  - domain\_conf.h, [287](#)
- virDomainChrDefGetSecurityLabelDef
  - domain\_conf.c, [238](#)
  - domain\_conf.h, [287](#)
- virDomainChrDefIterator
  - domain\_conf.h, [265](#)
- virDomainChrDefNew
  - domain\_conf.c, [238](#)
  - domain\_conf.h, [287](#)
- virDomainChrDefParseTargetXML
  - domain\_conf.c, [238](#)
- virDomainChrDefParseXML
  - domain\_conf.c, [238](#)
- virDomainChrDefPtr
  - domain\_conf.h, [265](#)
- virDomainChrDefaultTargetType
  - domain\_conf.c, [237](#)
- virDomainChrDeviceType
  - domain\_conf.h, [270](#)
- virDomainChrSourceDef
  - domain\_conf.h, [265](#)
- virDomainChrSourceDefCopy
  - domain\_conf.c, [238](#)
  - domain\_conf.h, [287](#)
- virDomainChrSourceDefFormat
  - domain\_conf.c, [238](#)
- virDomainChrSourceDefFree
  - domain\_conf.c, [238](#)
  - domain\_conf.h, [287](#)
- virDomainChrSourceDefIsEqual
  - domain\_conf.c, [238](#)
- virDomainChrSourceDefParseXML
  - domain\_conf.c, [238](#)
- virDomainChrSourceDefPtr
  - domain\_conf.h, [265](#)
- virDomainChrSpicevmcName
  - domain\_conf.h, [271](#)
- virDomainChrTargetTypeFromString
  - domain\_conf.c, [238](#)
- virDomainChrTargetTypeToString
  - domain\_conf.c, [238](#)
- virDomainChrTcpProtocol
  - domain\_conf.h, [271](#)
- virDomainChrType
  - domain\_conf.h, [271](#)
- virDomainClockBasis
  - domain\_conf.h, [271](#)
- virDomainClockDef
  - domain\_conf.h, [265](#)
- virDomainClockDefClear
  - domain\_conf.c, [238](#)
- virDomainClockDefPtr
  - domain\_conf.h, [266](#)
- virDomainClockOffsetType
  - domain\_conf.h, [272](#)
- virDomainConfigFile
  - domain\_conf.c, [238](#)
  - domain\_conf.h, [287](#)
- virDomainConsoleDefCheckABIStability
  - domain\_conf.c, [238](#)
- virDomainConsoleFlags
  - libvirt.h, [136](#)
- virDomainControllInfo
  - libvirt.h, [124](#)

- virDomainControllInfoPtr  
libvirt.h, [124](#)
- virDomainControlState  
libvirt.h, [136](#)
- virDomainControllerDef  
domain\_conf.h, [266](#)
- virDomainControllerDefCheckABIStability  
domain\_conf.c, [238](#)
- virDomainControllerDefFormat  
domain\_conf.c, [238](#)
- virDomainControllerDefFree  
domain\_conf.c, [238](#)  
domain\_conf.h, [287](#)
- virDomainControllerDefParseXML  
domain\_conf.c, [238](#)
- virDomainControllerDefPtr  
domain\_conf.h, [266](#)
- virDomainControllerFind  
domain\_conf.c, [238](#)  
domain\_conf.h, [287](#)
- virDomainControllerInsert  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainControllerInsertPreAlloced  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainControllerMaster  
domain\_conf.h, [272](#)
- virDomainControllerModelSCSI  
domain\_conf.h, [272](#)
- virDomainControllerModelTypeFromString  
domain\_conf.c, [239](#)
- virDomainControllerModelTypeToString  
domain\_conf.c, [239](#)
- virDomainControllerModelUSB  
domain\_conf.h, [272](#)
- virDomainControllerRemove  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainControllerType  
domain\_conf.h, [273](#)
- virDomainCoreDump  
libvirt.c, [360](#)  
libvirt.h, [169](#)
- virDomainCoreDumpFlags  
libvirt.h, [136](#)
- virDomainCpuPlacementMode  
domain\_conf.h, [273](#)
- virDomainCrashedReason  
libvirt.h, [137](#)
- virDomainCreate  
libvirt.c, [360](#)  
libvirt.h, [170](#)
- virDomainCreateFlags  
libvirt.h, [137](#)
- virDomainCreateLinux  
libvirt.c, [360](#)  
libvirt.h, [170](#)
- virDomainCreateWithFlags  
libvirt.c, [360](#)  
libvirt.h, [170](#)
- virDomainCreateXML  
libvirt.c, [361](#)  
libvirt.h, [171](#)
- virDomainDef  
domain\_conf.h, [266](#)
- virDomainDefAddDiskControllersForType  
domain\_conf.c, [239](#)
- virDomainDefAddImplicitControllers  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefAddSecurityLabelDef  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefCheckABIStability  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefClearDeviceAliases  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefClearPCIAddresses  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefCompatibleDevice  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefDefaultEmulator  
domain\_conf.c, [239](#)
- virDomainDefFormat  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefFormatInternal  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefFree  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefGetSecurityLabelDef  
domain\_conf.c, [239](#)  
domain\_conf.h, [287](#)
- virDomainDefHasUSB  
domain\_conf.c, [239](#)
- virDomainDefMaybeAddController  
domain\_conf.c, [239](#)
- virDomainDefMaybeAddSmartcardController  
domain\_conf.c, [239](#)
- virDomainDefMaybeAddVirtioSerialController  
domain\_conf.c, [239](#)
- virDomainDefParse  
domain\_conf.c, [239](#)
- virDomainDefParseBootXML  
domain\_conf.c, [239](#)
- virDomainDefParseFile  
domain\_conf.c, [239](#)  
domain\_conf.h, [288](#)
- virDomainDefParseNode



- domain\_conf.c, [239](#)
- domain\_conf.h, [288](#)
- virDomainDefParseString
  - domain\_conf.c, [240](#)
  - domain\_conf.h, [288](#)
- virDomainDefParseXML
  - domain\_conf.c, [240](#)
- virDomainDefPtr
  - domain\_conf.h, [266](#)
- virDomainDefineXML
  - libvirt.c, [361](#)
  - libvirt.h, [171](#)
- virDomainDeleteConfig
  - domain\_conf.c, [240](#)
  - domain\_conf.h, [288](#)
- virDomainDestroy
  - libvirt.c, [361](#)
  - libvirt.h, [171](#)
- virDomainDestroyFlags
  - libvirt.c, [361](#)
  - libvirt.h, [171](#)
- virDomainDestroyFlagsValues
  - libvirt.h, [137](#)
- virDomainDetachDevice
  - libvirt.c, [362](#)
  - libvirt.h, [172](#)
- virDomainDetachDeviceFlags
  - libvirt.c, [362](#)
  - libvirt.h, [172](#)
- virDomainDeviceAddressIsValid
  - domain\_conf.c, [240](#)
  - domain\_conf.h, [288](#)
- virDomainDeviceAddressType
  - domain\_conf.h, [273](#)
- virDomainDeviceBootParseXML
  - domain\_conf.c, [240](#)
- virDomainDeviceCcidAddress
  - domain\_conf.h, [266](#)
- virDomainDeviceCcidAddressParseXML
  - domain\_conf.c, [240](#)
- virDomainDeviceCcidAddressPtr
  - domain\_conf.h, [266](#)
- virDomainDeviceDef
  - domain\_conf.h, [266](#)
- virDomainDeviceDefCopy
  - domain\_conf.c, [240](#)
  - domain\_conf.h, [288](#)
- virDomainDeviceDefFree
  - domain\_conf.c, [240](#)
  - domain\_conf.h, [288](#)
- virDomainDeviceDefParse
  - domain\_conf.c, [240](#)
  - domain\_conf.h, [288](#)
- virDomainDeviceDefPtr
  - domain\_conf.h, [266](#)
- virDomainDeviceDriveAddress
  - domain\_conf.h, [266](#)
- virDomainDeviceDriveAddressParseXML
  - domain\_conf.c, [240](#)
- virDomainDeviceDriveAddressPtr
  - domain\_conf.h, [266](#)
- virDomainDeviceInfo
  - domain\_conf.h, [266](#)
- virDomainDeviceInfoCallback
  - domain\_conf.h, [266](#)
- virDomainDeviceInfoCheckABIStability
  - domain\_conf.c, [240](#)
- virDomainDeviceInfoClear
  - domain\_conf.c, [240](#)
  - domain\_conf.h, [288](#)
- virDomainDeviceInfoClearAlias
  - domain\_conf.c, [240](#)
- virDomainDeviceInfoClearPCIAddress
  - domain\_conf.c, [240](#)
- virDomainDeviceInfoFree
  - domain\_conf.c, [240](#)
- virDomainDeviceInfosSet
  - domain\_conf.c, [240](#)
- virDomainDeviceInfoIterate
  - domain\_conf.c, [240](#)
  - domain\_conf.h, [288](#)
- virDomainDeviceInfoParseXML
  - domain\_conf.c, [240](#)
- virDomainDeviceInfoPtr
  - domain\_conf.h, [266](#)
- virDomainDevicesUSB
  - domain\_conf.c, [240](#)
- virDomainDeviceModifyFlags
  - libvirt.h, [137](#)
- virDomainDeviceSpaprVioAddress
  - domain\_conf.h, [266](#)
- virDomainDeviceSpaprVioAddressParseXML
  - domain\_conf.c, [240](#)
- virDomainDeviceSpaprVioAddressPtr
  - domain\_conf.h, [266](#)
- virDomainDeviceType
  - domain\_conf.h, [273](#)
- virDomainDeviceUSBAddress
  - domain\_conf.h, [266](#)
- virDomainDeviceUSBAddressParseXML
  - domain\_conf.c, [240](#)
- virDomainDeviceUSBAddressPtr
  - domain\_conf.h, [266](#)
- virDomainDeviceUSBMaster
  - domain\_conf.h, [266](#)
- virDomainDeviceUSBMasterParseXML
  - domain\_conf.c, [241](#)
- virDomainDeviceUSBMasterPtr
  - domain\_conf.h, [266](#)
- virDomainDeviceVirtioSerialAddress
  - domain\_conf.h, [266](#)
- virDomainDeviceVirtioSerialAddressParseXML
  - domain\_conf.c, [241](#)
- virDomainDeviceVirtioSerialAddressPtr
  - domain\_conf.h, [266](#)
- virDomainDiskBlockIoDefFormat

- domain\_conf.c, [241](#)
- virDomainDiskBus
  - domain\_conf.h, [274](#)
- virDomainDiskCache
  - domain\_conf.h, [274](#)
- virDomainDiskCopyOnRead
  - domain\_conf.h, [274](#)
- virDomainDiskDef
  - domain\_conf.h, [266](#)
- virDomainDiskDefAssignAddress
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskDefCheckABIStability
  - domain\_conf.c, [241](#)
- virDomainDiskDefForeachPath
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskDefFormat
  - domain\_conf.c, [241](#)
- virDomainDiskDefFree
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskDefGetSecurityLabelDef
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskDefParseXML
  - domain\_conf.c, [241](#)
- virDomainDiskDefPathIterator
  - domain\_conf.h, [266](#)
- virDomainDiskDefPtr
  - domain\_conf.h, [266](#)
- virDomainDiskDevice
  - domain\_conf.h, [274](#)
- virDomainDiskError
  - libvirt.h, [124](#)
- virDomainDiskErrorCode
  - libvirt.h, [137](#)
- virDomainDiskErrorPolicy
  - domain\_conf.h, [275](#)
- virDomainDiskErrorPtr
  - libvirt.h, [124](#)
- virDomainDiskFindControllerModel
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskGeometryDefFormat
  - domain\_conf.c, [241](#)
- virDomainDiskGeometryTrans
  - domain\_conf.h, [275](#)
- virDomainDiskHostDef
  - domain\_conf.h, [266](#)
- virDomainDiskHostDefFree
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskHostDefPtr
  - domain\_conf.h, [266](#)
- virDomainDiskIndexByName
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskInsert
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskInsertPreAlloced
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskIo
  - domain\_conf.h, [275](#)
- virDomainDiskPathByName
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskProtocol
  - domain\_conf.h, [275](#)
- virDomainDiskRemove
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskRemoveByName
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainDiskSecretType
  - domain\_conf.h, [275](#)
- virDomainDiskTray
  - domain\_conf.h, [276](#)
- virDomainDiskType
  - domain\_conf.h, [276](#)
- virDomainEmulatorPinAdd
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainEmulatorPinDel
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainEventDefinedDetailType
  - libvirt.h, [138](#)
- virDomainEventGraphicsAddress
  - libvirt.h, [125](#)
- virDomainEventGraphicsAddressPtr
  - libvirt.h, [125](#)
- virDomainEventGraphicsAddressType
  - libvirt.h, [138](#)
- virDomainEventGraphicsPhase
  - libvirt.h, [138](#)
- virDomainEventGraphicsSubject
  - libvirt.h, [125](#)
- virDomainEventGraphicsSubjectIdentity
  - libvirt.h, [125](#)
- virDomainEventGraphicsSubjectIdentityPtr
  - libvirt.h, [125](#)
- virDomainEventGraphicsSubjectPtr
  - libvirt.h, [125](#)
- virDomainEventID
  - libvirt.h, [138](#)
- virDomainEventIOErrorAction
  - libvirt.h, [139](#)
- virDomainEventPMSuspendedDetailType
  - libvirt.h, [139](#)
- virDomainEventResumedDetailType
  - libvirt.h, [139](#)
- virDomainEventShutdownDetailType

- libvirt.h, [139](#)
- virDomainEventStartedDetailType
  - libvirt.h, [139](#)
- virDomainEventStoppedDetailType
  - libvirt.h, [140](#)
- virDomainEventSuspendedDetailType
  - libvirt.h, [140](#)
- virDomainEventTrayChangeReason
  - libvirt.h, [140](#)
- virDomainEventType
  - libvirt.h, [140](#)
- virDomainEventUndefinedDetailType
  - libvirt.h, [141](#)
- virDomainEventWatchdogAction
  - libvirt.h, [141](#)
- virDomainFSAccessMode
  - domain\_conf.h, [276](#)
- virDomainFSDef
  - domain\_conf.h, [267](#)
- virDomainFSDefFormat
  - domain\_conf.c, [242](#)
- virDomainFSDefFree
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainFSDefParseXML
  - domain\_conf.c, [242](#)
- virDomainFSDefPtr
  - domain\_conf.h, [267](#)
- virDomainFSDriverType
  - domain\_conf.h, [276](#)
- virDomainFSIndexByName
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainFSType
  - domain\_conf.h, [277](#)
- virDomainFSWrpolicy
  - domain\_conf.h, [277](#)
- virDomainFeature
  - domain\_conf.h, [276](#)
- virDomainFindByID
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [288](#)
- virDomainFindByName
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [289](#)
- virDomainFindByUUID
  - domain\_conf.c, [241](#)
  - domain\_conf.h, [289](#)
- virDomainFree
  - libvirt.c, [362](#)
  - libvirt.h, [172](#)
- virDomainFsDefCheckABIStability
  - domain\_conf.c, [242](#)
- virDomainGetAutostart
  - libvirt.c, [362](#)
  - libvirt.h, [172](#)
- virDomainGetBlkioParameters
  - libvirt.c, [363](#)
- libvirt.h, [172](#)
- virDomainGetBlockInfo
  - libvirt.c, [363](#)
  - libvirt.h, [173](#)
- virDomainGetBlockioTune
  - libvirt.c, [363](#)
  - libvirt.h, [173](#)
- virDomainGetBlockJobInfo
  - libvirt.c, [363](#)
  - libvirt.h, [173](#)
- virDomainGetCPUStats
  - libvirt.c, [364](#)
  - libvirt.h, [174](#)
- virDomainGetConnect
  - libvirt.c, [364](#)
  - libvirt.h, [174](#)
- virDomainGetControllInfo
  - libvirt.c, [364](#)
  - libvirt.h, [174](#)
- virDomainGetDiskErrors
  - libvirt.c, [365](#)
  - libvirt.h, [175](#)
- virDomainGetEmulatorPinInfo
  - libvirt.c, [365](#)
  - libvirt.h, [175](#)
- virDomainGetHostname
  - libvirt.c, [365](#)
  - libvirt.h, [175](#)
- virDomainGetID
  - libvirt.c, [365](#)
  - libvirt.h, [175](#)
- virDomainGetInfo
  - libvirt.c, [366](#)
  - libvirt.h, [175](#)
- virDomainGetInterfaceParameters
  - libvirt.c, [366](#)
  - libvirt.h, [176](#)
- virDomainGetJobInfo
  - libvirt.c, [366](#)
  - libvirt.h, [176](#)
- virDomainGetMaxMemory
  - libvirt.c, [366](#)
  - libvirt.h, [176](#)
- virDomainGetMaxVcpus
  - libvirt.c, [366](#)
  - libvirt.h, [176](#)
- virDomainGetMemoryParameters
  - libvirt.c, [366](#)
  - libvirt.h, [176](#)
- virDomainGetMetadata
  - libvirt.c, [367](#)
  - libvirt.h, [177](#)
- virDomainGetName
  - libvirt.c, [367](#)
  - libvirt.h, [177](#)
- virDomainGetNumaParameters
  - libvirt.c, [367](#)
  - libvirt.h, [177](#)

- virDomainGetOSType
  - libvirt.c, [368](#)
  - libvirt.h, [177](#)
- virDomainGetRootFilesystem
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainGetSchedulerParameters
  - libvirt.c, [368](#)
  - libvirt.h, [177](#)
- virDomainGetSchedulerParametersFlags
  - libvirt.c, [368](#)
  - libvirt.h, [178](#)
- virDomainGetSchedulerType
  - libvirt.c, [368](#)
  - libvirt.h, [178](#)
- virDomainGetSecurityLabel
  - libvirt.c, [368](#)
  - libvirt.h, [178](#)
- virDomainGetSecurityLabelList
  - libvirt.c, [369](#)
  - libvirt.h, [178](#)
- virDomainGetState
  - libvirt.c, [369](#)
  - libvirt.h, [178](#)
- virDomainGetUUID
  - libvirt.c, [369](#)
  - libvirt.h, [179](#)
- virDomainGetUUIDString
  - libvirt.c, [369](#)
  - libvirt.h, [179](#)
- virDomainGetVcpuPinInfo
  - libvirt.c, [369](#)
  - libvirt.h, [179](#)
- virDomainGetVcpus
  - libvirt.c, [369](#)
  - libvirt.h, [179](#)
- virDomainGetVcpusFlags
  - libvirt.c, [370](#)
  - libvirt.h, [179](#)
- virDomainGetXMLDesc
  - libvirt.c, [370](#)
  - libvirt.h, [180](#)
- virDomainGraphicsAuthConnectedType
  - domain\_conf.h, [277](#)
- virDomainGraphicsAuthDef
  - domain\_conf.h, [267](#)
- virDomainGraphicsAuthDefClear
  - domain\_conf.c, [242](#)
- virDomainGraphicsAuthDefFormatAttr
  - domain\_conf.c, [242](#)
- virDomainGraphicsAuthDefParseXML
  - domain\_conf.c, [242](#)
- virDomainGraphicsAuthDefPtr
  - domain\_conf.h, [267](#)
- virDomainGraphicsDef
  - domain\_conf.h, [267](#)
- virDomainGraphicsDefFormat
  - domain\_conf.c, [242](#)
- virDomainGraphicsDefFree
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainGraphicsDefParseXML
  - domain\_conf.c, [242](#)
- virDomainGraphicsDefPtr
  - domain\_conf.h, [267](#)
- virDomainGraphicsGetListen
  - domain\_conf.c, [242](#)
- virDomainGraphicsListenDef
  - domain\_conf.h, [267](#)
- virDomainGraphicsListenDefClear
  - domain\_conf.c, [242](#)
- virDomainGraphicsListenDefFormat
  - domain\_conf.c, [242](#)
- virDomainGraphicsListenDefParseXML
  - domain\_conf.c, [242](#)
- virDomainGraphicsListenDefPtr
  - domain\_conf.h, [267](#)
- virDomainGraphicsListenGetAddress
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainGraphicsListenGetNetwork
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainGraphicsListenGetType
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainGraphicsListenSetAddress
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainGraphicsListenSetNetwork
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainGraphicsListenSetType
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainGraphicsListenType
  - domain\_conf.h, [277](#)
- virDomainGraphicsSpiceChannelMode
  - domain\_conf.h, [277](#)
- virDomainGraphicsSpiceChannelName
  - domain\_conf.h, [278](#)
- virDomainGraphicsSpiceClipboardCopypaste
  - domain\_conf.h, [278](#)
- virDomainGraphicsSpiceImageCompression
  - domain\_conf.h, [278](#)
- virDomainGraphicsSpiceJpegCompression
  - domain\_conf.h, [278](#)
- virDomainGraphicsSpiceMouseMode
  - domain\_conf.h, [279](#)
- virDomainGraphicsSpicePlaybackCompression
  - domain\_conf.h, [279](#)
- virDomainGraphicsSpiceStreamingMode
  - domain\_conf.h, [279](#)
- virDomainGraphicsSpiceZlibCompression
  - domain\_conf.h, [279](#)
- virDomainGraphicsType

- domain\_conf.h, [279](#)
- virDomainHasCurrentSnapshot
  - libvirt.c, [370](#)
  - libvirt.h, [180](#)
- virDomainHasManagedSavImage
  - libvirt.c, [370](#)
  - libvirt.h, [180](#)
- virDomainHostdevDef
  - domain\_conf.h, [267](#)
- virDomainHostdevDefAlloc
  - domain\_conf.c, [242](#)
  - domain\_conf.h, [289](#)
- virDomainHostdevDefCheckABIStability
  - domain\_conf.c, [243](#)
- virDomainHostdevDefClear
  - domain\_conf.c, [243](#)
  - domain\_conf.h, [289](#)
- virDomainHostdevDefFormat
  - domain\_conf.c, [243](#)
- virDomainHostdevDefFree
  - domain\_conf.c, [243](#)
  - domain\_conf.h, [289](#)
- virDomainHostdevDefParseXML
  - domain\_conf.c, [243](#)
- virDomainHostdevDefPtr
  - domain\_conf.h, [267](#)
- virDomainHostdevFind
  - domain\_conf.c, [243](#)
  - domain\_conf.h, [289](#)
- virDomainHostdevInsert
  - domain\_conf.c, [243](#)
  - domain\_conf.h, [289](#)
- virDomainHostdevMode
  - domain\_conf.h, [280](#)
- virDomainHostdevOrigStates
  - domain\_conf.h, [267](#)
- virDomainHostdevOrigStatesPtr
  - domain\_conf.h, [267](#)
- virDomainHostdevPartsParse
  - domain\_conf.c, [243](#)
- virDomainHostdevRemove
  - domain\_conf.c, [243](#)
  - domain\_conf.h, [289](#)
- virDomainHostdevSourceFormat
  - domain\_conf.c, [243](#)
- virDomainHostdevSubsys
  - domain\_conf.h, [267](#)
- virDomainHostdevSubsysPciDefParseXML
  - domain\_conf.c, [243](#)
- virDomainHostdevSubsysPciOrigStatesDefParseXML
  - domain\_conf.c, [243](#)
- virDomainHostdevSubsysPtr
  - domain\_conf.h, [267](#)
- virDomainHostdevSubsysType
  - domain\_conf.h, [280](#)
- virDomainHostdevSubsysUsbDefParseXML
  - domain\_conf.c, [243](#)
- virDomainHubDef
  - domain\_conf.h, [267](#)
- virDomainHubDefCheckABIStability
  - domain\_conf.c, [243](#)
- virDomainHubDefFormat
  - domain\_conf.c, [243](#)
- virDomainHubDefFree
  - domain\_conf.c, [243](#)
  - domain\_conf.h, [289](#)
- virDomainHubDefParseXML
  - domain\_conf.c, [243](#)
- virDomainHubDefPtr
  - domain\_conf.h, [267](#)
- virDomainHubType
  - domain\_conf.h, [280](#)
- virDomainIDDData, [90](#)
  - ids, [90](#)
  - maxids, [90](#)
  - numids, [90](#)
- virDomainInfo
  - libvirt.h, [125](#)
- virDomainInfoPtr
  - libvirt.h, [125](#)
- virDomainInjectNML
  - libvirt.c, [370](#)
  - libvirt.h, [180](#)
- virDomainInputBus
  - domain\_conf.h, [280](#)
- virDomainInputDef
  - domain\_conf.h, [267](#)
- virDomainInputDefCheckABIStability
  - domain\_conf.c, [243](#)
- virDomainInputDefFormat
  - domain\_conf.c, [243](#)
- virDomainInputDefFree
  - domain\_conf.c, [243](#)
  - domain\_conf.h, [289](#)
- virDomainInputDefParseXML
  - domain\_conf.c, [243](#)
- virDomainInputDefPtr
  - domain\_conf.h, [267](#)
- virDomainInputType
  - domain\_conf.h, [280](#)
- virDomainInterfaceStats
  - libvirt.c, [371](#)
  - libvirt.h, [180](#)
- virDomainInterfaceStatsPtr
  - libvirt.h, [125](#)
- virDomainInterfaceStatsStruct
  - libvirt.h, [125](#)
- virDomainIoEventFd
  - domain\_conf.h, [280](#)
- virDomainIsActive
  - libvirt.c, [371](#)
  - libvirt.h, [181](#)
- virDomainIsPersistent
  - libvirt.c, [371](#)
  - libvirt.h, [181](#)
- virDomainIsUpdated

- libvirt.c, [371](#)
- libvirt.h, [181](#)
- virDomainJobInfo
  - libvirt.h, [125](#)
- virDomainJobInfoPtr
  - libvirt.h, [125](#)
- virDomainJobType
  - libvirt.h, [141](#)
- virDomainLeaseDef
  - domain\_conf.h, [267](#)
- virDomainLeaseDefFormat
  - domain\_conf.c, [244](#)
- virDomainLeaseDefFree
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [289](#)
- virDomainLeaseDefParseXML
  - domain\_conf.c, [244](#)
- virDomainLeaseDefPtr
  - domain\_conf.h, [267](#)
- virDomainLeaseIndex
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [289](#)
- virDomainLeaseInsert
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [289](#)
- virDomainLeaseInsertPreAlloc
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [289](#)
- virDomainLeaseInsertPreAlloced
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [289](#)
- virDomainLeaseRemove
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [289](#)
- virDomainLeaseRemoveAt
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [289](#)
- virDomainLifecycleAction
  - domain\_conf.h, [281](#)
- virDomainLifecycleCrashAction
  - domain\_conf.h, [281](#)
- virDomainLifecycleDefFormat
  - domain\_conf.c, [244](#)
- virDomainLifecycleParseXML
  - domain\_conf.c, [244](#)
- virDomainList
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [290](#)
- virDomainListAllSnapshots
  - libvirt.c, [371](#)
  - libvirt.h, [181](#)
- virDomainListData, [90](#)
  - conn, [91](#)
  - domains, [91](#)
  - error, [91](#)
  - flags, [91](#)
  - ndomains, [91](#)
- virDomainListPopulate
  - domain\_conf.c, [244](#)
- virDomainLiveConfigHelperMethod
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [290](#)
- virDomainLoadAllConfigs
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [290](#)
- virDomainLoadConfig
  - domain\_conf.c, [244](#)
- virDomainLoadConfigNotify
  - domain\_conf.h, [267](#)
- virDomainLoadStatus
  - domain\_conf.c, [244](#)
- virDomainLookupByID
  - libvirt.c, [372](#)
  - libvirt.h, [182](#)
- virDomainLookupByName
  - libvirt.c, [372](#)
  - libvirt.h, [182](#)
- virDomainLookupByUUID
  - libvirt.c, [372](#)
  - libvirt.h, [182](#)
- virDomainLookupByUUIDString
  - libvirt.c, [372](#)
  - libvirt.h, [182](#)
- virDomainManagedSave
  - libvirt.c, [372](#)
  - libvirt.h, [182](#)
- virDomainManagedSaveRemove
  - libvirt.c, [373](#)
  - libvirt.h, [182](#)
- virDomainMemDump
  - domain\_conf.h, [281](#)
- virDomainMemballoonDef
  - domain\_conf.h, [267](#)
- virDomainMemballoonDefCheckABIStability
  - domain\_conf.c, [244](#)
- virDomainMemballoonDefFormat
  - domain\_conf.c, [244](#)
- virDomainMemballoonDefFree
  - domain\_conf.c, [244](#)
  - domain\_conf.h, [290](#)
- virDomainMemballoonDefParseXML
  - domain\_conf.c, [244](#)
- virDomainMemballoonDefPtr
  - domain\_conf.h, [267](#)
- virDomainMemoryFlags
  - libvirt.h, [141](#)
- virDomainMemoryModFlags
  - libvirt.h, [142](#)
- virDomainMemoryPeek
  - libvirt.c, [373](#)
  - libvirt.h, [183](#)
- virDomainMemoryStatPtr
  - libvirt.h, [125](#)
- virDomainMemoryStatStruct
  - libvirt.h, [125](#)
- virDomainMemoryStatTags

- libvirt.h, [142](#)
- virDomainMemoryStats
  - libvirt.c, [373](#)
  - libvirt.h, [183](#)
- virDomainMetadataType
  - libvirt.h, [142](#)
- virDomainMigrate
  - libvirt.c, [373](#)
  - libvirt.h, [183](#)
- virDomainMigrate2
  - libvirt.c, [374](#)
  - libvirt.h, [184](#)
- virDomainMigrateBegin3
  - libvirt.c, [375](#)
- virDomainMigrateConfirm3
  - libvirt.c, [375](#)
- virDomainMigrateDirect
  - libvirt.c, [375](#)
- virDomainMigrateFinish
  - libvirt.c, [375](#)
- virDomainMigrateFinish2
  - libvirt.c, [375](#)
- virDomainMigrateFinish3
  - libvirt.c, [375](#)
- virDomainMigrateFlags
  - libvirt.h, [142](#)
- virDomainMigrateGetMaxSpeed
  - libvirt.c, [375](#)
  - libvirt.h, [185](#)
- virDomainMigratePeer2Peer
  - libvirt.c, [376](#)
- virDomainMigratePerform
  - libvirt.c, [376](#)
- virDomainMigratePerform3
  - libvirt.c, [376](#)
- virDomainMigratePrepare
  - libvirt.c, [376](#)
- virDomainMigratePrepare2
  - libvirt.c, [376](#)
- virDomainMigratePrepare3
  - libvirt.c, [376](#)
- virDomainMigratePrepareTunnel
  - libvirt.c, [376](#)
- virDomainMigratePrepareTunnel3
  - libvirt.c, [376](#)
- virDomainMigrateSetMaxDowntime
  - libvirt.c, [376](#)
  - libvirt.h, [185](#)
- virDomainMigrateSetMaxSpeed
  - libvirt.c, [376](#)
  - libvirt.h, [185](#)
- virDomainMigrateToURI
  - libvirt.c, [376](#)
  - libvirt.h, [186](#)
- virDomainMigrateToURI2
  - libvirt.c, [377](#)
  - libvirt.h, [186](#)
- virDomainMigrateVersion1
  - libvirt.c, [378](#)
- virDomainMigrateVersion2
  - libvirt.c, [378](#)
- virDomainMigrateVersion3
  - libvirt.c, [378](#)
- virDomainModificationImpact
  - libvirt.h, [143](#)
- virDomainNameData, [91](#)
  - maxnames, [91](#)
  - names, [91](#)
  - numnames, [91](#)
  - oom, [91](#)
- virDomainNetBackendType
  - domain\_conf.h, [281](#)
- virDomainNetDef
  - domain\_conf.h, [267](#)
- virDomainNetDefCheckABIStability
  - domain\_conf.c, [245](#)
- virDomainNetDefFormat
  - domain\_conf.c, [245](#)
- virDomainNetDefFree
  - domain\_conf.c, [245](#)
  - domain\_conf.h, [290](#)
- virDomainNetDefParseXML
  - domain\_conf.c, [245](#)
- virDomainNetDefPtr
  - domain\_conf.h, [267](#)
- virDomainNetFind
  - domain\_conf.c, [245](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualBandwidth
  - domain\_conf.c, [245](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualBridgeName
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualBridgeType
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualDirectDev
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualDirectMode
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualHostdev
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualType
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualVirtPortProfile
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [290](#)
- virDomainNetGetActualVlan
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [291](#)
- virDomainNetIndexByMac



- domain\_conf.c, [246](#)
- domain\_conf.h, [291](#)
- virDomainNetInsert
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [291](#)
- virDomainNetInterfaceLinkState
  - domain\_conf.h, [281](#)
- virDomainNetRemove
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [291](#)
- virDomainNetRemoveByMac
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [291](#)
- virDomainNetType
  - domain\_conf.h, [282](#)
- virDomainNetVirtioTxModeType
  - domain\_conf.h, [282](#)
- virDomainNostateReason
  - libvirt.h, [143](#)
- virDomainNumatuneDef
  - domain\_conf.h, [267](#)
- virDomainNumatuneDefPtr
  - domain\_conf.h, [267](#)
- virDomainNumatuneMemMode
  - libvirt.h, [143](#)
- virDomainNumatuneMemPlacementMode
  - domain\_conf.h, [282](#)
- virDomainOSDef
  - domain\_conf.h, [268](#)
- virDomainOSDefPtr
  - domain\_conf.h, [268](#)
- virDomainObj
  - domain\_conf.h, [267](#)
- virDomainObjAssignDef
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [291](#)
- virDomainObjCopyPersistentDef
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [291](#)
- virDomainObjDispose
  - domain\_conf.c, [246](#)
- virDomainObjFormat
  - domain\_conf.c, [246](#)
- virDomainObjGetPersistentDef
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [291](#)
- virDomainObjGetState
  - domain\_conf.c, [246](#)
  - domain\_conf.h, [291](#)
- virDomainObjsActive
  - domain\_conf.h, [291](#)
- virDomainObjsDuplicate
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjList
  - domain\_conf.h, [268](#)
- virDomainObjListCopyActiveIDs
  - domain\_conf.c, [247](#)
- virDomainObjListCopyInactiveNames
  - domain\_conf.c, [247](#)
- virDomainObjListCountActive
  - domain\_conf.c, [247](#)
- virDomainObjListCountInactive
  - domain\_conf.c, [247](#)
- virDomainObjListDataFree
  - domain\_conf.c, [247](#)
- virDomainObjListDeinit
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjListGetActiveIDs
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjListGetInactiveNames
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjListInit
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjListNumOfDomains
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjListPtr
  - domain\_conf.h, [268](#)
- virDomainObjListSearchID
  - domain\_conf.c, [247](#)
- virDomainObjListSearchName
  - domain\_conf.c, [247](#)
- virDomainObjLock
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjNew
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjParseFile
  - domain\_conf.c, [247](#)
- virDomainObjParseNode
  - domain\_conf.c, [247](#)
- virDomainObjParseXML
  - domain\_conf.c, [247](#)
- virDomainObjPtr
  - domain\_conf.h, [268](#)
- virDomainObjSetDefTransient
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjSetState
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjTaint
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainObjUnlock
  - domain\_conf.c, [247](#)
  - domain\_conf.h, [291](#)
- virDomainOpenConsole
  - libvirt.c, [378](#)
  - libvirt.h, [187](#)



- virDomainOpenGraphics
  - libvirt.c, [378](#)
  - libvirt.h, [187](#)
- virDomainOpenGraphicsFlags
  - libvirt.h, [143](#)
- virDomainPMState
  - domain\_conf.h, [282](#)
- virDomainPMStateParseXML
  - domain\_conf.c, [248](#)
- virDomainPMSuspendForDuration
  - libvirt.c, [380](#)
  - libvirt.h, [188](#)
- virDomainPMSuspendedReason
  - libvirt.h, [144](#)
- virDomainPMWakeup
  - libvirt.c, [380](#)
  - libvirt.h, [189](#)
- virDomainParallelDefCheckABIStability
  - domain\_conf.c, [247](#)
- virDomainParseLegacyDeviceAddress
  - domain\_conf.c, [248](#)
- virDomainParseMemory
  - domain\_conf.c, [248](#)
- virDomainParseScaledValue
  - domain\_conf.c, [248](#)
- virDomainPausedReason
  - libvirt.h, [143](#)
- virDomainPciRombarMode
  - domain\_conf.h, [282](#)
- virDomainPinEmulator
  - libvirt.c, [379](#)
  - libvirt.h, [188](#)
- virDomainPinVcpu
  - libvirt.c, [379](#)
  - libvirt.h, [188](#)
- virDomainPinVcpuFlags
  - libvirt.c, [379](#)
  - libvirt.h, [188](#)
- virDomainPtr
  - libvirt.h, [125](#)
- virDomainReboot
  - libvirt.c, [380](#)
  - libvirt.h, [189](#)
- virDomainRebootFlagValues
  - libvirt.h, [144](#)
- virDomainRedirFilterDef
  - domain\_conf.h, [268](#)
- virDomainRedirFilterDefCheckABIStability
  - domain\_conf.c, [248](#)
- virDomainRedirFilterDefFormat
  - domain\_conf.c, [248](#)
- virDomainRedirFilterDefFree
  - domain\_conf.c, [248](#)
  - domain\_conf.h, [291](#)
- virDomainRedirFilterDefParseXML
  - domain\_conf.c, [248](#)
- virDomainRedirFilterDefPtr
  - domain\_conf.h, [268](#)
- virDomainRedirFilterUsbDevDef
  - domain\_conf.h, [268](#)
- virDomainRedirFilterUsbDevDefParseXML
  - domain\_conf.c, [248](#)
- virDomainRedirFilterUsbDevDefPtr
  - domain\_conf.h, [268](#)
- virDomainRedirFilterUsbVersionHelper
  - domain\_conf.c, [248](#)
- virDomainRedirdevBus
  - domain\_conf.h, [283](#)
- virDomainRedirdevDef
  - domain\_conf.h, [268](#)
- virDomainRedirdevDefFormat
  - domain\_conf.c, [248](#)
- virDomainRedirdevDefFree
  - domain\_conf.c, [248](#)
  - domain\_conf.h, [291](#)
- virDomainRedirdevDefParseXML
  - domain\_conf.c, [248](#)
- virDomainRedirdevDefPtr
  - domain\_conf.h, [268](#)
- virDomainRef
  - libvirt.c, [380](#)
  - libvirt.h, [189](#)
- virDomainRemoveInactive
  - domain\_conf.c, [248](#)
  - domain\_conf.h, [292](#)
- virDomainReset
  - libvirt.c, [380](#)
  - libvirt.h, [189](#)
- virDomainRestore
  - libvirt.c, [381](#)
  - libvirt.h, [189](#)
- virDomainRestoreFlags
  - libvirt.c, [381](#)
  - libvirt.h, [190](#)
- virDomainResume
  - libvirt.c, [381](#)
  - libvirt.h, [190](#)
- virDomainRevertToSnapshot
  - libvirt.c, [381](#)
  - libvirt.h, [190](#)
- virDomainRunningReason
  - libvirt.h, [144](#)
- virDomainSave
  - libvirt.c, [382](#)
  - libvirt.h, [191](#)
- virDomainSaveConfig
  - domain\_conf.c, [248](#)
  - domain\_conf.h, [292](#)
- virDomainSaveFlags
  - libvirt.c, [382](#)
  - libvirt.h, [191](#)
- virDomainSavelImageDefineXML
  - libvirt.c, [382](#)
  - libvirt.h, [191](#)
- virDomainSavelImageGetXMLDesc
  - libvirt.c, [383](#)

- libvirt.h, 191
- virDomainSaveRestoreFlags
  - libvirt.h, 144
- virDomainSaveStatus
  - domain\_conf.c, 248
  - domain\_conf.h, 292
- virDomainSaveXML
  - domain\_conf.c, 248
  - domain\_conf.h, 292
- virDomainScreenshot
  - libvirt.c, 383
  - libvirt.h, 191
- virDomainSeclabelType
  - domain\_conf.h, 283
- virDomainSendKey
  - libvirt.c, 383
  - libvirt.h, 192
- virDomainSerialDefCheckABIStability
  - domain\_conf.c, 248
- virDomainSetAutostart
  - libvirt.c, 383
  - libvirt.h, 192
- virDomainSetBlkioParameters
  - libvirt.c, 383
  - libvirt.h, 192
- virDomainSetBlockIoTune
  - libvirt.c, 383
  - libvirt.h, 192
- virDomainSetInterfaceParameters
  - libvirt.c, 384
  - libvirt.h, 192
- virDomainSetMaxMemory
  - libvirt.c, 384
  - libvirt.h, 193
- virDomainSetMemory
  - libvirt.c, 384
  - libvirt.h, 193
- virDomainSetMemoryFlags
  - libvirt.c, 384
  - libvirt.h, 193
- virDomainSetMemoryParameters
  - libvirt.c, 385
  - libvirt.h, 193
- virDomainSetMetadata
  - libvirt.c, 385
  - libvirt.h, 194
- virDomainSetNumaParameters
  - libvirt.c, 385
  - libvirt.h, 194
- virDomainSetSchedulerParameters
  - libvirt.c, 385
  - libvirt.h, 194
- virDomainSetSchedulerParametersFlags
  - libvirt.c, 385
  - libvirt.h, 194
- virDomainSetVcpus
  - libvirt.c, 386
  - libvirt.h, 194
- virDomainSetVcpusFlags
  - libvirt.c, 386
  - libvirt.h, 195
- virDomainShutdown
  - libvirt.c, 386
  - libvirt.h, 195
- virDomainShutdownFlagValues
  - libvirt.h, 144
- virDomainShutdownFlags
  - libvirt.c, 387
  - libvirt.h, 195
- virDomainShutdownReason
  - libvirt.h, 145
- virDomainShutoffReason
  - libvirt.h, 145
- virDomainSmartcardDef
  - domain\_conf.h, 268
- virDomainSmartcardDefCheckABIStability
  - domain\_conf.c, 248
- virDomainSmartcardDefForeach
  - domain\_conf.c, 248
  - domain\_conf.h, 292
- virDomainSmartcardDefFormat
  - domain\_conf.c, 248
- virDomainSmartcardDefFree
  - domain\_conf.c, 248
  - domain\_conf.h, 292
- virDomainSmartcardDefIterator
  - domain\_conf.h, 268
- virDomainSmartcardDefParseXML
  - domain\_conf.c, 249
- virDomainSmartcardDefPtr
  - domain\_conf.h, 268
- virDomainSmartcardType
  - domain\_conf.h, 283
- virDomainSmbiosMode
  - domain\_conf.h, 283
- virDomainSnapshot
  - libvirt.h, 125
- virDomainSnapshotCreateFlags
  - libvirt.h, 145
- virDomainSnapshotCreateXML
  - libvirt.c, 387
  - libvirt.h, 196
- virDomainSnapshotCurrent
  - libvirt.c, 388
  - libvirt.h, 197
- virDomainSnapshotDelete
  - libvirt.c, 388
  - libvirt.h, 197
- virDomainSnapshotDeleteFlags
  - libvirt.h, 145
- virDomainSnapshotFree
  - libvirt.c, 388
  - libvirt.h, 197
- virDomainSnapshotGetConnect
  - libvirt.c, 388
  - libvirt.h, 197

- virDomainSnapshotGetDomain
  - libvirt.c, [388](#)
  - libvirt.h, [197](#)
- virDomainSnapshotGetName
  - libvirt.c, [389](#)
  - libvirt.h, [197](#)
- virDomainSnapshotGetParent
  - libvirt.c, [389](#)
  - libvirt.h, [198](#)
- virDomainSnapshotGetXMLDesc
  - libvirt.c, [389](#)
  - libvirt.h, [198](#)
- virDomainSnapshotHasMetadata
  - libvirt.c, [389](#)
  - libvirt.h, [198](#)
- virDomainSnapshotIsCurrent
  - libvirt.c, [389](#)
  - libvirt.h, [198](#)
- virDomainSnapshotListAllChildren
  - libvirt.c, [389](#)
  - libvirt.h, [198](#)
- virDomainSnapshotListChildrenNames
  - libvirt.c, [390](#)
  - libvirt.h, [199](#)
- virDomainSnapshotListFlags
  - libvirt.h, [145](#)
- virDomainSnapshotListNames
  - libvirt.c, [390](#)
  - libvirt.h, [199](#)
- virDomainSnapshotLookupByName
  - libvirt.c, [391](#)
  - libvirt.h, [200](#)
- virDomainSnapshotNum
  - libvirt.c, [391](#)
  - libvirt.h, [200](#)
- virDomainSnapshotNumChildren
  - libvirt.c, [391](#)
  - libvirt.h, [200](#)
- virDomainSnapshotObj
  - domain\_conf.h, [268](#)
- virDomainSnapshotObjList
  - domain\_conf.h, [268](#)
- virDomainSnapshotObjListPtr
  - domain\_conf.h, [268](#)
- virDomainSnapshotObjPtr
  - domain\_conf.h, [268](#)
- virDomainSnapshotPtr
  - libvirt.h, [126](#)
- virDomainSnapshotRef
  - libvirt.c, [392](#)
  - libvirt.h, [201](#)
- virDomainSnapshotRevertFlags
  - libvirt.h, [146](#)
- virDomainSoundCodecDef
  - domain\_conf.h, [268](#)
- virDomainSoundCodecDefFormat
  - domain\_conf.c, [249](#)
- virDomainSoundCodecDefFree
  - domain\_conf.c, [249](#)
- virDomainSoundCodecDefParseXML
  - domain\_conf.c, [249](#)
- virDomainSoundCodecDefPtr
  - domain\_conf.h, [268](#)
- virDomainSoundCodecType
  - domain\_conf.h, [283](#)
- virDomainSoundDef
  - domain\_conf.h, [268](#)
- virDomainSoundDefCheckABIStability
  - domain\_conf.c, [249](#)
- virDomainSoundDefFormat
  - domain\_conf.c, [249](#)
- virDomainSoundDefFree
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainSoundDefParseXML
  - domain\_conf.c, [249](#)
- virDomainSoundDefPtr
  - domain\_conf.h, [268](#)
- virDomainSoundModel
  - domain\_conf.h, [283](#)
- virDomainStartupPolicy
  - domain\_conf.h, [284](#)
- virDomainState
  - libvirt.h, [146](#)
- virDomainStateReason
  - domain\_conf.h, [268](#)
- virDomainStateReasonFromString
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainStateReasonToString
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainSuspend
  - libvirt.c, [392](#)
  - libvirt.h, [201](#)
- virDomainSysinfoDefFormat
  - domain\_conf.c, [249](#)
- virDomainTaintFlags
  - domain\_conf.h, [284](#)
- virDomainTimerCatchupDef
  - domain\_conf.h, [268](#)
- virDomainTimerCatchupDefPtr
  - domain\_conf.h, [268](#)
- virDomainTimerDef
  - domain\_conf.h, [268](#)
- virDomainTimerDefCheckABIStability
  - domain\_conf.c, [249](#)
- virDomainTimerDefFormat
  - domain\_conf.c, [249](#)
- virDomainTimerDefParseXML
  - domain\_conf.c, [249](#)
- virDomainTimerDefPtr
  - domain\_conf.h, [268](#)
- virDomainTimerModeType
  - domain\_conf.h, [284](#)

- virDomainTimerNameType
  - domain\_conf.h, [284](#)
- virDomainTimerTickpolicyType
  - domain\_conf.h, [285](#)
- virDomainTimerTrackType
  - domain\_conf.h, [285](#)
- virDomainUndefine
  - libvirt.c, [392](#)
  - libvirt.h, [201](#)
- virDomainUndefineFlags
  - libvirt.c, [392](#)
  - libvirt.h, [201](#)
- virDomainUndefineFlagsValues
  - libvirt.h, [146](#)
- virDomainUpdateDeviceFlags
  - libvirt.c, [393](#)
  - libvirt.h, [201](#)
- virDomainVcpuFlags
  - libvirt.h, [146](#)
- virDomainVcpuPinAdd
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainVcpuPinDef
  - domain\_conf.h, [268](#)
- virDomainVcpuPinDefArrayFree
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainVcpuPinDefCopy
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainVcpuPinDefFree
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainVcpuPinDefParseXML
  - domain\_conf.c, [249](#)
- virDomainVcpuPinDefPtr
  - domain\_conf.h, [269](#)
- virDomainVcpuPinDel
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainVcpuPinFindByVcpu
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainVcpuPinsDuplicate
  - domain\_conf.c, [249](#)
  - domain\_conf.h, [292](#)
- virDomainVideoAccelDef
  - domain\_conf.h, [269](#)
- virDomainVideoAccelDefFormat
  - domain\_conf.c, [249](#)
- virDomainVideoAccelDefParseXML
  - domain\_conf.c, [250](#)
- virDomainVideoAccelDefPtr
  - domain\_conf.h, [269](#)
- virDomainVideoDef
  - domain\_conf.h, [269](#)
- virDomainVideoDefCheckABIStability
  - domain\_conf.c, [250](#)
- virDomainVideoDefFormat
  - domain\_conf.c, [250](#)
- virDomainVideoDefFree
  - domain\_conf.c, [250](#)
  - domain\_conf.h, [292](#)
- virDomainVideoDefParseXML
  - domain\_conf.c, [250](#)
- virDomainVideoDefPtr
  - domain\_conf.h, [269](#)
- virDomainVideoDefaultRAM
  - domain\_conf.c, [250](#)
  - domain\_conf.h, [292](#)
- virDomainVideoDefaultType
  - domain\_conf.c, [250](#)
  - domain\_conf.h, [292](#)
- virDomainVideoType
  - domain\_conf.h, [285](#)
- virDomainVirtType
  - domain\_conf.h, [285](#)
- virDomainVirtioEventIdx
  - domain\_conf.h, [285](#)
- virDomainVirtioSerialOpts
  - domain\_conf.h, [269](#)
- virDomainVirtioSerialOptsPtr
  - domain\_conf.h, [269](#)
- virDomainWatchdogAction
  - domain\_conf.h, [286](#)
- virDomainWatchdogDef
  - domain\_conf.h, [269](#)
- virDomainWatchdogDefCheckABIStability
  - domain\_conf.c, [250](#)
- virDomainWatchdogDefFormat
  - domain\_conf.c, [250](#)
- virDomainWatchdogDefFree
  - domain\_conf.c, [250](#)
  - domain\_conf.h, [292](#)
- virDomainWatchdogDefParseXML
  - domain\_conf.c, [250](#)
- virDomainWatchdogDefPtr
  - domain\_conf.h, [269](#)
- virDomainWatchdogModel
  - domain\_conf.h, [286](#)
- virDomainXMLFlags
  - libvirt.h, [147](#)
- virDomainXMLInternalFlags
  - domain\_conf.c, [236](#)
- virDriver
  - driver.h, [316](#)
- virDriverLoadModule
  - driver.h, [329](#)
- virDriverModuleInitialize
  - driver.h, [329](#)
- virDriverPtr
  - driver.h, [316](#)
- virDriverTab
  - libvirt.c, [420](#)
- virDriverTabCount
  - libvirt.c, [420](#)

- virDrvBaselineCPU
  - driver.h, [316](#)
- virDrvClose
  - driver.h, [316](#)
- virDrvCompareCPU
  - driver.h, [316](#)
- virDrvConnectDomainXMLFromNative
  - driver.h, [316](#)
- virDrvConnectDomainXMLToNative
  - driver.h, [316](#)
- virDrvConnectFindStoragePoolSources
  - driver.h, [316](#)
- virDrvConnectIsAlive
  - driver.h, [316](#)
- virDrvConnectIsEncrypted
  - driver.h, [316](#)
- virDrvConnectIsSecure
  - driver.h, [316](#)
- virDrvConnectListAllNWFilters
  - driver.h, [316](#)
- virDrvConnectListAllStoragePools
  - driver.h, [316](#)
- virDrvConnectListDefinedStoragePools
  - driver.h, [316](#)
- virDrvConnectListNWFilters
  - driver.h, [316](#)
- virDrvConnectListStoragePools
  - driver.h, [317](#)
- virDrvConnectNumOfDefinedStoragePools
  - driver.h, [317](#)
- virDrvConnectNumOfNWFilters
  - driver.h, [317](#)
- virDrvConnectNumOfStoragePools
  - driver.h, [317](#)
- virDrvDomainAbortJob
  - driver.h, [317](#)
- virDrvDomainAttachDevice
  - driver.h, [317](#)
- virDrvDomainAttachDeviceFlags
  - driver.h, [317](#)
- virDrvDomainBlockCommit
  - driver.h, [317](#)
- virDrvDomainBlockJobAbort
  - driver.h, [317](#)
- virDrvDomainBlockJobSetSpeed
  - driver.h, [317](#)
- virDrvDomainBlockPeek
  - driver.h, [317](#)
- virDrvDomainBlockPull
  - driver.h, [317](#)
- virDrvDomainBlockRebase
  - driver.h, [317](#)
- virDrvDomainBlockResize
  - driver.h, [317](#)
- virDrvDomainBlockStats
  - driver.h, [317](#)
- virDrvDomainBlockStatsFlags
  - driver.h, [317](#)
- virDrvDomainCoreDump
  - driver.h, [317](#)
- virDrvDomainCreate
  - driver.h, [317](#)
- virDrvDomainCreateWithFlags
  - driver.h, [317](#)
- virDrvDomainCreateXML
  - driver.h, [317](#)
- virDrvDomainDefineXML
  - driver.h, [317](#)
- virDrvDomainDestroy
  - driver.h, [317](#)
- virDrvDomainDestroyFlags
  - driver.h, [317](#)
- virDrvDomainDetachDevice
  - driver.h, [317](#)
- virDrvDomainDetachDeviceFlags
  - driver.h, [318](#)
- virDrvDomainEventDeregister
  - driver.h, [318](#)
- virDrvDomainEventDeregisterAny
  - driver.h, [318](#)
- virDrvDomainEventRegister
  - driver.h, [318](#)
- virDrvDomainEventRegisterAny
  - driver.h, [318](#)
- virDrvDomainGetAutostart
  - driver.h, [318](#)
- virDrvDomainGetBlkioParameters
  - driver.h, [318](#)
- virDrvDomainGetBlockInfo
  - driver.h, [318](#)
- virDrvDomainGetBlockIoTune
  - driver.h, [318](#)
- virDrvDomainGetBlockJobInfo
  - driver.h, [318](#)
- virDrvDomainGetCPUStats
  - driver.h, [318](#)
- virDrvDomainGetCtrlInfo
  - driver.h, [318](#)
- virDrvDomainGetDiskErrors
  - driver.h, [318](#)
- virDrvDomainGetEmulatorPinInfo
  - driver.h, [318](#)
- virDrvDomainGetHostname
  - driver.h, [318](#)
- virDrvDomainGetInfo
  - driver.h, [318](#)
- virDrvDomainGetInterfaceParameters
  - driver.h, [318](#)
- virDrvDomainGetJobInfo
  - driver.h, [318](#)
- virDrvDomainGetMaxMemory
  - driver.h, [318](#)
- virDrvDomainGetMaxVcpus
  - driver.h, [318](#)
- virDrvDomainGetMemoryParameters
  - driver.h, [318](#)

- virDrvDomainGetMetadata  
driver.h, [318](#)
- virDrvDomainGetNumaParameters  
driver.h, [319](#)
- virDrvDomainGetOSType  
driver.h, [319](#)
- virDrvDomainGetSchedulerParameters  
driver.h, [319](#)
- virDrvDomainGetSchedulerParametersFlags  
driver.h, [319](#)
- virDrvDomainGetSchedulerType  
driver.h, [319](#)
- virDrvDomainGetSecurityLabel  
driver.h, [319](#)
- virDrvDomainGetSecurityLabelList  
driver.h, [319](#)
- virDrvDomainGetState  
driver.h, [319](#)
- virDrvDomainGetVcpuPinInfo  
driver.h, [319](#)
- virDrvDomainGetVcpus  
driver.h, [319](#)
- virDrvDomainGetVcpusFlags  
driver.h, [319](#)
- virDrvDomainGetXMLDesc  
driver.h, [319](#)
- virDrvDomainHasCurrentSnapshot  
driver.h, [319](#)
- virDrvDomainHasManagedSaveImage  
driver.h, [319](#)
- virDrvDomainInjectNMI  
driver.h, [319](#)
- virDrvDomainInterfaceStats  
driver.h, [319](#)
- virDrvDomainIsActive  
driver.h, [319](#)
- virDrvDomainIsPersistent  
driver.h, [319](#)
- virDrvDomainIsUpdated  
driver.h, [319](#)
- virDrvDomainListAllSnapshots  
driver.h, [319](#)
- virDrvDomainLookupByID  
driver.h, [319](#)
- virDrvDomainLookupByName  
driver.h, [319](#)
- virDrvDomainLookupByUUID  
driver.h, [319](#)
- virDrvDomainManagedSave  
driver.h, [319](#)
- virDrvDomainManagedSaveRemove  
driver.h, [319](#)
- virDrvDomainMemoryPeek  
driver.h, [320](#)
- virDrvDomainMemoryStats  
driver.h, [320](#)
- virDrvDomainMigrateBegin3  
driver.h, [320](#)
- virDrvDomainMigrateConfirm3  
driver.h, [320](#)
- virDrvDomainMigrateFinish  
driver.h, [320](#)
- virDrvDomainMigrateFinish2  
driver.h, [320](#)
- virDrvDomainMigrateFinish3  
driver.h, [320](#)
- virDrvDomainMigrateGetMaxSpeed  
driver.h, [320](#)
- virDrvDomainMigratePerform  
driver.h, [320](#)
- virDrvDomainMigratePerform3  
driver.h, [320](#)
- virDrvDomainMigratePrepare  
driver.h, [320](#)
- virDrvDomainMigratePrepare2  
driver.h, [320](#)
- virDrvDomainMigratePrepare3  
driver.h, [320](#)
- virDrvDomainMigratePrepareTunnel  
driver.h, [320](#)
- virDrvDomainMigratePrepareTunnel3  
driver.h, [320](#)
- virDrvDomainMigrateSetMaxDowntime  
driver.h, [320](#)
- virDrvDomainMigrateSetMaxSpeed  
driver.h, [320](#)
- virDrvDomainOpenConsole  
driver.h, [320](#)
- virDrvDomainOpenGraphics  
driver.h, [321](#)
- virDrvDomainPMSuspendForDuration  
driver.h, [321](#)
- virDrvDomainPMWakeup  
driver.h, [321](#)
- virDrvDomainPinEmulator  
driver.h, [321](#)
- virDrvDomainPinVcpu  
driver.h, [321](#)
- virDrvDomainPinVcpuFlags  
driver.h, [321](#)
- virDrvDomainQemuAgentCommand  
driver.h, [321](#)
- virDrvDomainQemuAttach  
driver.h, [321](#)
- virDrvDomainQemuMonitorCommand  
driver.h, [321](#)
- virDrvDomainReboot  
driver.h, [321](#)
- virDrvDomainReset  
driver.h, [321](#)
- virDrvDomainRestore  
driver.h, [321](#)
- virDrvDomainRestoreFlags  
driver.h, [321](#)
- virDrvDomainResume  
driver.h, [321](#)

- virDrvDomainRevertToSnapshot  
driver.h, [321](#)
- virDrvDomainSave  
driver.h, [321](#)
- virDrvDomainSaveFlags  
driver.h, [321](#)
- virDrvDomainSaveImageDefineXML  
driver.h, [321](#)
- virDrvDomainSaveImageGetXMLDesc  
driver.h, [321](#)
- virDrvDomainScreenshot  
driver.h, [321](#)
- virDrvDomainSendKey  
driver.h, [321](#)
- virDrvDomainSetAutostart  
driver.h, [321](#)
- virDrvDomainSetBlkioParameters  
driver.h, [321](#)
- virDrvDomainSetBlockIoTune  
driver.h, [322](#)
- virDrvDomainSetInterfaceParameters  
driver.h, [322](#)
- virDrvDomainSetMaxMemory  
driver.h, [322](#)
- virDrvDomainSetMemory  
driver.h, [322](#)
- virDrvDomainSetMemoryFlags  
driver.h, [322](#)
- virDrvDomainSetMemoryParameters  
driver.h, [322](#)
- virDrvDomainSetMetadata  
driver.h, [322](#)
- virDrvDomainSetNumaParameters  
driver.h, [322](#)
- virDrvDomainSetSchedulerParameters  
driver.h, [322](#)
- virDrvDomainSetSchedulerParametersFlags  
driver.h, [322](#)
- virDrvDomainSetVcpus  
driver.h, [322](#)
- virDrvDomainSetVcpusFlags  
driver.h, [322](#)
- virDrvDomainShutdown  
driver.h, [322](#)
- virDrvDomainShutdownFlags  
driver.h, [322](#)
- virDrvDomainSnapshotCreateXML  
driver.h, [322](#)
- virDrvDomainSnapshotCurrent  
driver.h, [322](#)
- virDrvDomainSnapshotDelete  
driver.h, [322](#)
- virDrvDomainSnapshotGetParent  
driver.h, [322](#)
- virDrvDomainSnapshotGetXMLDesc  
driver.h, [322](#)
- virDrvDomainSnapshotHasMetadata  
driver.h, [322](#)
- virDrvDomainSnapshotsCurrent  
driver.h, [322](#)
- virDrvDomainSnapshotListAllChildren  
driver.h, [322](#)
- virDrvDomainSnapshotListChildrenNames  
driver.h, [322](#)
- virDrvDomainSnapshotListNames  
driver.h, [323](#)
- virDrvDomainSnapshotLookupByName  
driver.h, [323](#)
- virDrvDomainSnapshotNum  
driver.h, [323](#)
- virDrvDomainSnapshotNumChildren  
driver.h, [323](#)
- virDrvDomainSuspend  
driver.h, [323](#)
- virDrvDomainUndefine  
driver.h, [323](#)
- virDrvDomainUndefineFlags  
driver.h, [323](#)
- virDrvDomainUpdateDeviceFlags  
driver.h, [323](#)
- virDrvDomainSupportsFeature  
driver.h, [323](#)
- virDrvGetCapabilities  
driver.h, [323](#)
- virDrvGetHostname  
driver.h, [323](#)
- virDrvGetLibVersion  
driver.h, [323](#)
- virDrvGetMaxVcpus  
driver.h, [323](#)
- virDrvGetSysinfo  
driver.h, [323](#)
- virDrvGetType  
driver.h, [323](#)
- virDrvGetURI  
driver.h, [323](#)
- virDrvGetVersion  
driver.h, [323](#)
- virDrvInterfaceChangeBegin  
driver.h, [323](#)
- virDrvInterfaceChangeCommit  
driver.h, [323](#)
- virDrvInterfaceChangeRollback  
driver.h, [323](#)
- virDrvInterfaceCreate  
driver.h, [323](#)
- virDrvInterfaceDefineXML  
driver.h, [323](#)
- virDrvInterfaceDestroy  
driver.h, [323](#)
- virDrvInterfaceGetXMLDesc  
driver.h, [323](#)
- virDrvInterfaceIsActive  
driver.h, [323](#)
- virDrvInterfaceLookupByMACString  
driver.h, [323](#)

- virDrvInterfaceLookupByName  
driver.h, [324](#)
- virDrvInterfaceUndefine  
driver.h, [324](#)
- virDrvListAllDomains  
driver.h, [324](#)
- virDrvListAllInterfaces  
driver.h, [324](#)
- virDrvListAllNetworks  
driver.h, [324](#)
- virDrvListAllSecrets  
driver.h, [324](#)
- virDrvListDefinedDomains  
driver.h, [324](#)
- virDrvListDefinedInterfaces  
driver.h, [324](#)
- virDrvListDefinedNetworks  
driver.h, [324](#)
- virDrvListDomains  
driver.h, [324](#)
- virDrvListInterfaces  
driver.h, [324](#)
- virDrvListNetworks  
driver.h, [324](#)
- virDrvListSecrets  
driver.h, [324](#)
- virDrvNWFilterDefineXML  
driver.h, [325](#)
- virDrvNWFilterGetXMLDesc  
driver.h, [325](#)
- virDrvNWFilterLookupByName  
driver.h, [325](#)
- virDrvNWFilterLookupByUUID  
driver.h, [326](#)
- virDrvNWFilterUndefine  
driver.h, [326](#)
- virDrvNetworkCreate  
driver.h, [324](#)
- virDrvNetworkCreateXML  
driver.h, [324](#)
- virDrvNetworkDefineXML  
driver.h, [324](#)
- virDrvNetworkDestroy  
driver.h, [324](#)
- virDrvNetworkGetAutostart  
driver.h, [324](#)
- virDrvNetworkGetBridgeName  
driver.h, [324](#)
- virDrvNetworkGetBridgeType  
driver.h, [324](#)
- virDrvNetworkGetXMLDesc  
driver.h, [324](#)
- virDrvNetworkIsActive  
driver.h, [324](#)
- virDrvNetworkIsPersistent  
driver.h, [324](#)
- virDrvNetworkLookupByName  
driver.h, [324](#)
- virDrvNetworkLookupByUUID  
driver.h, [324](#)
- virDrvNetworkSetAutostart  
driver.h, [324](#)
- virDrvNetworkUndefine  
driver.h, [324](#)
- virDrvNetworkUpdate  
driver.h, [324](#)
- virDrvNo  
driver.h, [328](#)
- virDrvNodeDeviceCreateXML  
driver.h, [325](#)
- virDrvNodeDeviceDestroy  
driver.h, [325](#)
- virDrvNodeDeviceDetach  
driver.h, [325](#)
- virDrvNodeDeviceReAttach  
driver.h, [325](#)
- virDrvNodeDeviceReset  
driver.h, [325](#)
- virDrvNodeGetCPUStats  
driver.h, [325](#)
- virDrvNodeGetCellsFreeMemory  
driver.h, [325](#)
- virDrvNodeGetFreeMemory  
driver.h, [325](#)
- virDrvNodeGetInfo  
driver.h, [325](#)
- virDrvNodeGetMemoryParameters  
driver.h, [325](#)
- virDrvNodeGetMemoryStats  
driver.h, [325](#)
- virDrvNodeGetSecurityModel  
driver.h, [325](#)
- virDrvNodeSetMemoryParameters  
driver.h, [325](#)
- virDrvNodeSuspendForDuration  
driver.h, [325](#)
- virDrvNumOfDefinedDomains  
driver.h, [325](#)
- virDrvNumOfDefinedInterfaces  
driver.h, [325](#)
- virDrvNumOfDefinedNetworks  
driver.h, [325](#)
- virDrvNumOfDomains  
driver.h, [325](#)
- virDrvNumOfInterfaces  
driver.h, [325](#)
- virDrvNumOfNetworks  
driver.h, [325](#)
- virDrvNumOfSecrets  
driver.h, [325](#)
- virDrvOpen  
driver.h, [326](#)
- virDrvOpenStatus  
driver.h, [329](#)
- virDrvSecretDefineXML  
driver.h, [326](#)



- virDrvSecretGetValue  
driver.h, [326](#)
- virDrvSecretGetXMLDesc  
driver.h, [326](#)
- virDrvSecretLookupByUUID  
driver.h, [326](#)
- virDrvSecretLookupByUsage  
driver.h, [326](#)
- virDrvSecretSetValue  
driver.h, [326](#)
- virDrvSecretUndefine  
driver.h, [326](#)
- virDrvSetKeepAlive  
driver.h, [326](#)
- virDrvStoragePoolBuild  
driver.h, [326](#)
- virDrvStoragePoolCreate  
driver.h, [326](#)
- virDrvStoragePoolCreateXML  
driver.h, [326](#)
- virDrvStoragePoolDefineXML  
driver.h, [326](#)
- virDrvStoragePoolDelete  
driver.h, [326](#)
- virDrvStoragePoolDestroy  
driver.h, [326](#)
- virDrvStoragePoolGetAutostart  
driver.h, [326](#)
- virDrvStoragePoolGetInfo  
driver.h, [326](#)
- virDrvStoragePoolGetXMLDesc  
driver.h, [326](#)
- virDrvStoragePoolsActive  
driver.h, [326](#)
- virDrvStoragePoolsPersistent  
driver.h, [326](#)
- virDrvStoragePoolListAllVolumes  
driver.h, [326](#)
- virDrvStoragePoolListVolumes  
driver.h, [326](#)
- virDrvStoragePoolLookupByName  
driver.h, [326](#)
- virDrvStoragePoolLookupByUUID  
driver.h, [326](#)
- virDrvStoragePoolLookupByVolume  
driver.h, [327](#)
- virDrvStoragePoolNumOfVolumes  
driver.h, [327](#)
- virDrvStoragePoolRefresh  
driver.h, [327](#)
- virDrvStoragePoolSetAutostart  
driver.h, [327](#)
- virDrvStoragePoolUndefine  
driver.h, [327](#)
- virDrvStorageVolCreateXML  
driver.h, [327](#)
- virDrvStorageVolCreateXMLFrom  
driver.h, [327](#)
- virDrvStorageVolDelete  
driver.h, [327](#)
- virDrvStorageVolDownload  
driver.h, [327](#)
- virDrvStorageVolGetInfo  
driver.h, [327](#)
- virDrvStorageVolGetPath  
driver.h, [327](#)
- virDrvStorageVolGetXMLDesc  
driver.h, [327](#)
- virDrvStorageVolLookupByKey  
driver.h, [327](#)
- virDrvStorageVolLookupByName  
driver.h, [327](#)
- virDrvStorageVolLookupByPath  
driver.h, [327](#)
- virDrvStorageVolResize  
driver.h, [327](#)
- virDrvStorageVolUpload  
driver.h, [327](#)
- virDrvStorageVolWipe  
driver.h, [327](#)
- virDrvStorageVolWipePattern  
driver.h, [327](#)
- virDrvStreamAbort  
driver.h, [327](#)
- virDrvStreamEventAddCallback  
driver.h, [327](#)
- virDrvStreamEventRemoveCallback  
driver.h, [327](#)
- virDrvStreamEventUpdateCallback  
driver.h, [327](#)
- virDrvStreamFinish  
driver.h, [327](#)
- virDrvStreamRecv  
driver.h, [327](#)
- virDrvStreamSend  
driver.h, [328](#)
- virDrvSupportsFeature  
libvirt.c, [393](#)
- virEventAddHandle  
libvirt.h, [202](#)
- virEventAddHandleFunc  
libvirt.h, [126](#)
- virEventAddTimeout  
libvirt.h, [202](#)
- virEventAddTimeoutFunc  
libvirt.h, [126](#)
- virEventHandleCallback  
libvirt.h, [126](#)
- virEventHandleType  
libvirt.h, [147](#)
- virEventRegisterDefaultImpl  
libvirt.h, [202](#)
- virEventRegisterImpl  
libvirt.h, [202](#)
- virEventRemoveHandle  
libvirt.h, [202](#)

- virEventRemoveHandleFunc
  - libvirt.h, [126](#)
- virEventRemoveTimeout
  - libvirt.h, [202](#)
- virEventRemoveTimeoutFunc
  - libvirt.h, [126](#)
- virEventRunDefaultImpl
  - libvirt.h, [202](#)
- virEventTimeoutCallback
  - libvirt.h, [127](#)
- virEventUpdateHandle
  - libvirt.h, [202](#)
- virEventUpdateHandleFunc
  - libvirt.h, [127](#)
- virEventUpdateTimeout
  - libvirt.h, [202](#)
- virEventUpdateTimeoutFunc
  - libvirt.h, [127](#)
- virFreeCallback
  - libvirt.h, [127](#)
- virGetVersion
  - libvirt.c, [393](#)
  - libvirt.h, [202](#)
- virInitialize
  - libvirt.c, [393](#)
  - libvirt.h, [202](#)
- virInterface
  - libvirt.h, [127](#)
- virInterfaceChangeBegin
  - libvirt.c, [393](#)
  - libvirt.h, [202](#)
- virInterfaceChangeCommit
  - libvirt.c, [394](#)
  - libvirt.h, [203](#)
- virInterfaceChangeRollback
  - libvirt.c, [394](#)
  - libvirt.h, [203](#)
- virInterfaceCreate
  - libvirt.c, [394](#)
  - libvirt.h, [203](#)
- virInterfaceDefineXML
  - libvirt.c, [394](#)
  - libvirt.h, [203](#)
- virInterfaceDestroy
  - libvirt.c, [394](#)
  - libvirt.h, [203](#)
- virInterfaceDriver
  - driver.h, [328](#)
- virInterfaceDriverPtr
  - driver.h, [328](#)
- virInterfaceDriverTab
  - libvirt.c, [420](#)
- virInterfaceDriverTabCount
  - libvirt.c, [420](#)
- virInterfaceFree
  - libvirt.c, [395](#)
  - libvirt.h, [204](#)
- virInterfaceGetConnect
  - libvirt.c, [395](#)
  - libvirt.h, [204](#)
- virInterfaceGetMACString
  - libvirt.c, [395](#)
  - libvirt.h, [204](#)
- virInterfaceGetName
  - libvirt.c, [395](#)
  - libvirt.h, [204](#)
- virInterfaceGetXMLDesc
  - libvirt.c, [395](#)
  - libvirt.h, [204](#)
- virInterfaceIsActive
  - libvirt.c, [395](#)
  - libvirt.h, [204](#)
- virInterfaceLookupByMACString
  - libvirt.c, [396](#)
  - libvirt.h, [205](#)
- virInterfaceLookupByName
  - libvirt.c, [396](#)
  - libvirt.h, [205](#)
- virInterfacePtr
  - libvirt.h, [127](#)
- virInterfaceRef
  - libvirt.c, [396](#)
  - libvirt.h, [205](#)
- virInterfaceUndefine
  - libvirt.c, [396](#)
  - libvirt.h, [205](#)
- virInterfaceXMLFlags
  - libvirt.h, [147](#)
- virKeyCodeSet
  - libvirt.h, [147](#)
- virLibConnError
  - libvirt.c, [341](#)
- virLibDomainError
  - libvirt.c, [341](#)
- virLibDomainSnapshotError
  - libvirt.c, [341](#)
- virLibInterfaceError
  - libvirt.c, [342](#)
- virLibNWFilterError
  - libvirt.c, [342](#)
- virLibNetworkError
  - libvirt.c, [342](#)
- virLibNodeDeviceError
  - libvirt.c, [342](#)
- virLibSecretError
  - libvirt.c, [342](#)
- virLibStoragePoolError
  - libvirt.c, [342](#)
- virLibStorageVolError
  - libvirt.c, [342](#)
- virLibStreamError
  - libvirt.c, [343](#)
- virLifecycleFromStringFunc
  - domain\_conf.h, [269](#)
- virLifecycleToStringFunc
  - domain\_conf.h, [269](#)

- virMemoryParameter
  - libvirt.h, [127](#)
- virMemoryParameterPtr
  - libvirt.h, [127](#)
- virMemoryParameterType
  - libvirt.h, [148](#)
- virNWFilter
  - libvirt.h, [129](#)
- virNWFilterDefineXML
  - libvirt.c, [404](#)
  - libvirt.h, [213](#)
- virNWFilterDriver
  - driver.h, [328](#)
- virNWFilterDriverPtr
  - driver.h, [328](#)
- virNWFilterDriverTab
  - libvirt.c, [420](#)
- virNWFilterDriverTabCount
  - libvirt.c, [420](#)
- virNWFilterFree
  - libvirt.c, [404](#)
  - libvirt.h, [213](#)
- virNWFilterGetName
  - libvirt.c, [404](#)
  - libvirt.h, [213](#)
- virNWFilterGetUUID
  - libvirt.c, [404](#)
  - libvirt.h, [213](#)
- virNWFilterGetUUIDString
  - libvirt.c, [405](#)
  - libvirt.h, [214](#)
- virNWFilterGetXMLDesc
  - libvirt.c, [405](#)
  - libvirt.h, [214](#)
- virNWFilterLookupByName
  - libvirt.c, [405](#)
  - libvirt.h, [214](#)
- virNWFilterLookupByUUID
  - libvirt.c, [405](#)
  - libvirt.h, [214](#)
- virNWFilterLookupByUUIDString
  - libvirt.c, [405](#)
  - libvirt.h, [214](#)
- virNWFilterPtr
  - libvirt.h, [129](#)
- virNWFilterRef
  - libvirt.c, [405](#)
  - libvirt.h, [214](#)
- virNWFilterUndefine
  - libvirt.c, [405](#)
  - libvirt.h, [214](#)
- virNetDevBridgeAddPort
  - virnetdevbridge.c, [440](#)
- virNetDevBridgeCreate
  - virnetdevbridge.c, [440](#)
- virNetDevBridgeDelete
  - virnetdevbridge.c, [440](#)
- virNetDevBridgeGetSTP
  - virnetdevbridge.c, [441](#)
- virNetDevBridgeGetSTPDelay
  - virnetdevbridge.c, [441](#)
- virNetDevBridgeRemovePort
  - virnetdevbridge.c, [441](#)
- virNetDevBridgeSetSTP
  - virnetdevbridge.c, [441](#)
- virNetDevBridgeSetSTPDelay
  - virnetdevbridge.c, [441](#)
- virNetDevOpenvswitchAddPort
  - virnetdevopenvswitch.c, [442](#)
- virNetDevOpenvswitchBridgeCreate
  - virnetdevopenvswitch.c, [442](#)
  - virnetdevopenvswitch.h, [444](#)
- virNetDevOpenvswitchBridgeDelete
  - virnetdevopenvswitch.c, [442](#)
  - virnetdevopenvswitch.h, [444](#)
- virNetDevOpenvswitchBridgeGetSTP
  - virnetdevopenvswitch.c, [442](#)
  - virnetdevopenvswitch.h, [444](#)
- virNetDevOpenvswitchBridgeGetSTPDelay
  - virnetdevopenvswitch.c, [442](#)
  - virnetdevopenvswitch.h, [445](#)
- virNetDevOpenvswitchBridgeSetSTP
  - virnetdevopenvswitch.c, [442](#)
  - virnetdevopenvswitch.h, [445](#)
- virNetDevOpenvswitchBridgeSetSTPDelay
  - virnetdevopenvswitch.c, [443](#)
  - virnetdevopenvswitch.h, [445](#)
- virNetDevOpenvswitchFakeBridgeCreate
  - virnetdevopenvswitch.c, [443](#)
  - virnetdevopenvswitch.h, [445](#)
- virNetDevOpenvswitchRemovePort
  - virnetdevopenvswitch.c, [443](#)
  - virnetdevopenvswitch.h, [445](#)
- virNetDevOpenvswitchTunnelCreate
  - virnetdevopenvswitch.c, [443](#)
  - virnetdevopenvswitch.h, [445](#)
- virNetDevOpenvswitchTunnelDestroy
  - virnetdevopenvswitch.c, [443](#)
  - virnetdevopenvswitch.h, [445](#)
- virNetDevTapCreate
  - virnetdevtap.c, [446](#)
  - virnetdevtap.h, [448](#)
- virNetDevTapCreateFlags
  - virnetdevtap.h, [447](#)
- virNetDevTapCreateInBridgePort
  - virnetdevtap.c, [446](#)
  - virnetdevtap.h, [448](#)
- virNetDevTapDelete
  - virnetdevtap.c, [447](#)
  - virnetdevtap.h, [448](#)
- virNetwork
  - libvirt.h, [127](#)
- virNetworkAllocateBridge
  - network\_conf.c, [295](#)
  - network\_conf.h, [305](#)
- virNetworkAssignDef

- network\_conf.c, [295](#)
- network\_conf.h, [305](#)
- virNetworkBridgeInUse
  - network\_conf.c, [295](#)
  - network\_conf.h, [305](#)
- virNetworkConfigChangeSetup
  - network\_conf.c, [295](#)
  - network\_conf.h, [305](#)
- virNetworkConfigFile
  - network\_conf.c, [296](#)
  - network\_conf.h, [305](#)
- virNetworkCreate
  - libvirt.c, [396](#)
  - libvirt.h, [205](#)
- virNetworkCreateXML
  - libvirt.c, [396](#)
  - libvirt.h, [205](#)
- virNetworkDHCPDefParse
  - network\_conf.c, [298](#)
- virNetworkDHCPHostDef
  - network\_conf.h, [304](#)
- virNetworkDHCPHostDefClear
  - network\_conf.c, [298](#)
- virNetworkDHCPHostDefParse
  - network\_conf.c, [298](#)
- virNetworkDHCPHostDefPtr
  - network\_conf.h, [304](#)
- virNetworkDHCPRangeDef
  - network\_conf.h, [304](#)
- virNetworkDHCPRangeDefParse
  - network\_conf.c, [298](#)
- virNetworkDHCPRangeDefPtr
  - network\_conf.h, [304](#)
- virNetworkDNSDefFormat
  - network\_conf.c, [298](#)
- virNetworkDNSDefFree
  - network\_conf.c, [298](#)
- virNetworkDNSDefParseXML
  - network\_conf.c, [298](#)
- virNetworkDNSDefPtr
  - network\_conf.h, [304](#)
- virNetworkDNSHostsDefParseXML
  - network\_conf.c, [298](#)
- virNetworkDNSHostsDefPtr
  - network\_conf.h, [304](#)
- virNetworkDNSSrvDefParseXML
  - network\_conf.c, [298](#)
- virNetworkDNSSrvRecordsDef
  - network\_conf.h, [304](#)
- virNetworkDNSSrvRecordsDefPtr
  - network\_conf.h, [304](#)
- virNetworkDNSTxtRecordsDef
  - network\_conf.h, [304](#)
- virNetworkDNSTxtRecordsDefPtr
  - network\_conf.h, [304](#)
- virNetworkDef
  - network\_conf.h, [304](#)
- virNetworkDefCopy
  - network\_conf.c, [296](#)
  - network\_conf.h, [305](#)
- virNetworkDefFormat
  - network\_conf.c, [296](#)
  - network\_conf.h, [305](#)
- virNetworkDefForwardIf
  - network\_conf.h, [306](#)
- virNetworkDefFree
  - network\_conf.c, [296](#)
  - network\_conf.h, [306](#)
- virNetworkDefGetIpByIndex
  - network\_conf.c, [296](#)
  - network\_conf.h, [306](#)
- virNetworkDefParse
  - network\_conf.c, [296](#)
- virNetworkDefParseFile
  - network\_conf.c, [296](#)
  - network\_conf.h, [306](#)
- virNetworkDefParseNode
  - network\_conf.c, [296](#)
  - network\_conf.h, [306](#)
- virNetworkDefParseString
  - network\_conf.c, [296](#)
  - network\_conf.h, [306](#)
- virNetworkDefParseXML
  - network\_conf.c, [296](#)
- virNetworkDefPtr
  - network\_conf.h, [304](#)
- virNetworkDefUpdateBridge
  - network\_conf.c, [297](#)
- virNetworkDefUpdateCheckElementName
  - network\_conf.c, [297](#)
- virNetworkDefUpdateDNSHost
  - network\_conf.c, [297](#)
- virNetworkDefUpdateDNSSrv
  - network\_conf.c, [297](#)
- virNetworkDefUpdateDNSTxt
  - network\_conf.c, [297](#)
- virNetworkDefUpdateDomain
  - network\_conf.c, [297](#)
- virNetworkDefUpdateForward
  - network\_conf.c, [297](#)
- virNetworkDefUpdateForwardInterface
  - network\_conf.c, [297](#)
- virNetworkDefUpdateForwardPF
  - network\_conf.c, [297](#)
- virNetworkDefUpdateIP
  - network\_conf.c, [297](#)
- virNetworkDefUpdateIPDHCPHost
  - network\_conf.c, [297](#)
- virNetworkDefUpdateIPDHCPRange
  - network\_conf.c, [297](#)
- virNetworkDefUpdateNoSupport
  - network\_conf.c, [297](#)
- virNetworkDefUpdatePortGroup
  - network\_conf.c, [297](#)
- virNetworkDefUpdateSection
  - network\_conf.c, [297](#)

- virNetworkDefUpdateTunnel
  - network\_conf.c, [298](#)
- virNetworkDefUpdateUnknownCommand
  - network\_conf.c, [298](#)
- virNetworkDefineXML
  - libvirt.c, [397](#)
  - libvirt.h, [206](#)
- virNetworkDeleteConfig
  - network\_conf.c, [298](#)
  - network\_conf.h, [306](#)
- virNetworkDestroy
  - libvirt.c, [397](#)
  - libvirt.h, [206](#)
- virNetworkDriver
  - driver.h, [328](#)
- virNetworkDriverPtr
  - driver.h, [328](#)
- virNetworkDriverTab
  - libvirt.c, [420](#)
- virNetworkDriverTabCount
  - libvirt.c, [420](#)
- virNetworkFindByName
  - network\_conf.c, [298](#)
  - network\_conf.h, [306](#)
- virNetworkFindByUUID
  - network\_conf.h, [306](#)
- virNetworkForwardHostdevDeviceType
  - network\_conf.h, [305](#)
- virNetworkForwardIfDef
  - network\_conf.h, [304](#)
- virNetworkForwardIfDefClear
  - network\_conf.c, [299](#)
- virNetworkForwardIfDefPtr
  - network\_conf.h, [304](#)
- virNetworkForwardPfDef
  - network\_conf.h, [304](#)
- virNetworkForwardPfDefClear
  - network\_conf.c, [299](#)
- virNetworkForwardPfDefPtr
  - network\_conf.h, [304](#)
- virNetworkForwardType
  - network\_conf.h, [305](#)
- virNetworkFree
  - libvirt.c, [397](#)
  - libvirt.h, [206](#)
- virNetworkGetAutostart
  - libvirt.c, [397](#)
  - libvirt.h, [206](#)
- virNetworkGetBridgeName
  - libvirt.c, [397](#)
  - libvirt.h, [206](#)
- virNetworkGetBridgeType
  - libvirt.c, [397](#)
  - libvirt.h, [206](#)
- virNetworkGetConnect
  - libvirt.c, [397](#)
  - libvirt.h, [206](#)
- virNetworkGetName
  - libvirt.c, [398](#)
  - libvirt.h, [207](#)
- virNetworkGetUUID
  - libvirt.c, [398](#)
  - libvirt.h, [207](#)
- virNetworkGetUUIDString
  - libvirt.c, [398](#)
  - libvirt.h, [207](#)
- virNetworkGetXMLDesc
  - libvirt.c, [398](#)
  - libvirt.h, [207](#)
- virNetworkIPParseXML
  - network\_conf.c, [299](#)
- virNetworkIpDef
  - network\_conf.h, [304](#)
- virNetworkIpDefByIndex
  - network\_conf.c, [299](#)
- virNetworkIpDefClear
  - network\_conf.c, [299](#)
- virNetworkIpDefFormat
  - network\_conf.c, [299](#)
- virNetworkIpDefNetmask
  - network\_conf.c, [299](#)
  - network\_conf.h, [306](#)
- virNetworkIpDefPrefix
  - network\_conf.c, [299](#)
  - network\_conf.h, [306](#)
- virNetworkIpDefPtr
  - network\_conf.h, [304](#)
- virNetworkIsActive
  - libvirt.c, [398](#)
  - libvirt.h, [207](#)
- virNetworkIsPersistent
  - libvirt.c, [398](#)
  - libvirt.h, [207](#)
- virNetworkList
  - network\_conf.c, [299](#)
  - network\_conf.h, [306](#)
- virNetworkLoadAllConfigs
  - network\_conf.c, [299](#)
  - network\_conf.h, [306](#)
- virNetworkLoadConfig
  - network\_conf.c, [299](#)
  - network\_conf.h, [306](#)
- virNetworkLookupByName
  - libvirt.c, [399](#)
  - libvirt.h, [208](#)
- virNetworkLookupByUUID
  - libvirt.c, [399](#)
  - libvirt.h, [208](#)
- virNetworkLookupByUUIDString
  - libvirt.c, [399](#)
  - libvirt.h, [208](#)
- virNetworkMatch
  - network\_conf.c, [299](#)
- virNetworkObj
  - network\_conf.h, [304](#)
- virNetworkObjAssignDef

- network\_conf.c, 299
- network\_conf.h, 306
- virNetworkObjFree
  - network\_conf.c, 299
  - network\_conf.h, 306
- virNetworkObjGetPersistentDef
  - network\_conf.c, 299
  - network\_conf.h, 306
- virNetworkObjsActive
  - network\_conf.h, 306
- virNetworkObjsDuplicate
  - network\_conf.c, 299
  - network\_conf.h, 306
- virNetworkObjList
  - network\_conf.h, 304
- virNetworkObjListFree
  - network\_conf.c, 299
  - network\_conf.h, 306
- virNetworkObjListPtr
  - network\_conf.h, 304
- virNetworkObjLock
  - network\_conf.c, 299
  - network\_conf.h, 307
- virNetworkObjPtr
  - network\_conf.h, 304
- virNetworkObjReplacePersistentDef
  - network\_conf.c, 299
  - network\_conf.h, 307
- virNetworkObjSetDefTransient
  - network\_conf.c, 299
  - network\_conf.h, 307
- virNetworkObjUnlock
  - network\_conf.c, 299
  - network\_conf.h, 307
- virNetworkObjUnsetDefTransient
  - network\_conf.c, 299
  - network\_conf.h, 307
- virNetworkObjUpdate
  - network\_conf.c, 299
  - network\_conf.h, 307
- virNetworkPortGroupParseXML
  - network\_conf.c, 299
- virNetworkPtr
  - libvirt.h, 128
- virNetworkRef
  - libvirt.c, 399
  - libvirt.h, 208
- virNetworkRemoveInactive
  - network\_conf.c, 299
  - network\_conf.h, 307
- virNetworkSaveConfig
  - network\_conf.c, 299
  - network\_conf.h, 307
- virNetworkSaveStatus
  - network\_conf.c, 300
  - network\_conf.h, 307
- virNetworkSaveXML
  - network\_conf.c, 300
- network\_conf.h, 307
- virNetworkSetAutostart
  - libvirt.c, 399
  - libvirt.h, 208
- virNetworkSetBridgeMacAddr
  - network\_conf.c, 300
  - network\_conf.h, 307
- virNetworkSetBridgeName
  - network\_conf.c, 300
  - network\_conf.h, 307
- virNetworkTunnelParseXML
  - network\_conf.c, 300
- virNetworkUndefine
  - libvirt.c, 399
  - libvirt.h, 208
- virNetworkUpdate
  - libvirt.c, 399
  - libvirt.h, 208
- virNetworkUpdateCommand
  - libvirt.h, 148
- virNetworkUpdateFlags
  - libvirt.h, 148
- virNetworkUpdateSection
  - libvirt.h, 148
- virNetworkXMLFlags
  - libvirt.h, 149
- virNodeCPUStats
  - libvirt.h, 128
- virNodeCPUStatsPtr
  - libvirt.h, 128
- virNodeDevice
  - libvirt.h, 128
- virNodeDeviceCreateXML
  - libvirt.c, 399
  - libvirt.h, 208
- virNodeDeviceDestroy
  - libvirt.c, 400
  - libvirt.h, 209
- virNodeDeviceDetach
  - libvirt.c, 400
  - libvirt.h, 209
- virNodeDeviceFree
  - libvirt.c, 400
  - libvirt.h, 209
- virNodeDeviceGetName
  - libvirt.c, 400
  - libvirt.h, 209
- virNodeDeviceGetParent
  - libvirt.c, 400
  - libvirt.h, 209
- virNodeDeviceGetXMLDesc
  - libvirt.c, 400
  - libvirt.h, 209
- virNodeDeviceListCaps
  - libvirt.c, 401
  - libvirt.h, 210
- virNodeDeviceLookupByName
  - libvirt.c, 401

- libvirt.h, [210](#)
- virNodeDeviceNumOfCaps
  - libvirt.c, [401](#)
  - libvirt.h, [210](#)
- virNodeDevicePtr
  - libvirt.h, [128](#)
- virNodeDeviceReAttach
  - libvirt.c, [401](#)
  - libvirt.h, [210](#)
- virNodeDeviceRef
  - libvirt.c, [401](#)
  - libvirt.h, [210](#)
- virNodeDeviceReset
  - libvirt.c, [401](#)
  - libvirt.h, [210](#)
- virNodeGetCPUStats
  - libvirt.c, [402](#)
  - libvirt.h, [211](#)
- virNodeGetCPUStatsAllCPUs
  - libvirt.h, [149](#)
- virNodeGetCellsFreeMemory
  - libvirt.c, [401](#)
  - libvirt.h, [210](#)
- virNodeGetFreeMemory
  - libvirt.c, [402](#)
  - libvirt.h, [211](#)
- virNodeGetInfo
  - libvirt.c, [402](#)
  - libvirt.h, [211](#)
- virNodeGetMemoryParameters
  - libvirt.c, [403](#)
  - libvirt.h, [212](#)
- virNodeGetMemoryStats
  - libvirt.c, [403](#)
  - libvirt.h, [212](#)
- virNodeGetMemoryStatsAllCells
  - libvirt.h, [149](#)
- virNodeGetSecurityModel
  - libvirt.c, [403](#)
  - libvirt.h, [212](#)
- virNodeInfo
  - libvirt.h, [128](#)
- virNodeInfoPtr
  - libvirt.h, [128](#)
- virNodeListDevices
  - libvirt.c, [403](#)
  - libvirt.h, [212](#)
- virNodeMemoryStats
  - libvirt.h, [128](#)
- virNodeMemoryStatsPtr
  - libvirt.h, [128](#)
- virNodeNumOfDevices
  - libvirt.c, [403](#)
  - libvirt.h, [212](#)
- virNodeSetMemoryParameters
  - libvirt.c, [404](#)
  - libvirt.h, [213](#)
- virNodeSuspendForDuration
  - libvirt.c, [404](#)
  - libvirt.h, [213](#)
- virNodeSuspendTarget
  - libvirt.h, [149](#)
- virPortGroupDef
  - network\_conf.h, [304](#)
- virPortGroupDefClear
  - network\_conf.c, [300](#)
- virPortGroupDefFormat
  - network\_conf.c, [300](#)
- virPortGroupDefPtr
  - network\_conf.h, [304](#)
- virPortGroupFindByName
  - network\_conf.c, [300](#)
  - network\_conf.h, [307](#)
- virRegisterDeviceMonitor
  - driver.h, [329](#)
  - libvirt.c, [405](#)
- virRegisterDriver
  - driver.h, [329](#)
  - libvirt.c, [406](#)
- virRegisterInterfaceDriver
  - driver.h, [329](#)
  - libvirt.c, [406](#)
- virRegisterNWFilterDriver
  - driver.h, [329](#)
  - libvirt.c, [406](#)
- virRegisterNetworkDriver
  - driver.h, [329](#)
  - libvirt.c, [406](#)
- virRegisterSecretDriver
  - driver.h, [329](#)
  - libvirt.c, [406](#)
- virRegisterStorageDriver
  - driver.h, [330](#)
  - libvirt.c, [406](#)
- virSchedParameter
  - libvirt.h, [129](#)
- virSchedParameterPtr
  - libvirt.h, [129](#)
- virSchedParameterType
  - libvirt.h, [150](#)
- virSecret
  - libvirt.h, [129](#)
- virSecretDefineXML
  - libvirt.c, [406](#)
  - libvirt.h, [214](#)
- virSecretDriver
  - driver.h, [328](#)
- virSecretDriverPtr
  - driver.h, [328](#)
- virSecretDriverTab
  - libvirt.c, [420](#)
- virSecretDriverTabCount
  - libvirt.c, [420](#)
- virSecretFree
  - libvirt.c, [407](#)
  - libvirt.h, [215](#)

- virSecretGetConnect
  - libvirt.c, [407](#)
  - libvirt.h, [215](#)
- virSecretGetUUID
  - libvirt.c, [407](#)
  - libvirt.h, [215](#)
- virSecretGetUUIDString
  - libvirt.c, [407](#)
  - libvirt.h, [215](#)
- virSecretGetUsageID
  - libvirt.c, [407](#)
  - libvirt.h, [215](#)
- virSecretGetUsageType
  - libvirt.c, [407](#)
  - libvirt.h, [215](#)
- virSecretGetValue
  - libvirt.c, [408](#)
  - libvirt.h, [216](#)
- virSecretGetXMLDesc
  - libvirt.c, [408](#)
  - libvirt.h, [216](#)
- virSecretLookupByUUID
  - libvirt.c, [408](#)
  - libvirt.h, [216](#)
- virSecretLookupByUUIDString
  - libvirt.c, [408](#)
  - libvirt.h, [216](#)
- virSecretLookupByUsage
  - libvirt.c, [408](#)
  - libvirt.h, [216](#)
- virSecretPtr
  - libvirt.h, [129](#)
- virSecretRef
  - libvirt.c, [408](#)
  - libvirt.h, [216](#)
- virSecretSetValue
  - libvirt.c, [409](#)
  - libvirt.h, [217](#)
- virSecretUndefine
  - libvirt.c, [409](#)
  - libvirt.h, [217](#)
- virSecretUsageType
  - libvirt.h, [150](#)
- virSecurityDeviceLabelDef
  - domain\_conf.h, [269](#)
- virSecurityDeviceLabelDefFormat
  - domain\_conf.c, [250](#)
- virSecurityDeviceLabelDefFree
  - domain\_conf.c, [250](#)
- virSecurityDeviceLabelDefParseXML
  - domain\_conf.c, [250](#)
- virSecurityDeviceLabelDefPtr
  - domain\_conf.h, [269](#)
- virSecurityLabel
  - libvirt.h, [129](#)
- virSecurityLabelDef
  - domain\_conf.h, [269](#)
- virSecurityLabelDefFormat
  - domain\_conf.c, [250](#)
- virSecurityLabelDefFree
  - domain\_conf.c, [250](#)
- virSecurityLabelDefParseXML
  - domain\_conf.c, [250](#)
- virSecurityLabelDefPtr
  - domain\_conf.h, [269](#)
- virSecurityLabelDefsParseXML
  - domain\_conf.c, [250](#)
- virSecurityLabelPtr
  - libvirt.h, [129](#)
- virSecurityModel
  - libvirt.h, [129](#)
- virSecurityModelPtr
  - libvirt.h, [129](#)
- virStorageDriver
  - driver.h, [328](#)
- virStorageDriverPtr
  - driver.h, [328](#)
- virStorageDriverTab
  - libvirt.c, [420](#)
- virStorageDriverTabCount
  - libvirt.c, [420](#)
- virStoragePool
  - libvirt.h, [130](#)
- virStoragePoolBuild
  - libvirt.c, [409](#)
  - libvirt.h, [217](#)
- virStoragePoolBuildFlags
  - libvirt.h, [150](#)
- virStoragePoolCreate
  - libvirt.c, [409](#)
  - libvirt.h, [217](#)
- virStoragePoolCreateXML
  - libvirt.c, [409](#)
  - libvirt.h, [217](#)
- virStoragePoolDefineXML
  - libvirt.c, [409](#)
  - libvirt.h, [217](#)
- virStoragePoolDelete
  - libvirt.c, [409](#)
  - libvirt.h, [218](#)
- virStoragePoolDeleteFlags
  - libvirt.h, [150](#)
- virStoragePoolDestroy
  - libvirt.c, [410](#)
  - libvirt.h, [218](#)
- virStoragePoolFree
  - libvirt.c, [410](#)
  - libvirt.h, [218](#)
- virStoragePoolGetAutostart
  - libvirt.c, [410](#)
  - libvirt.h, [218](#)
- virStoragePoolGetConnect
  - libvirt.c, [410](#)
  - libvirt.h, [218](#)
- virStoragePoolGetInfo
  - libvirt.c, [410](#)



- libvirt.h, [218](#)
- virStoragePoolGetName
  - libvirt.c, [410](#)
  - libvirt.h, [218](#)
- virStoragePoolGetUUID
  - libvirt.c, [410](#)
  - libvirt.h, [219](#)
- virStoragePoolGetUUIDString
  - libvirt.c, [411](#)
  - libvirt.h, [219](#)
- virStoragePoolGetXMLDesc
  - libvirt.c, [411](#)
  - libvirt.h, [219](#)
- virStoragePoolInfo
  - libvirt.h, [130](#)
- virStoragePoolInfoPtr
  - libvirt.h, [130](#)
- virStoragePoolsActive
  - libvirt.c, [411](#)
  - libvirt.h, [219](#)
- virStoragePoolsPersistent
  - libvirt.c, [411](#)
  - libvirt.h, [219](#)
- virStoragePoolListAllVolumes
  - libvirt.c, [411](#)
  - libvirt.h, [219](#)
- virStoragePoolListVolumes
  - libvirt.c, [411](#)
  - libvirt.h, [219](#)
- virStoragePoolLookupByName
  - libvirt.c, [412](#)
  - libvirt.h, [220](#)
- virStoragePoolLookupByUUID
  - libvirt.c, [412](#)
  - libvirt.h, [220](#)
- virStoragePoolLookupByUUIDString
  - libvirt.c, [412](#)
  - libvirt.h, [220](#)
- virStoragePoolLookupByVolume
  - libvirt.c, [412](#)
  - libvirt.h, [220](#)
- virStoragePoolNumOfVolumes
  - libvirt.c, [412](#)
  - libvirt.h, [220](#)
- virStoragePoolPtr
  - libvirt.h, [130](#)
- virStoragePoolRef
  - libvirt.c, [412](#)
  - libvirt.h, [220](#)
- virStoragePoolRefresh
  - libvirt.c, [412](#)
  - libvirt.h, [220](#)
- virStoragePoolSetAutostart
  - libvirt.c, [412](#)
  - libvirt.h, [221](#)
- virStoragePoolState
  - libvirt.h, [150](#)
- virStoragePoolUndefine
  - libvirt.c, [413](#)
  - libvirt.h, [221](#)
- virStorageVol
  - libvirt.h, [130](#)
- virStorageVolCreateXML
  - libvirt.c, [413](#)
  - libvirt.h, [221](#)
- virStorageVolCreateXMLFrom
  - libvirt.c, [413](#)
  - libvirt.h, [221](#)
- virStorageVolDelete
  - libvirt.c, [413](#)
  - libvirt.h, [221](#)
- virStorageVolDeleteFlags
  - libvirt.h, [151](#)
- virStorageVolDownload
  - libvirt.c, [413](#)
  - libvirt.h, [221](#)
- virStorageVolFree
  - libvirt.c, [413](#)
  - libvirt.h, [222](#)
- virStorageVolGetConnect
  - libvirt.c, [414](#)
  - libvirt.h, [222](#)
- virStorageVolGetInfo
  - libvirt.c, [414](#)
  - libvirt.h, [222](#)
- virStorageVolGetKey
  - libvirt.c, [414](#)
  - libvirt.h, [222](#)
- virStorageVolGetName
  - libvirt.c, [414](#)
  - libvirt.h, [222](#)
- virStorageVolGetPath
  - libvirt.c, [414](#)
  - libvirt.h, [222](#)
- virStorageVolGetXMLDesc
  - libvirt.c, [414](#)
  - libvirt.h, [222](#)
- virStorageVolInfo
  - libvirt.h, [130](#)
- virStorageVolInfoPtr
  - libvirt.h, [130](#)
- virStorageVolLookupByKey
  - libvirt.c, [414](#)
  - libvirt.h, [223](#)
- virStorageVolLookupByName
  - libvirt.c, [415](#)
  - libvirt.h, [223](#)
- virStorageVolLookupByPath
  - libvirt.c, [415](#)
  - libvirt.h, [223](#)
- virStorageVolPtr
  - libvirt.h, [130](#)
- virStorageVolRef
  - libvirt.c, [415](#)
  - libvirt.h, [223](#)
- virStorageVolResize

- libvirt.c, 415
- libvirt.h, 223
- virStorageVolResizeFlags
  - libvirt.h, 151
- virStorageVolType
  - libvirt.h, 151
- virStorageVolUpload
  - libvirt.c, 415
  - libvirt.h, 223
- virStorageVolWipe
  - libvirt.c, 416
  - libvirt.h, 224
- virStorageVolWipeAlgorithm
  - libvirt.h, 151
- virStorageVolWipePattern
  - libvirt.c, 416
  - libvirt.h, 224
- virStorageXMLFlags
  - libvirt.h, 151
- virStream
  - libvirt.h, 130
- virStreamAbort
  - libvirt.c, 416
  - libvirt.h, 224
- virStreamDriver
  - driver.h, 328
- virStreamDriverPtr
  - driver.h, 328
- virStreamEventAddCallback
  - libvirt.c, 416
  - libvirt.h, 224
- virStreamEventCallback
  - libvirt.h, 130
- virStreamEventRemoveCallback
  - libvirt.c, 416
  - libvirt.h, 224
- virStreamEventType
  - libvirt.h, 152
- virStreamEventUpdateCallback
  - libvirt.c, 416
  - libvirt.h, 224
- virStreamFinish
  - libvirt.c, 416
  - libvirt.h, 225
- virStreamFlags
  - libvirt.h, 152
- virStreamFree
  - libvirt.c, 417
  - libvirt.h, 225
- virStreamNew
  - libvirt.c, 417
  - libvirt.h, 225
- virStreamPtr
  - libvirt.h, 130
- virStreamRecv
  - libvirt.c, 417
  - libvirt.h, 225
- virStreamRecvAll
  - libvirt.c, 417
  - libvirt.h, 226
- virStreamRef
  - libvirt.c, 418
  - libvirt.h, 226
- virStreamSend
  - libvirt.c, 418
  - libvirt.h, 226
- virStreamSendAll
  - libvirt.c, 418
  - libvirt.h, 227
- virStreamSinkFunc
  - libvirt.h, 131
- virStreamSourceFunc
  - libvirt.h, 131
- virSysinfoParseXML
  - domain\_conf.c, 250
- virTLSMutexDestroy
  - libvirt.c, 419
- virTLSMutexInit
  - libvirt.c, 419
- virTLSMutexLock
  - libvirt.c, 419
- virTLSMutexUnlock
  - libvirt.c, 419
- virTLSThreadImpl
  - libvirt.c, 420
- virTunnelDef
  - network\_conf.h, 304
- virTunnelDefFormat
  - network\_conf.c, 300
- virTunnelDefFree
  - network\_conf.c, 300
- virTunnelDefPtr
  - network\_conf.h, 305
- virTypedParameter
  - libvirt.h, 131
- virTypedParameterFlags
  - libvirt.h, 152
- virTypedParameterPtr
  - libvirt.h, 131
- virTypedParameterType
  - libvirt.h, 152
- virTypedParameterValidateSet
  - libvirt.c, 419
- virVcpuInfo
  - libvirt.h, 131
- virVcpuInfoPtr
  - libvirt.h, 131
- virVcpuState
  - libvirt.h, 152
- virnetdevbridge.c
  - VIR\_FROM\_THIS, 440
  - virNetDevBridgeAddPort, 440
  - virNetDevBridgeCreate, 440
  - virNetDevBridgeDelete, 440
  - virNetDevBridgeGetSTP, 441
  - virNetDevBridgeGetSTPDelay, 441

- virNetDevBridgeRemovePort, [441](#)
- virNetDevBridgeSetSTP, [441](#)
- virNetDevBridgeSetSTPDelay, [441](#)
- virnetdevopenvswitch.c
  - VIR\_FROM\_THIS, [442](#)
  - virNetDevOpenvswitchAddPort, [442](#)
  - virNetDevOpenvswitchBridgeCreate, [442](#)
  - virNetDevOpenvswitchBridgeDelete, [442](#)
  - virNetDevOpenvswitchBridgeGetSTP, [442](#)
  - virNetDevOpenvswitchBridgeGetSTPDelay, [442](#)
  - virNetDevOpenvswitchBridgeSetSTP, [442](#)
  - virNetDevOpenvswitchBridgeSetSTPDelay, [443](#)
  - virNetDevOpenvswitchFakeBridgeCreate, [443](#)
  - virNetDevOpenvswitchRemovePort, [443](#)
  - virNetDevOpenvswitchTunnelCreate, [443](#)
  - virNetDevOpenvswitchTunnelDestroy, [443](#)
- virnetdevopenvswitch.h
  - ATTRIBUTE\_NONNULL, [444](#)
  - ifname, [445](#)
  - macaddr, [445](#)
  - ovsport, [445](#)
  - virNetDevOpenvswitchBridgeCreate, [444](#)
  - virNetDevOpenvswitchBridgeDelete, [444](#)
  - virNetDevOpenvswitchBridgeGetSTP, [444](#)
  - virNetDevOpenvswitchBridgeGetSTPDelay, [445](#)
  - virNetDevOpenvswitchBridgeSetSTP, [445](#)
  - virNetDevOpenvswitchBridgeSetSTPDelay, [445](#)
  - virNetDevOpenvswitchFakeBridgeCreate, [445](#)
  - virNetDevOpenvswitchRemovePort, [445](#)
  - virNetDevOpenvswitchTunnelCreate, [445](#)
  - virNetDevOpenvswitchTunnelDestroy, [445](#)
  - vmuuid, [445](#)
- virnetdevtap.h
  - VIR\_NETDEV\_TAP\_CREATE\_IFUP, [447](#)
  - VIR\_NETDEV\_TAP\_CREATE\_NONE, [447](#)
  - VIR\_NETDEV\_TAP\_CREATE\_PERSIST, [448](#)
  - VIR\_NETDEV\_TAP\_CREATE\_USE\_MAC\_FOR\_BRIDGE, [448](#)
  - VIR\_NETDEV\_TAP\_CREATE\_VNET\_HDR, [447](#)
- virnetdevtap.c
  - VIR\_FROM\_THIS, [446](#)
  - virNetDevTapCreate, [446](#)
  - virNetDevTapCreateInBridgePort, [446](#)
  - virNetDevTapDelete, [447](#)
- virnetdevtap.h
  - virNetDevTapCreate, [448](#)
  - virNetDevTapCreateFlags, [447](#)
  - virNetDevTapCreateInBridgePort, [448](#)
  - virNetDevTapDelete, [448](#)
- virsh-network.c
  - cmdNetworkAutostart, [450](#)
  - cmdNetworkCreate, [450](#)
  - cmdNetworkDefine, [450](#)
  - cmdNetworkDestroy, [450](#)
  - cmdNetworkDumpXML, [450](#)
  - cmdNetworkEdit, [450](#)
  - cmdNetworkInfo, [450](#)
  - cmdNetworkList, [451](#)
  - cmdNetworkName, [451](#)
  - cmdNetworkStart, [451](#)
  - cmdNetworkUndefine, [451](#)
  - cmdNetworkUpdate, [451](#)
  - cmdNetworkUuid, [451](#)
  - EDIT\_DEFINE, [450](#)
  - EDIT\_FREE, [450](#)
  - EDIT\_GET\_XML, [450](#)
  - EDIT\_NOT\_CHANGED, [450](#)
  - info\_network\_autostart, [451](#)
  - info\_network\_create, [451](#)
  - info\_network\_define, [451](#)
  - info\_network\_destroy, [452](#)
  - info\_network\_dumpxml, [452](#)
  - info\_network\_edit, [452](#)
  - info\_network\_info, [452](#)
  - info\_network\_list, [452](#)
  - info\_network\_name, [452](#)
  - info\_network\_start, [453](#)
  - info\_network\_undefine, [453](#)
  - info\_network\_update, [453](#)
  - info\_network\_uuid, [453](#)
  - networkCmds, [453](#)
  - opts\_network\_autostart, [454](#)
  - opts\_network\_create, [454](#)
  - opts\_network\_define, [454](#)
  - opts\_network\_destroy, [454](#)
  - opts\_network\_dumpxml, [454](#)
  - opts\_network\_edit, [455](#)
  - opts\_network\_info, [455](#)
  - opts\_network\_list, [455](#)
  - opts\_network\_name, [455](#)
  - opts\_network\_start, [455](#)
  - opts\_network\_undefine, [455](#)
  - opts\_network\_update, [456](#)
  - opts\_network\_uuid, [456](#)
  - VIR\_ENUM\_IMPL, [451](#)
  - vshCommandOptNetworkBy, [451](#)
  - vshNetworkGetXMLDesc, [451](#)
  - vshNetworkListCollect, [451](#)
  - vshNetworkListFree, [451](#)
  - vshNetworkListPtr, [450](#)
  - vshNetworkSorter, [451](#)
- virtPortProfile
  - \_virDomainActualNetDef, [11](#)
  - \_virDomainNetDef, [46](#)
  - \_virNetworkDef, [69](#)
  - \_virPortGroupDef, [80](#)
- virtType
  - \_virDomainDef, [22](#)
- virtio
  - \_virDomainNetDef, [46](#)
- vlan
  - \_virDomainActualNetDef, [11](#)
  - \_virDomainNetDef, [46](#)
  - \_virNetworkDef, [69](#)
  - \_virPortGroupDef, [80](#)
- vmuuid

- virnetdevopenvswitch.h, 445
- vnc
  - \_virDomainGraphicsDef, 36
- volCreateXML
  - \_virStorageDriver, 85
- volCreateXMLFrom
  - \_virStorageDriver, 85
- volDelete
  - \_virStorageDriver, 85
- volDownload
  - \_virStorageDriver, 85
- volGetInfo
  - \_virStorageDriver, 85
- volGetPath
  - \_virStorageDriver, 85
- volGetXMLDesc
  - \_virStorageDriver, 85
- volLookupByKey
  - \_virStorageDriver, 86
- volLookupByName
  - \_virStorageDriver, 86
- volLookupByPath
  - \_virStorageDriver, 86
- volResize
  - \_virStorageDriver, 86
- volUpload
  - \_virStorageDriver, 86
- volWipe
  - \_virStorageDriver, 86
- volWipePattern
  - \_virStorageDriver, 86
- vram
  - \_virDomainVideoDef, 55
- vshCommandOptNetworkBy
  - virsh-network.c, 451
- vshNetworkGetXMLDesc
  - virsh-network.c, 451
- vshNetworkList, 91
  - nets, 91
  - nnets, 91
- vshNetworkListCollect
  - virsh-network.c, 451
- vshNetworkListFree
  - virsh-network.c, 451
- vshNetworkListPtr
  - virsh-network.c, 450
- vshNetworkSorter
  - virsh-network.c, 451
- WANT\_VALUE
  - qemu\_command.c, 433
- watchdog
  - \_virDomainDef, 22
  - \_virDomainDeviceDef, 24
- weight
  - \_virBlkioDeviceWeight, 7
  - \_virDomainDef, 22
  - \_virNetworkDNSSrvRecordsDef, 71
- wr\_bytes
  - \_virDomainBlockStats, 13
- wr\_req
  - \_virDomainBlockStats, 13
- write\_bytes\_sec
  - \_virDomainBlockIoTuneInfo, 12
- write\_iops\_sec
  - \_virDomainBlockIoTuneInfo, 12
- wrpolicy
  - \_virDomainFSDef, 34
- wwn
  - \_virDomainDiskDef, 30
- xauth
  - \_virDomainGraphicsDef, 36
- zlib
  - \_virDomainGraphicsDef, 36