

# libvserver Reference Manual

## 2.0\_rc1

Generated by Doxygen 1.5.2

Wed Jun 20 19:50:08 2007



# Contents

<b>1</b>	<b>libvserver Module Index</b>	<b>1</b>
1.1	libvserver Modules . . . . .	1
<b>2</b>	<b>libvserver Data Structure Index</b>	<b>3</b>
2.1	libvserver Data Structures . . . . .	3
<b>3</b>	<b>libvserver File Index</b>	<b>5</b>
3.1	libvserver File List . . . . .	5
<b>4</b>	<b>libvserver Module Documentation</b>	<b>7</b>
4.1	Context commands . . . . .	7
4.2	CPU scheduler commands . . . . .	27
4.3	Resource limit commands . . . . .	32
4.4	Disk limit commands . . . . .	37
4.5	Inode attribute commands . . . . .	41
4.6	Namespace commands . . . . .	44
4.7	Network context commands . . . . .	48
<b>5</b>	<b>libvserver Data Structure Documentation</b>	<b>59</b>
5.1	_dx_limit Struct Reference . . . . .	59
5.2	_ix_attr Struct Reference . . . . .	61
5.3	_nx_addr Struct Reference . . . . .	62
5.4	_nx_flags Struct Reference . . . . .	63
5.5	_nx_info Struct Reference . . . . .	64
5.6	_nx_sock_stat Struct Reference . . . . .	65
5.7	_vx_flags Struct Reference . . . . .	66
5.8	_vx_info Struct Reference . . . . .	67
5.9	_vx_limit Struct Reference . . . . .	68
5.10	_vx_limit_stat Struct Reference . . . . .	69
5.11	_vx_sched Struct Reference . . . . .	71

5.12	_vx_sched_info Struct Reference . . . . .	73
5.13	_vx_stat Struct Reference . . . . .	75
5.14	_vx_uname Struct Reference . . . . .	78
5.15	_vx_wait Struct Reference . . . . .	79
<b>6</b>	<b>libvserver File Documentation</b>	<b>81</b>
6.1	context.c File Reference . . . . .	81
6.2	dlimit.c File Reference . . . . .	83
6.3	inode.c File Reference . . . . .	84
6.4	limit.c File Reference . . . . .	85
6.5	namespace.c File Reference . . . . .	86
6.6	network.c File Reference . . . . .	87
6.7	sched.c File Reference . . . . .	88
6.8	switch.c File Reference . . . . .	89
6.9	syscall.c File Reference . . . . .	90
6.10	vserver.h File Reference . . . . .	91

# Chapter 1

## libvserver Module Index

### 1.1 libvserver Modules

Here is a list of all modules:

Context commands . . . . .	7
CPU scheduler commands . . . . .	27
Resource limit commands . . . . .	32
Disk limit commands . . . . .	37
Inode attribute commands . . . . .	41
Namespace commands . . . . .	44
Network context commands . . . . .	48



## Chapter 2

# libvserver Data Structure Index

### 2.1 libvserver Data Structures

Here are the data structures with brief descriptions:

<b>_dx_limit</b> (Disk limit values ) . . . . .	59
<b>_ix_attr</b> (Inode attributes ) . . . . .	61
<b>_nx_addr</b> (Network address information ) . . . . .	62
<b>_nx_flags</b> (Network context flags ) . . . . .	63
<b>_nx_info</b> (Network context information ) . . . . .	64
<b>_nx_sock_stat</b> (Accounting data ) . . . . .	65
<b>_vx_flags</b> (Context/migration flags ) . . . . .	66
<b>_vx_info</b> (Context information ) . . . . .	67
<b>_vx_limit</b> (Resource limits ) . . . . .	68
<b>_vx_limit_stat</b> (Resource limit accounting ) . . . . .	69
<b>_vx_sched</b> (Scheduler values ) . . . . .	71
<b>_vx_sched_info</b> (Scheduler information ) . . . . .	73
<b>_vx_stat</b> (Context statistics ) . . . . .	75
<b>_vx_uname</b> (Virtual system information data ) . . . . .	78
<b>_vx_wait</b> (Wait results ) . . . . .	79





## Chapter 3

# libvserver File Index

### 3.1 libvserver File List

Here is a list of all files with brief descriptions:

<b>context.c</b>	81
<b>dlimit.c</b>	83
<b>inode.c</b>	84
<b>limit.c</b>	85
<b>namespace.c</b>	86
<b>network.c</b>	87
<b>sched.c</b>	88
<b>switch.c</b>	89
<b>syscall.c</b>	90
<b>vserver.h</b> (Interface to the vserver syscalls )	91



## Chapter 4

# libvserver Module Documentation

### 4.1 Context commands

#### Data Structures

- `struct __vx_info`  
*Context information.*
- `struct __vx_stat`  
*Context statistics.*
- `struct __vx_flags`  
*Context/migration flags.*
- `struct __vx_uname`  
*Virtual system information data.*
- `struct __vx_wait`  
*Wait results.*

#### Defines

- `#define CAP__CHOWN 0`
- `#define CAP__DAC__OVERRIDE 1`
- `#define CAP__DAC__READ__SEARCH 2`
- `#define CAP__FOWNER 3`
- `#define CAP__FSETID 4`
- `#define CAP__KILL 5`
- `#define CAP__SETGID 6`
- `#define CAP__SETUID 7`
- `#define CAP__SETPCAP 8`
- `#define CAP__LINUX__IMMUTABLE 9`
- `#define CAP__NET__BIND__SERVICE 10`
- `#define CAP__NET__BROADCAST 11`

- #define CAP\_NET\_ADMIN 12
- #define CAP\_NET\_RAW 13
- #define CAP\_IPC\_LOCK 14
- #define CAP\_IPC\_OWNER 15
- #define CAP\_SYS\_MODULE 16
- #define CAP\_SYS\_RAWIO 17
- #define CAP\_SYS\_CHROOT 18
- #define CAP\_SYS\_PTRACE 19
- #define CAP\_SYS\_PACCT 20
- #define CAP\_SYS\_ADMIN 21
- #define CAP\_SYS\_BOOT 22
- #define CAP\_SYS\_NICE 23
- #define CAP\_SYS\_RESOURCE 24
- #define CAP\_SYS\_TIME 25
- #define CAP\_SYS\_TTY\_CONFIG 26
- #define CAP\_MKNOD 27
- #define CAP\_LEASE 28
- #define CAP\_AUDIT\_WRITE 29
- #define CAP\_AUDIT\_CONTROL 30
- #define CAP\_CONTEXT 31
- #define VXC\_SET\_UTSNAME 0x00000001
- #define VXC\_SET\_RLIMIT 0x00000002
- #define VXC\_RAW\_ICMP 0x00000100
- #define VXC\_SYSLOG 0x00001000
- #define VXC\_SECURE\_MOUNT 0x00010000
- #define VXC\_SECURE\_REMOUNT 0x00020000
- #define VXC\_BINARY\_MOUNT 0x00040000
- #define VXC\_QUOTA\_CTL 0x00100000
- #define VXC\_ADMIN\_MAPPER 0x00200000
- #define VXC\_ADMIN\_CLOOP 0x00400000
- #define VXF\_INFO\_SCHED 0x00000002
- #define VXF\_INFO\_NPROC 0x00000004
- #define VXF\_INFO\_PRIVATE 0x00000008
- #define VXF\_INFO\_INIT 0x00000010
- #define VXF\_INFO\_HIDE 0x00000020
- #define VXF\_INFO\_ULIMIT 0x00000040
- #define VXF\_INFO\_NSPACE 0x00000080
- #define VXF\_SCHED\_HARD 0x00000100
- #define VXF\_SCHED\_PRIO 0x00000200
- #define VXF\_SCHED\_PAUSE 0x00000400
- #define VXF\_VIRT\_MEM 0x00010000
- #define VXF\_VIRT\_UPTIME 0x00020000
- #define VXF\_VIRT\_CPU 0x00040000
- #define VXF\_VIRT\_LOAD 0x00080000
- #define VXF\_VIRT\_TIME 0x00100000
- #define VXF\_HIDE\_MOUNT 0x01000000
- #define VXF\_HIDE\_NETIF 0x02000000
- #define VXF\_HIDE\_VINFO 0x04000000
- #define VXF\_STATE\_SETUP (1ULL<<32)
- #define VXF\_STATE\_INIT (1ULL<<33)

- `#define VXF_STATE_ADMIN (1ULL<<34)`
- `#define VXF_SC_HELPER (1ULL<<36)`
- `#define VXF_REBOOT_KILL (1ULL<<37)`
- `#define VXF_PERSISTENT (1ULL<<38)`
- `#define VXF_FORK_RSS (1ULL<<48)`
- `#define VXF_PROLIFIC (1ULL<<49)`
- `#define VXF_IGNEG_NICE (1ULL<<52)`
- `#define VXM_SET_INIT 0x00000001`
- `#define VXM_SET_REAPER 0x00000002`
- `#define VHIN_CONTEXT 0`
- `#define VHIN_SYSNAME 1`
- `#define VHIN_NODENAME 2`
- `#define VHIN_RELEASE 3`
- `#define VHIN_VERSION 4`
- `#define VHIN_MACHINE 5`
- `#define VHIN_DOMAINNAME 6`

## Typedefs

- `typedef uint32_t xid_t`
- `typedef __vx_info vx_info_t`  
*Context information.*
- `typedef __vx_stat vx_stat_t`  
*Context statistics.*
- `typedef __vx_flags vx_flags_t`  
*Context/migration flags.*
- `typedef __vx_uname vx_uname_t`  
*Virtual system information data.*
- `typedef __vx_wait vx_wait_t`  
*Wait results.*

## Functions

- `int vx_create (xid_t xid, vx_flags_t *data)`  
*Create a new context.*
- `int vx_migrate (xid_t xid, vx_flags_t *data)`  
*Migrate to an existing context.*
- `int vx_task_xid (pid_t pid)`  
*Get the context ID of a process.*
- `int vx_info (xid_t xid, vx_info_t *data)`  
*Get context information.*

- `int vx_stat (xid_t xid, vx_stat_t *data)`  
*Get context statistics.*
- `int vx_bcaps_set (xid_t xid, vx_flags_t *data)`  
*Set system capabilities.*
- `int vx_bcaps_get (xid_t xid, vx_flags_t *data)`  
*Get system capabilities.*
- `int vx_ccaps_set (xid_t xid, vx_flags_t *data)`  
*Set context capabilities.*
- `int vx_ccaps_get (xid_t xid, vx_flags_t *data)`  
*Get context capabilities.*
- `int vx_flags_set (xid_t xid, vx_flags_t *data)`  
*Set context flags.*
- `int vx_flags_get (xid_t xid, vx_flags_t *data)`  
*Get context flags.*
- `int vx_uname_set (xid_t xid, vx_uname_t *data)`  
*Set virtual system information.*
- `int vx_uname_get (xid_t xid, vx_uname_t *data)`  
*Get virtual system information.*
- `int vx_kill (xid_t xid, pid_t pid, int sig)`  
*Kill one or more processes.*
- `int vx_wait (xid_t xid, vx_wait_t *data)`  
*Wait for context death.*

### 4.1.1 Define Documentation

#### 4.1.1.1 `#define CAP_CHOWN 0`

Definition at line 85 of file vserver.h.

#### 4.1.1.2 `#define CAP_DAC_OVERRIDE 1`

Definition at line 86 of file vserver.h.

#### 4.1.1.3 `#define CAP_DAC_READ_SEARCH 2`

Definition at line 87 of file vserver.h.

**4.1.1.4 #define CAP\_FOWNER 3**

Definition at line 88 of file vserver.h.

**4.1.1.5 #define CAP\_FSETID 4**

Definition at line 89 of file vserver.h.

**4.1.1.6 #define CAP\_KILL 5**

Definition at line 90 of file vserver.h.

**4.1.1.7 #define CAP\_SETGID 6**

Definition at line 91 of file vserver.h.

**4.1.1.8 #define CAP\_SETUID 7**

Definition at line 92 of file vserver.h.

**4.1.1.9 #define CAP\_SETPCAP 8**

Definition at line 93 of file vserver.h.

**4.1.1.10 #define CAP\_LINUX\_IMMUTABLE 9**

Definition at line 94 of file vserver.h.

**4.1.1.11 #define CAP\_NET\_BIND\_SERVICE 10**

Definition at line 95 of file vserver.h.

**4.1.1.12 #define CAP\_NET\_BROADCAST 11**

Definition at line 96 of file vserver.h.

**4.1.1.13 #define CAP\_NET\_ADMIN 12**

Definition at line 97 of file vserver.h.

**4.1.1.14 #define CAP\_NET\_RAW 13**

Definition at line 98 of file vserver.h.

**4.1.1.15 #define CAP\_IPC\_LOCK 14**

Definition at line 99 of file vserver.h.

**4.1.1.16 #define CAP\_IPC\_OWNER 15**

Definition at line 100 of file vserver.h.

**4.1.1.17 #define CAP\_SYS\_MODULE 16**

Definition at line 101 of file vserver.h.

**4.1.1.18 #define CAP\_SYS\_RAWIO 17**

Definition at line 102 of file vserver.h.

**4.1.1.19 #define CAP\_SYS\_CHROOT 18**

Definition at line 103 of file vserver.h.

**4.1.1.20 #define CAP\_SYS\_PTRACE 19**

Definition at line 104 of file vserver.h.

**4.1.1.21 #define CAP\_SYS\_PACCT 20**

Definition at line 105 of file vserver.h.

**4.1.1.22 #define CAP\_SYS\_ADMIN 21**

Definition at line 106 of file vserver.h.

**4.1.1.23 #define CAP\_SYS\_BOOT 22**

Definition at line 107 of file vserver.h.

**4.1.1.24 #define CAP\_SYS\_NICE 23**

Definition at line 108 of file vserver.h.

**4.1.1.25 #define CAP\_SYS\_RESOURCE 24**

Definition at line 109 of file vserver.h.

**4.1.1.26 #define CAP\_SYS\_TIME 25**

Definition at line 110 of file vserver.h.

**4.1.1.27 #define CAP\_SYS\_TTY\_CONFIG 26**

Definition at line 111 of file vserver.h.



**4.1.1.28 #define CAP\_MKNOD 27**

Definition at line 112 of file vserver.h.

**4.1.1.29 #define CAP\_LEASE 28**

Definition at line 113 of file vserver.h.

**4.1.1.30 #define CAP\_AUDIT\_WRITE 29**

Definition at line 114 of file vserver.h.

**4.1.1.31 #define CAP\_AUDIT\_CONTROL 30**

Definition at line 115 of file vserver.h.

**4.1.1.32 #define CAP\_CONTEXT 31**

Definition at line 116 of file vserver.h.

**4.1.1.33 #define VXC\_SET\_UTSNAME 0x00000001**

Allow setdomainname(2) and sethostname(2)

Definition at line 120 of file vserver.h.

**4.1.1.34 #define VXC\_SET\_RLIMIT 0x00000002**

Allow setrlimit(2)

Definition at line 121 of file vserver.h.

**4.1.1.35 #define VXC\_RAW\_ICMP 0x00000100**

Allow raw ICMP sockets

Definition at line 122 of file vserver.h.

**4.1.1.36 #define VXC\_SYSLOG 0x00001000**

Allow syslog(2)

Definition at line 123 of file vserver.h.

**4.1.1.37 #define VXC\_SECURE\_MOUNT 0x00010000**

Allow secure mount(2)

Definition at line 124 of file vserver.h.

**4.1.1.38 #define VXC\_SECURE\_REMOUNT 0x00020000**

Allow secure remount

Definition at line 125 of file vserver.h.

**4.1.1.39 #define VXC\_BINARY\_MOUNT 0x00040000**

Allow binary/network mounts

Definition at line 126 of file vserver.h.

**4.1.1.40 #define VXC\_QUOTA\_CTL 0x00100000**

Allow quota ioctl

Definition at line 127 of file vserver.h.

**4.1.1.41 #define VXC\_ADMIN\_MAPPER 0x00200000**

Allow access to device mapper

Definition at line 128 of file vserver.h.

**4.1.1.42 #define VXC\_ADMIN\_CLOOP 0x00400000**

Allow access to loop devices

Definition at line 129 of file vserver.h.

**4.1.1.43 #define VXF\_INFO\_SCHED 0x00000002**

Account all processes as one (L)

Definition at line 131 of file vserver.h.

**4.1.1.44 #define VXF\_INFO\_NPROC 0x00000004**

Apply process limits to context (L)

Definition at line 132 of file vserver.h.

**4.1.1.45 #define VXF\_INFO\_PRIVATE 0x00000008**

Context cannot be entered

Definition at line 133 of file vserver.h.

**4.1.1.46 #define VXF\_INFO\_INIT 0x00000010**

Show a fake init process

Definition at line 134 of file vserver.h.

**4.1.1.47 #define VXF\_INFO\_HIDE 0x00000020**

Hide context information in task status

Definition at line 135 of file vserver.h.

**4.1.1.48 #define VXF\_INFO\_ULIMIT 0x00000040**

Apply ulimits to context (L)

Definition at line 136 of file vserver.h.

**4.1.1.49 #define VXF\_INFO\_NAMESPACE 0x00000080**

Use private namespace (L)

Definition at line 137 of file vserver.h.

**4.1.1.50 #define VXF\_SCHED\_HARD 0x00000100**

Enable hard scheduler

Definition at line 138 of file vserver.h.

**4.1.1.51 #define VXF\_SCHED\_PRIO 0x00000200**

Enable priority scheduler

Definition at line 139 of file vserver.h.

**4.1.1.52 #define VXF\_SCHED\_PAUSE 0x00000400**

Pause context (unschedule)

Definition at line 140 of file vserver.h.

**4.1.1.53 #define VXF\_VIRT\_MEM 0x00010000**

Virtualize memory information

Definition at line 141 of file vserver.h.

**4.1.1.54 #define VXF\_VIRT\_UPTIME 0x00020000**

Virtualize uptime information

Definition at line 142 of file vserver.h.

**4.1.1.55 #define VXF\_VIRT\_CPU 0x00040000**

Virtualize cpu usage information

Definition at line 143 of file vserver.h.

**4.1.1.56 #define VXF\_VIRT\_LOAD 0x00080000**

Virtualize load average information

Definition at line 144 of file vserver.h.

**4.1.1.57 #define VXF\_VIRT\_TIME 0x00100000**

Allow per guest time offsets

Definition at line 145 of file vserver.h.

**4.1.1.58 #define VXF\_HIDE\_MOUNT 0x01000000**

Hide entries in /proc/\$pid/mounts

Definition at line 146 of file vserver.h.

**4.1.1.59 #define VXF\_HIDE\_NETIF 0x02000000**

Hide foreign network interfaces

Definition at line 147 of file vserver.h.

**4.1.1.60 #define VXF\_HIDE\_VINFO 0x04000000**

Hide context information in task status

Definition at line 148 of file vserver.h.

**4.1.1.61 #define VXF\_STATE\_SETUP (1ULL<<32)**

Enable setup state

Definition at line 149 of file vserver.h.

**4.1.1.62 #define VXF\_STATE\_INIT (1ULL<<33)**

Enable init state

Definition at line 150 of file vserver.h.

**4.1.1.63 #define VXF\_STATE\_ADMIN (1ULL<<34)**

Enable admin state

Definition at line 151 of file vserver.h.

**4.1.1.64 #define VXF\_SC\_HELPER (1ULL<<36)**

Context state change helper

Definition at line 152 of file vserver.h.

**4.1.1.65 #define VXF\_REBOOT\_KILL (1ULL<<37)**

Kill all processes on reboot(2)

Definition at line 153 of file vserver.h.

**4.1.1.66 #define VXF\_PERSISTENT (1ULL<<38)**

Make context persistent

Definition at line 154 of file vserver.h.

**4.1.1.67 #define VXF\_FORK\_RSS (1ULL<<48)**

Block fork when over RSS

Definition at line 155 of file vserver.h.

**4.1.1.68 #define VXF\_PROLIFIC (1ULL<<49)**

Allow context to create new contexts

Definition at line 156 of file vserver.h.

**4.1.1.69 #define VXF\_IGNEG\_NICE (1ULL<<52)**

Ignore priority raise

Definition at line 157 of file vserver.h.

**4.1.1.70 #define VXM\_SET\_INIT 0x00000001**

Make current process the new init

Definition at line 159 of file vserver.h.

**4.1.1.71 #define VXM\_SET\_REAPER 0x00000002**

Make current process the new reaper

Definition at line 160 of file vserver.h.

**4.1.1.72 #define VHIN\_CONTEXT 0**

Definition at line 164 of file vserver.h.

**4.1.1.73 #define VHIN\_SYSNAME 1**

Definition at line 165 of file vserver.h.

**4.1.1.74 #define VHIN\_NODENAME 2**

Definition at line 166 of file vserver.h.

**4.1.1.75 #define VHIN\_RELEASE 3**

Definition at line 167 of file vserver.h.

**4.1.1.76 #define VHIN\_VERSION 4**

Definition at line 168 of file vserver.h.

**4.1.1.77 #define VHIN\_MACHINE 5**

Definition at line 169 of file vserver.h.

**4.1.1.78 #define VHIN\_DOMAINNAME 6**

Definition at line 170 of file vserver.h.

**4.1.2 Typedef Documentation****4.1.2.1 typedef uint32\_t xid\_t**

Context ID type

Definition at line 173 of file vserver.h.

**4.1.2.2 typedef struct \_vx\_info vx\_info\_t**

Context information.

**4.1.2.3 typedef struct \_vx\_stat vx\_stat\_t**

Context statistics.

**4.1.2.4 typedef struct \_vx\_flags vx\_flags\_t**

Context/migration flags.

**4.1.2.5 typedef struct \_vx\_uname vx\_uname\_t**

Virtual system information data.

**4.1.2.6 typedef struct \_vx\_wait vx\_wait\_t**

Wait results.

### 4.1.3 Function Documentation

#### 4.1.3.1 `int vx_create (xid_t xid, vx_flags_t * data)`

Create a new context.

**Parameters:**

*xid* Context ID

*data* Initial context flags

Definition at line 61 of file context.c.

References `_vx_flags::flags`, and `vserver()`.

```
62 {
63     struct vcmd_ctx_create kdata = {
64         .flagword = 0,
65     };
66
67     if (data)
68         kdata.flagword = data->flags;
69
70     return vserver(VCMD_ctx_create, xid, &kdata);
71 }
```

#### 4.1.3.2 `int vx_migrate (xid_t xid, vx_flags_t * data)`

Migrate to an existing context.

**Parameters:**

*xid* Context ID

*data* Migration flags

Definition at line 73 of file context.c.

References `_vx_flags::flags`, and `vserver()`.

```
74 {
75     struct vcmd_ctx_migrate kdata = {
76         .flagword = 0,
77     };
78
79     if (data)
80         kdata.flagword = data->flags;
81
82     return vserver(VCMD_ctx_migrate, xid, &kdata);
83 }
```

#### 4.1.3.3 `int vx_task_xid (pid_t pid)`

Get the context ID of a process.

**Parameters:**

*pid* Process ID

**Returns:**

Context ID

Definition at line 85 of file context.c.

References `vserver()`.

```
86 {
87     return vserver(VCMD_task_xid, pid, NULL);
88 }
```

**4.1.3.4 int vx\_info (xid\_t xid, vx\_info\_t \* data)**

Get context information.

**Parameters:**

*xid* Context ID

*data* Empty vx\_info\_t struct to be filled

Definition at line 90 of file context.c.

References `_vx_info::initpid`, `vserver()`, and `_vx_info::xid`.

```
91 {
92     struct vcmd_vx_info_v0 kdata;
93
94     int rc = vserver(VCMD_vx_info, xid, &kdata);
95
96     if (rc == -1)
97         return rc;
98
99     if (data) {
100         data->xid = kdata.xid;
101         data->initpid = kdata.initpid;
102     }
103
104     return rc;
105 }
```

**4.1.3.5 int vx\_stat (xid\_t xid, vx\_stat\_t \* data)**

Get context statistics.

**Parameters:**

*xid* Context ID

*data* Empty vx\_stat\_t struct to be filled

Definition at line 107 of file context.c.

References `_vx_stat::load`, `_vx_stat::nr_forks`, `_vx_stat::nr_onhold`, `_vx_stat::nr_running`, `_vx_stat::nr_threads`, `_vx_stat::nr_unintr`, `_vx_stat::offset`, `_vx_stat::tasks`, `_vx_stat::uptime`, `_vx_stat::usecnt`, and `vserver()`.



```

108 {
109     int rc;
110     struct vcmd_ctx_stat_v0 kdata1;
111     struct vcmd_virt_stat_v0 kdata2;
112
113     rc = vserver(VCMD_ctx_stat, xid, &kdata1);
114
115     if (rc == -1)
116         return rc;
117
118     rc = vserver(VCMD_virt_stat, xid, &kdata2);
119
120     if (rc == -1)
121         return rc;
122
123     if (data) {
124         data->usecnt    = kdata1.usecnt;
125         data->tasks     = kdata1.tasks;
126         data->nr_threads = kdata2.nr_threads;
127         data->nr_running = kdata2.nr_running;
128         data->nr_unintr  = kdata2.nr_uninterruptible;
129         data->nr_onhold  = kdata2.nr_onhold;
130         data->nr_forks   = kdata2.nr_forks;
131         data->load[0]    = kdata2.load[0];
132         data->load[1]    = kdata2.load[1];
133         data->load[2]    = kdata2.load[2];
134         data->offset     = kdata2.offset;
135         data->uptime     = kdata2.uptime;
136     }
137
138     return rc;
139 }

```

#### 4.1.3.6 int vx\_bcaps\_set (xid\_t xid, vx\_flags\_t \* data)

Set system capabilities.

##### Parameters:

*xid* Context ID

*data* System capabilities

Definition at line 141 of file context.c.

References `_vx_flags::flags`, `_vx_flags::mask`, and `vserver()`.

```

142 {
143     struct vcmd_bcaps kdata;
144
145     if (!data)
146         return errno = EINVAL, -1;
147
148     kdata.bcaps = data->flags;
149     kdata.bmask = data->mask;
150
151     return vserver(VCMD_set_bcaps, xid, &kdata);
152 }

```

#### 4.1.3.7 int vx\_bcaps\_get (xid\_t xid, vx\_flags\_t \* data)

Get system capabilities.

**Parameters:***xid* Context ID*data* Empty vx\_flags\_t struct to be filled

Definition at line 154 of file context.c.

References \_vx\_flags::flags, \_vx\_flags::mask, and vserver().

```

155 {
156     struct vcmd_bcaps kdata;
157
158     if (!data)
159         return errno = EINVAL, -1;
160
161     int rc = vserver(VCMD_get_bcaps, xid, &kdata);
162
163     if (rc == -1)
164         return rc;
165
166     data->flags = kdata.bcaps;
167     data->mask = kdata.bmask;
168
169     return rc;
170 }

```

**4.1.3.8 int vx\_ccaps\_set (xid\_t xid, vx\_flags\_t \* data)**

Set context capabilities.

**Parameters:***xid* Context ID*data* Context capabilities

Definition at line 172 of file context.c.

References \_vx\_flags::flags, \_vx\_flags::mask, and vserver().

```

173 {
174     struct vcmd_ctx_caps_v1 kdata;
175
176     if (!data)
177         return errno = EINVAL, -1;
178
179     kdata.ccaps = data->flags;
180     kdata.cmask = data->mask;
181
182     return vserver(VCMD_set_ccaps, xid, &kdata);
183 }

```

**4.1.3.9 int vx\_ccaps\_get (xid\_t xid, vx\_flags\_t \* data)**

Get context capabilities.

**Parameters:***xid* Context ID

**data** Empty vx\_flags\_t struct to be filled

Definition at line 185 of file context.c.

References \_vx\_flags::flags, \_vx\_flags::mask, and vserver().

```
186 {
187     struct vcmd_ctx_caps_v1 kdata;
188
189     if (!data)
190         return errno = EINVAL, -1;
191
192     int rc = vserver(VCMD_get_ccaps, xid, &kdata);
193
194     if (rc == -1)
195         return rc;
196
197     data->flags = kdata.ccaps;
198     data->mask  = kdata.cmask;
199
200     return rc;
201 }
```

#### 4.1.3.10 int vx\_flags\_set (xid\_t xid, vx\_flags\_t \* data)

Set context flags.

##### Parameters:

**xid** Context ID

**data** Context flags

Definition at line 203 of file context.c.

References \_vx\_flags::flags, \_vx\_flags::mask, and vserver().

```
204 {
205     struct vcmd_ctx_flags_v0 kdata;
206
207     if (!data)
208         return errno = EINVAL, -1;
209
210     kdata.flagword = data->flags;
211     kdata.mask     = data->mask;
212
213     return vserver(VCMD_set_cflags, xid, &kdata);
214 }
```

#### 4.1.3.11 int vx\_flags\_get (xid\_t xid, vx\_flags\_t \* data)

Get context flags.

##### Parameters:

**xid** Context ID

**data** Empty vx\_flags\_t struct to be filled

Definition at line 216 of file context.c.

References `_vx_flags::flags`, `_vx_flags::mask`, and `vserver()`.

```

217 {
218     struct vcmd_ctx_flags_v0 kdata;
219
220     if (!data)
221         return errno = EINVAL, -1;
222
223     int rc = vserver(VCMD_get_cflags, xid, &kdata);
224
225     if (rc == -1)
226         return rc;
227
228     data->flags = kdata.flagword;
229     data->mask  = kdata.mask;
230
231     return rc;
232 }
```

#### 4.1.3.12 `int vx__uname__set (xid__t xid, vx__uname__t * data)`

Set virtual system information.

##### Parameters:

*xid* Context ID

*data* Virtual system information data

Definition at line 234 of file context.c.

References `_vx__uname::id`, `_vx__uname::value`, and `vserver()`.

```

235 {
236     struct vcmd_vhi_name_v0 kdata;
237
238     if (!data)
239         return errno = EINVAL, -1;
240
241     if (str_len(data->value) >= sizeof(kdata.name))
242         return errno = EINVAL, -1;
243
244     kdata.field = data->id;
245
246     str_zero(kdata.name, sizeof(kdata.name));
247     str_cpyn(kdata.name, data->value, sizeof(kdata.name) - 1);
248
249     return vserver(VCMD_set_vhi_name, xid, &kdata);
250 }
```

#### 4.1.3.13 `int vx__uname__get (xid__t xid, vx__uname__t * data)`

Get virtual system information.

##### Parameters:

*xid* Context ID

**data** Empty vx\_uname\_t struct to be filled

Definition at line 252 of file context.c.

References vx\_uname::id, vx\_uname::value, and vserver().

```

253 {
254     int rc;
255     struct vcmd_vhi_name_v0 kdata;
256
257     if (!data)
258         return errno = EINVAL, -1;
259
260     kdata.field = data->id;
261
262     rc = vserver(VCMD_get_vhi_name, xid, &kdata);
263
264     if (rc == -1)
265         return rc;
266
267     str_zero(data->value, sizeof(kdata.name));
268     str_cpyn(data->value, kdata.name, sizeof(kdata.name) - 1);
269
270     return rc;
271 }
```

#### 4.1.3.14 int vx\_kill (xid\_t xid, pid\_t pid, int sig)

Kill one or more processes.

##### Parameters:

**xid** Context ID

**pid** Process ID

**sig** Signal number

Definition at line 273 of file context.c.

References vserver().

```

274 {
275     struct vcmd_ctx_kill_v0 kdata;
276
277     kdata.pid = pid;
278     kdata.sig = sig;
279
280     return vserver(VCMD_ctx_kill, xid, &kdata);
281 }
```

#### 4.1.3.15 int vx\_wait (xid\_t xid, vx\_wait\_t \* data)

Wait for context death.

##### Parameters:

**xid** Context ID

**data** Empty vx\_wait\_t struct to be filled

Definition at line 283 of file context.c.

References `_vx_wait::exit_code`, `_vx_wait::reboot_cmd`, and `vserver()`.

```
284 {  
285     struct vcmd_wait_exit_v0 kdata;  
286  
287     int rc;  
288  
289     rc = vserver(VCMD_wait_exit, xid, &kdata);  
290  
291     if (rc == -1)  
292         return -1;  
293  
294     if (data) {  
295         data->reboot_cmd = kdata.reboot_cmd;  
296         data->exit_code  = kdata.exit_code;  
297     }  
298  
299     return rc;  
300 }
```

## 4.2 CPU scheduler commands

### Data Structures

- struct **\_vx\_sched**  
*Scheduler values.*
- struct **\_vx\_sched\_info**  
*Scheduler information.*

### Defines

- #define **VXSM\_FILL\_RATE** 0x0001
- #define **VXSM\_INTERVAL** 0x0002
- #define **VXSM\_FILL\_RATE2** 0x0004
- #define **VXSM\_INTERVAL2** 0x0008
- #define **VXSM\_TOKENS** 0x0010
- #define **VXSM\_TOKENS\_MIN** 0x0020
- #define **VXSM\_TOKENS\_MAX** 0x0040
- #define **VXSM\_PRIO\_BIAS** 0x0100
- #define **VXSM\_IDLE\_TIME** 0x0200
- #define **VXSM\_FORCE** 0x0400
- #define **VXSM\_CPU\_ID** 0x1000
- #define **VXSM\_BUCKET\_ID** 0x2000
- #define **VXSM\_MSEC** 0x4000
- #define **VXSM\_V3\_MASK** 0x0173
- #define **VXSM\_SET\_MASK** 0x01FF

### Typedefs

- typedef **\_vx\_sched vx\_sched\_t**  
*Scheduler values.*
- typedef **\_vx\_sched\_info vx\_sched\_info\_t**  
*Scheduler information.*

### Functions

- int **vx\_sched\_set** (xid\_t xid, vx\_sched\_t \*data)  
*Set scheduler values.*
- int **vx\_sched\_get** (xid\_t xid, vx\_sched\_t \*data)  
*Get scheduler values.*
- int **vx\_sched\_info** (xid\_t xid, vx\_sched\_info\_t \*data)  
*Get scheduler information.*

## 4.2.1 Define Documentation

### 4.2.1.1 `#define VXSM_FILL_RATE 0x0001`

Fill Rate

Definition at line 353 of file vserver.h.

### 4.2.1.2 `#define VXSM_INTERVAL 0x0002`

Interval

Definition at line 354 of file vserver.h.

### 4.2.1.3 `#define VXSM_FILL_RATE2 0x0004`

IDLE Fill Rate

Definition at line 355 of file vserver.h.

### 4.2.1.4 `#define VXSM_INTERVAL2 0x0008`

IDLE Interval

Definition at line 356 of file vserver.h.

### 4.2.1.5 `#define VXSM_TOKENS 0x0010`

Amount of tokens

Definition at line 357 of file vserver.h.

### 4.2.1.6 `#define VXSM_TOKENS_MIN 0x0020`

Minimum amount of tokens

Definition at line 358 of file vserver.h.

### 4.2.1.7 `#define VXSM_TOKENS_MAX 0x0040`

Maximum amount of tokens

Definition at line 359 of file vserver.h.

### 4.2.1.8 `#define VXSM_PRIO_BIAS 0x0100`

Priority bias

Definition at line 360 of file vserver.h.

### 4.2.1.9 `#define VXSM_IDLE_TIME 0x0200`

Use IDLE time settings



Definition at line 362 of file vserver.h.

#### 4.2.1.10 `#define VXSM_FORCE 0x0400`

Force scheduler reload (SMP only)

Definition at line 363 of file vserver.h.

#### 4.2.1.11 `#define VXSM_CPU_ID 0x1000`

CPU ID (SMP only)

Definition at line 364 of file vserver.h.

#### 4.2.1.12 `#define VXSM_BUCKET_ID 0x2000`

Bucket ID

Definition at line 365 of file vserver.h.

#### 4.2.1.13 `#define VXSM_MSEC 0x4000`

??

Definition at line 366 of file vserver.h.

#### 4.2.1.14 `#define VXSM_V3_MASK 0x0173`

Mask all fields for scheduler v3

Definition at line 368 of file vserver.h.

#### 4.2.1.15 `#define VXSM_SET_MASK 0x01FF`

Mask all fields for set\_sched

Definition at line 369 of file vserver.h.

### 4.2.2 Typedef Documentation

#### 4.2.2.1 `typedef struct _vx_sched vx_sched_t`

Scheduler values.

#### 4.2.2.2 `typedef struct _vx_sched_info vx_sched_info_t`

Scheduler information.

## 4.2.3 Function Documentation

### 4.2.3.1 `int vx_sched_set (xid_t xid, vx_sched_t * data)`

Set scheduler values.

**Parameters:**

*xid* Context ID

*data* Scheduler values

Definition at line 53 of file sched.c.

References `_vx_sched::cpu_id`, `_vx_sched::fill_rate`, `_vx_sched::interval`, `_vx_sched::mask`, `_vx_sched::prio_bias`, `_vx_sched::tokens`, `_vx_sched::tokens_max`, `_vx_sched::tokens_min`, and `vserver()`.

```

54 {
55     struct vcmd_sched_v5 kdata;
56
57     if (!data)
58         return errno = EINVAL, -1;
59
60     kdata.mask          = data->mask;
61     kdata.cpu_id        = data->cpu_id;
62     kdata.fill_rate[0]  = data->fill_rate[0];
63     kdata.fill_rate[1]  = data->fill_rate[1];
64     kdata.interval[0]   = data->interval[0];
65     kdata.interval[1]   = data->interval[1];
66     kdata.tokens         = data->tokens;
67     kdata.tokens_min     = data->tokens_min;
68     kdata.tokens_max     = data->tokens_max;
69     kdata.prio_bias      = data->prio_bias;
70
71     return vserver(VCMD_set_sched, xid, &kdata);
72 }
```

### 4.2.3.2 `int vx_sched_get (xid_t xid, vx_sched_t * data)`

Get scheduler values.

**Parameters:**

*xid* Context ID

*data* Scheduler values

Definition at line 25 of file sched.c.

References `_vx_sched::bucket_id`, `_vx_sched::cpu_id`, `_vx_sched::fill_rate`, `_vx_sched::interval`, `_vx_sched::mask`, `_vx_sched::prio_bias`, `_vx_sched::tokens`, `_vx_sched::tokens_max`, `_vx_sched::tokens_min`, and `vserver()`.

```

26 {
27     struct vcmd_sched_v5 kdata;
28
29     if (!data)
30         return errno = EINVAL, -1;
31 }
```

```

32     kdata.cpu_id    = data->cpu_id;
33     kdata.bucket_id = data->bucket_id;
34     kdata.mask      = data->mask;
35
36     int rc = vsrver(VCMD_get_sched, xid, &kdata);
37
38     if (rc == -1)
39         return -1;
40
41     data->fill_rate[0] = kdata.fill_rate[0];
42     data->fill_rate[1] = kdata.fill_rate[1];
43     data->interval[0]  = kdata.interval[0];
44     data->interval[1]  = kdata.interval[1];
45     data->tokens       = kdata.tokens;
46     data->tokens_min   = kdata.tokens_min;
47     data->tokens_max   = kdata.tokens_max;
48     data->prio_bias    = kdata.prio_bias;
49
50     return rc;
51 }

```

#### 4.2.3.3 int vx\_sched\_info (xid\_t xid, vx\_sched\_info\_t \* data)

Get scheduler information.

##### Parameters:

*xid* Context ID

*data* Scheduler information

Definition at line 74 of file sched.c.

References `_vx_sched_info::bucket_id`, `_vx_sched_info::cpu_id`, `_vx_sched_info::hold_msec`, `_vx_sched_info::sys_msec`, `_vx_sched_info::token_usec`, `_vx_sched_info::user_msec`, `_vx_sched_info::vavavoom`, and `vsrver()`.

```

75 {
76     struct vcmd_sched_info kdata;
77
78     if (!data)
79         return errno = EINVAL, -1;
80
81     kdata.cpu_id = data->cpu_id;
82     kdata.bucket_id = data->bucket_id;
83
84     int rc = vsrver(VCMD_sched_info, xid, &kdata);
85
86     if (rc == -1)
87         return -1;
88
89     data->user_msec = kdata.user_msec;
90     data->sys_msec  = kdata.sys_msec;
91     data->hold_msec = kdata.hold_msec;
92     data->token_usec = kdata.token_usec;
93     data->vavavoom  = kdata.vavavoom;
94
95     return rc;
96 }

```

## 4.3 Resource limit commands

### Data Structures

- struct **\_vx\_limit**  
*Resource limits.*
- struct **\_vx\_limit\_stat**  
*Resource limit accounting.*

### Defines

- #define **CRLIM\_UNSET** (0ULL)
- #define **CRLIM\_INFINITY** (~0ULL)
- #define **CRLIM\_KEEP** (~1ULL)
- #define **VLIMIT\_NSOCK** 16
- #define **VLIMIT\_OPENFD** 17
- #define **VLIMIT\_ANON** 18
- #define **VLIMIT\_SHMEM** 19
- #define **VLIMIT\_SEMARY** 20
- #define **VLIMIT\_NSEMS** 21
- #define **VLIMIT\_DENTRY** 22
- #define **VLIMIT\_MAPPED** 23

### Typedefs

- typedef **\_vx\_limit vx\_limit\_t**  
*Resource limits.*
- typedef **\_vx\_limit\_stat vx\_limit\_stat\_t**  
*Resource limit accounting.*

### Functions

- int **vx\_limit\_mask\_get** (vx\_limit\_t \*data)  
*Get resource limits mask.*
- int **vx\_limit\_set** (xid\_t xid, vx\_limit\_t \*data)  
*Set resource limit.*
- int **vx\_limit\_get** (xid\_t xid, vx\_limit\_t \*data)  
*Get resource limits.*
- int **vx\_limit\_stat** (xid\_t xid, vx\_limit\_stat\_t \*data)  
*Get resource limit accounting data.*
- int **vx\_limit\_reset** (xid\_t xid)  
*Reset resource limit accounting data.*

### 4.3.1 Define Documentation

#### 4.3.1.1 `#define CRLIM_UNSET (0ULL)`

Unset resource limit

Definition at line 434 of file vserver.h.

#### 4.3.1.2 `#define CRLIM_INFINITY (~0ULL)`

Infinity (no limit)

Definition at line 435 of file vserver.h.

#### 4.3.1.3 `#define CRLIM_KEEP (~1ULL)`

Keep current value

Definition at line 436 of file vserver.h.

#### 4.3.1.4 `#define VLIMIT_NSOCK 16`

Number of open sockets

Definition at line 440 of file vserver.h.

#### 4.3.1.5 `#define VLIMIT_OPENFD 17`

Number of open file descriptors

Definition at line 441 of file vserver.h.

#### 4.3.1.6 `#define VLIMIT_ANON 18`

Amount of anonymous memory

Definition at line 442 of file vserver.h.

#### 4.3.1.7 `#define VLIMIT_SHMEM 19`

Amount of shared memory

Definition at line 443 of file vserver.h.

#### 4.3.1.8 `#define VLIMIT_SEMARY 20`

Size of semary

Definition at line 444 of file vserver.h.

#### 4.3.1.9 `#define VLIMIT_NSEMS 21`

Number of semaphores

Definition at line 445 of file vserver.h.

#### 4.3.1.10 `#define VLIMIT_DENTRY 22`

Size of the dentry cache

Definition at line 446 of file vserver.h.

#### 4.3.1.11 `#define VLIMIT_MAPPED 23`

??

Definition at line 447 of file vserver.h.

### 4.3.2 Typedef Documentation

#### 4.3.2.1 `typedef struct _vx_limit vx_limit_t`

Resource limits.

#### 4.3.2.2 `typedef struct _vx_limit_stat vx_limit_stat_t`

Resource limit accounting.

### 4.3.3 Function Documentation

#### 4.3.3.1 `int vx_limit_mask_get (vx_limit_t * data)`

Get resource limits mask.

**Parameters:**

*data* Empty vx\_limit\_t struct to be filled

Definition at line 26 of file limit.c.

References `_vx_limit::maximum`, `_vx_limit::minimum`, `_vx_limit::softlimit`, and `vserver()`.

```

27 {
28     int rc;
29     struct vcmd_ctx_rlimit_mask_v0 kdata;
30
31     if (!data)
32         return errno = EINVAL, -1;
33
34     rc = vserver(VCMD_get_rlimit_mask, 0, &kdata);
35
36     if (rc == -1)
37         return rc;
38
39     data->minimum    = kdata.minimum;
40     data->softlimit  = kdata.softlimit;
41     data->maximum    = kdata.maximum;
42
43     return rc;
44 }
```

#### 4.3.3.2 int vx\_limit\_set (xid\_t xid, vx\_limit\_t \* data)

Set resource limit.

**Parameters:**

*xid* Context ID

*data* Resource limits

Definition at line 46 of file limit.c.

References `_vx_limit::id`, `_vx_limit::maximum`, `_vx_limit::minimum`, `_vx_limit::softlimit`, and `vserver()`.

```
47 {
48     struct vcmd_ctx_rlimit_v0 kdata;
49
50     if (!data)
51         return errno = EINVAL, -1;
52
53     kdata.id      = data->id;
54     kdata.minimum = data->minimum;
55     kdata.softlimit = data->softlimit;
56     kdata.maximum = data->maximum;
57
58     return vserver(VCMD_set_rlimit, xid, &kdata);
59 }
```

#### 4.3.3.3 int vx\_limit\_get (xid\_t xid, vx\_limit\_t \* data)

Get resource limits.

**Parameters:**

*xid* Context ID

*data* Empty vx\_limit\_t struct to be filled

Definition at line 61 of file limit.c.

References `_vx_limit::id`, `_vx_limit::maximum`, `_vx_limit::minimum`, `_vx_limit::softlimit`, and `vserver()`.

```
62 {
63     int rc;
64     struct vcmd_ctx_rlimit_v0 kdata;
65
66     if (!data)
67         return errno = EINVAL, -1;
68
69     kdata.id = data->id;
70
71     rc = vserver(VCMD_get_rlimit, xid, &kdata);
72
73     if (rc == -1)
74         return rc;
75
76     data->minimum = kdata.minimum;
77     data->softlimit = kdata.softlimit;
78     data->maximum = kdata.maximum;
79
80     return rc;
81 }
```

#### 4.3.3.4 int vx\_limit\_stat (xid\_t xid, vx\_limit\_stat\_t \* data)

Get resource limit accounting data.

##### Parameters:

*xid* Context ID

*data* Empty vx\_limit\_stat\_t struct to be filled

Definition at line 83 of file limit.c.

References vx\_limit\_stat::hits, vx\_limit\_stat::id, vx\_limit\_stat::maximum, vx\_limit\_stat::minimum, vx\_limit\_stat::value, and vserver().

```

84 {
85     int rc;
86     struct vcmd_rlimit_stat_v0 kdata;
87
88     if (!data)
89         return errno = EINVAL, -1;
90
91     kdata.id = data->id;
92
93     rc = vserver(VCMD_rlimit_stat, xid, &kdata);
94
95     if (rc == -1)
96         return rc;
97
98     data->hits    = kdata.hits;
99     data->value   = kdata.value;
100    data->minimum = kdata.minimum;
101    data->maximum = kdata.maximum;
102
103    return rc;
104 }
```

#### 4.3.3.5 int vx\_limit\_reset (xid\_t xid)

Reset resource limit accounting data.

##### Parameters:

*xid* Context ID

Definition at line 106 of file limit.c.

References vserver().

```

107 {
108     return vserver(VCMD_reset_minmax, xid, NULL);
109 }
```



## 4.4 Disk limit commands

### Data Structures

- struct **\_\_dx\_limit**  
*Disk limit values.*

### Defines

- #define **CDLIM\_UNSET** ((uint32\_t)0UL)
- #define **CDLIM\_INFINITY** ((uint32\_t)~0UL)
- #define **CDLIM\_KEEP** ((uint32\_t)~1UL)

### Typedefs

- typedef **\_\_dx\_limit dx\_limit\_t**  
*Disk limit values.*

### Functions

- int **dx\_limit\_add** (xid\_t xid, dx\_limit\_t \*data)  
*Add disk limit entry.*
- int **dx\_limit\_remove** (xid\_t xid, dx\_limit\_t \*data)  
*Remove disk limit.*
- int **dx\_limit\_set** (xid\_t xid, dx\_limit\_t \*data)  
*Set disk limit values.*
- int **dx\_limit\_get** (xid\_t xid, dx\_limit\_t \*data)  
*Get disk limit values.*

#### 4.4.1 Define Documentation

##### 4.4.1.1 #define CDLIM\_UNSET ((uint32\_t)0UL)

Unset disk limit

Definition at line 517 of file vserver.h.

##### 4.4.1.2 #define CDLIM\_INFINITY ((uint32\_t)~0UL)

Infinity (no limit)

Definition at line 518 of file vserver.h.

#### 4.4.1.3 `#define CDLIM_KEEP ((uint32_t)~1UL)`

Keep current value

Definition at line 519 of file vserver.h.

### 4.4.2 Typedef Documentation

#### 4.4.2.1 `typedef struct _dx_limit dx_limit_t`

Disk limit values.

### 4.4.3 Function Documentation

#### 4.4.3.1 `int dx_limit_add (xid_t xid, dx_limit_t * data)`

Add disk limit entry.

##### Parameters:

*xid* Context ID

*data* Disk limit information

Definition at line 26 of file dlimit.c.

References `_dx_limit::filename`, `_dx_limit::flags`, and `vserver()`.

```

27 {
28     struct vcmd_ctx_dlimit_base_v0 kdata;
29
30     if (!data)
31         return errno = EINVAL, -1;
32
33     kdata.name = data->filename;
34     kdata.flags = data->flags;
35
36     return vserver(VCMD_add_dlimit, xid, &kdata);
37 }
```

#### 4.4.3.2 `int dx_limit_remove (xid_t xid, dx_limit_t * data)`

Remove disk limit.

##### Parameters:

*xid* Context ID

*data* Disk limit information

Definition at line 39 of file dlimit.c.

References `_dx_limit::filename`, and `vserver()`.

```

40 {
41     struct vcmd_ctx_dlimit_base_v0 kdata;
```

```

42
43     if (!data)
44         return errno = EINVAL, -1;
45
46     kdata.name = data->filename;
47
48     return vservice(VCMD_rem_dlimit, xid, &kdata);
49 }

```

#### 4.4.3.3 int dx\_limit\_set (xid\_t xid, dx\_limit\_t \* data)

Set disk limit values.

##### Parameters:

**xid** Context ID

**data** Disk limit values

Definition at line 51 of file dlimit.c.

References `_dx_limit::filename`, `_dx_limit::flags`, `_dx_limit::inodes_total`, `_dx_limit::inodes_used`, `_dx_limit::reserved`, `_dx_limit::space_total`, `_dx_limit::space_used`, and `vservice()`.

```

52 {
53     struct vcmd_ctx_dlimit_v0 kdata;
54
55     if (!data)
56         return errno = EINVAL, -1;
57
58     kdata.name      = data->filename;
59     kdata.space_used = data->space_used;
60     kdata.space_total = data->space_total;
61     kdata.inodes_used = data->inodes_used;
62     kdata.inodes_total = data->inodes_total;
63     kdata.reserved     = data->reserved;
64     kdata.flags        = data->flags;
65
66     return vservice(VCMD_set_dlimit, xid, &kdata);
67 }

```

#### 4.4.3.4 int dx\_limit\_get (xid\_t xid, dx\_limit\_t \* data)

Get disk limit values.

##### Parameters:

**xid** Context ID

**data** Empty `dx_limit_t` struct to be filled

Definition at line 69 of file dlimit.c.

References `_dx_limit::filename`, `_dx_limit::flags`, `_dx_limit::inodes_total`, `_dx_limit::inodes_used`, `_dx_limit::reserved`, `_dx_limit::space_total`, `_dx_limit::space_used`, and `vservice()`.

```

70 {
71     int rc;
72     struct vcmd_ctx_dlimit_v0 kdata;

```

```
73
74     if (!data)
75         return errno = EINVAL, -1;
76
77     kdata.name = data->filename;
78
79     rc = vserver(VCMD_get_dlimit, xid, &kdata);
80
81     if (rc == -1)
82         return rc;
83
84     data->space_used   = kdata.space_used;
85     data->space_total  = kdata.space_total;
86     data->inodes_used  = kdata.inodes_used;
87     data->inodes_total = kdata.inodes_total;
88     data->reserved     = kdata.reserved;
89     data->flags        = kdata.flags;
90
91     return rc;
92 }
```

## 4.5 Inode attribute commands

### Data Structures

- struct **\_ix\_attr**  
*Inode attributes.*

### Defines

- #define **IATTR\_TAG** 0x01000000
- #define **IATTR\_ADMIN** 0x00000001
- #define **IATTR\_WATCH** 0x00000002
- #define **IATTR\_HIDE** 0x00000004
- #define **IATTR\_FLAGS** 0x00000007
- #define **IATTR\_BARRIER** 0x00010000
- #define **IATTR\_IUNLINK** 0x00020000
- #define **IATTR\_IMMUTABLE** 0x00040000

### Typedefs

- typedef **\_ix\_attr** **ix\_attr\_t**  
*Inode attributes.*

### Functions

- int **ix\_attr\_set** (**ix\_attr\_t** \*data)  
*Set inode attributes.*
- int **ix\_attr\_get** (**ix\_attr\_t** \*data)  
*Get inode attributes.*

#### 4.5.1 Define Documentation

##### 4.5.1.1 #define IATTR\_TAG 0x01000000

File is xid tagged

Definition at line 575 of file vserver.h.

##### 4.5.1.2 #define IATTR\_ADMIN 0x00000001

Accessible in xid=0 (only for /proc)

Definition at line 576 of file vserver.h.

#### 4.5.1.3 `#define IATTR_WATCH 0x00000002`

Accessible in xid=1 (only for /proc)

Definition at line 577 of file vserver.h.

#### 4.5.1.4 `#define IATTR_HIDE 0x00000004`

Not Accessible in xid!=(0|1) (only for /proc)

Definition at line 578 of file vserver.h.

#### 4.5.1.5 `#define IATTR_FLAGS 0x00000007`

Flag mask for /proc flags

Definition at line 579 of file vserver.h.

#### 4.5.1.6 `#define IATTR_BARRIER 0x00010000`

Directory barrier

Definition at line 580 of file vserver.h.

#### 4.5.1.7 `#define IATTR_IUNLINK 0x00020000`

Unlink file

Definition at line 581 of file vserver.h.

#### 4.5.1.8 `#define IATTR_IMMUTABLE 0x00040000`

File is immutable

Definition at line 582 of file vserver.h.

### 4.5.2 Typedef Documentation

#### 4.5.2.1 `typedef struct _ix_attr ix_attr_t`

Inode attributes.

### 4.5.3 Function Documentation

#### 4.5.3.1 `int ix_attr_set (ix_attr_t * data)`

Set inode attributes.

##### Parameters:

*data* Inode attributes

Definition at line 26 of file inode.c.

References `_ix_attr::filename`, `_ix_attr::flags`, `_ix_attr::mask`, `vserver()`, and `_ix_attr::xid`.

```
27 {
28     struct vcmd_ctx_iattr_v1 kdata;
29
30     if (!data)
31         return errno = EINVAL, -1;
32
33     kdata.name = data->filename;
34     kdata.xid = data->xid;
35     kdata.flags = data->flags;
36     kdata.mask = data->mask;
37
38     return vserver(VCMD_set_iattr, 0, &kdata);
39 }
```

#### 4.5.3.2 `int ix_attr_get (ix_attr_t * data)`

Get inode attributes.

##### Parameters:

*data* Empty `ix_attr_t` struct to be filled

Definition at line 41 of file inode.c.

References `_ix_attr::filename`, `_ix_attr::flags`, `_ix_attr::mask`, `vserver()`, and `_ix_attr::xid`.

```
42 {
43     int rc;
44     struct vcmd_ctx_iattr_v1 kdata;
45
46     if (!data)
47         return errno = EINVAL, -1;
48
49     kdata.name = data->filename;
50
51     rc = vserver(VCMD_get_iattr, 0, &kdata);
52
53     if (rc == -1)
54         return rc;
55
56     data->xid = kdata.xid;
57     data->flags = kdata.flags;
58     data->mask = kdata.mask;
59
60     return rc;
61 }
```

## 4.6 Namespace commands

### Defines

- `#define CLONE_VM 0x00000100`
- `#define CLONE_FS 0x00000200`
- `#define CLONE_FILES 0x00000400`
- `#define CLONE_SIGHAND 0x00000800`
- `#define CLONE_PTRACE 0x00002000`
- `#define CLONE_VFORK 0x00004000`
- `#define CLONE_PARENT 0x00008000`
- `#define CLONE_THREAD 0x00010000`
- `#define CLONE_NEWNS 0x00020000`
- `#define CLONE_SYSVSEM 0x00040000`
- `#define CLONE_SETTLS 0x00080000`
- `#define CLONE_PARENT_SETTID 0x00100000`
- `#define CLONE_CHILD_CLEARPID 0x00200000`
- `#define CLONE_DETACHED 0x00400000`
- `#define CLONE_UNTRACED 0x00800000`
- `#define CLONE_CHILD_SETTID 0x01000000`
- `#define CLONE_STOPPED 0x02000000`
- `#define CLONE_NEWUTS 0x04000000`
- `#define CLONE_NEWIPC 0x08000000`
- `#define CLONE_KTHREAD 0x10000000`

### Functions

- `int ns_clone (int flags, void *child_stack)`  
*Clone the current namespace (FS/IPC/UTS).*
- `int ns_enter (xid_t xid, uint64_t mask)`  
*Enter namespace.*
- `int ns_set (xid_t xid, uint64_t mask)`  
*Set namespace.*

#### 4.6.1 Define Documentation

##### 4.6.1.1 `#define CLONE_VM 0x00000100`

Definition at line 617 of file vserver.h.

##### 4.6.1.2 `#define CLONE_FS 0x00000200`

Definition at line 618 of file vserver.h.



**4.6.1.3 #define CLONE\_FILES 0x00000400**

Definition at line 619 of file vserver.h.

**4.6.1.4 #define CLONE\_SIGHAND 0x00000800**

Definition at line 620 of file vserver.h.

**4.6.1.5 #define CLONE\_PTRACE 0x00002000**

Definition at line 621 of file vserver.h.

**4.6.1.6 #define CLONE\_VFORK 0x00004000**

Definition at line 622 of file vserver.h.

**4.6.1.7 #define CLONE\_PARENT 0x00008000**

Definition at line 623 of file vserver.h.

**4.6.1.8 #define CLONE\_THREAD 0x00010000**

Definition at line 624 of file vserver.h.

**4.6.1.9 #define CLONE\_NEWNS 0x00020000**

Definition at line 625 of file vserver.h.

Referenced by ns\_clone().

**4.6.1.10 #define CLONE\_SYSVSEM 0x00040000**

Definition at line 626 of file vserver.h.

**4.6.1.11 #define CLONE\_SETTLS 0x00080000**

Definition at line 627 of file vserver.h.

**4.6.1.12 #define CLONE\_PARENT\_SETTID 0x00100000**

Definition at line 628 of file vserver.h.

**4.6.1.13 #define CLONE\_CHILD\_CLEARTID 0x00200000**

Definition at line 629 of file vserver.h.

**4.6.1.14 #define CLONE\_\_DETACHED 0x00400000**

Definition at line 630 of file vserver.h.

**4.6.1.15 #define CLONE\_\_UNTRACED 0x00800000**

Definition at line 631 of file vserver.h.

**4.6.1.16 #define CLONE\_\_CHILD\_\_SETTID 0x01000000**

Definition at line 632 of file vserver.h.

**4.6.1.17 #define CLONE\_\_STOPPED 0x02000000**

Definition at line 633 of file vserver.h.

**4.6.1.18 #define CLONE\_\_NEWUTS 0x04000000**

Definition at line 634 of file vserver.h.

Referenced by ns\_clone().

**4.6.1.19 #define CLONE\_\_NEWIPC 0x08000000**

Definition at line 635 of file vserver.h.

Referenced by ns\_clone().

**4.6.1.20 #define CLONE\_\_KTHREAD 0x10000000**

Definition at line 636 of file vserver.h.

**4.6.2 Function Documentation****4.6.2.1 int ns\_clone (int *flags*, void \* *child\_stack*)**

Clone the current namespace (FS/IPC/UTS).

**Parameters:**

***flags*** Clone flags

***child\_stack*** Child stack

Definition at line 25 of file namespace.c.

References clone(), CLONE\_\_NEWIPC, CLONE\_\_NEWNS, and CLONE\_\_NEWUTS.

```

26 {
27     return clone(flags|CLONE_NEWNS|CLONE_NEWUTS|CLONE_NEWIPC, child_stack);
28 }
```

#### 4.6.2.2 int ns\_enter (xid\_t *xid*, uint64\_t *mask*)

Enter namespace.

**Parameters:**

*xid* Context ID

Definition at line 30 of file namespace.c.

References vserver().

```
31 {  
32     struct vcmd_space_mask kdata = { .mask = mask };  
33     return vserver(VCMD_enter_space, xid, &kdata);  
34 }
```

#### 4.6.2.3 int ns\_set (xid\_t *xid*, uint64\_t *mask*)

Set namespace.

**Parameters:**

*xid* Context ID

Definition at line 36 of file namespace.c.

References vserver().

```
37 {  
38     struct vcmd_space_mask kdata = { .mask = mask };  
39     return vserver(VCMD_set_space, xid, &kdata);  
40 }
```

## 4.7 Network context commands

### Data Structures

- struct **\_nx\_info**  
*Network context information.*
- struct **\_nx\_addr**  
*Network address information.*
- struct **\_nx\_flags**  
*Network context flags.*
- struct **\_nx\_sock\_stat**  
*Accounting data.*

### Defines

- #define **NXF\_INFO\_PRIVATE** 0x00000008
- #define **NXF\_STATE\_SETUP** (1ULL<<32)
- #define **NXF\_STATE\_ADMIN** (1ULL<<34)
- #define **NXF\_SC\_HELPER** (1ULL<<36)
- #define **NXF\_PERSISTENT** (1ULL<<38)
- #define **NXA\_TYPE\_IPV4** 1
- #define **NXA\_TYPE\_IPV6** 2
- #define **NXA\_MOD\_BCAST** (1<<8)
- #define **NXA\_TYPE\_ANY** ((uint16\_t)-1)
- #define **VXA\_SOCK\_UNSPEC** 0
- #define **VXA\_SOCK\_UNIX** 1
- #define **VXA\_SOCK\_INET** 2
- #define **VXA\_SOCK\_INET6** 3
- #define **VXA\_SOCK\_PACKET** 4
- #define **VXA\_SOCK\_OTHER** 5
- #define **NXA\_SOCK\_UNSPEC** VXA\_SOCK\_UNSPEC
- #define **NXA\_SOCK\_UNIX** VXA\_SOCK\_UNIX
- #define **NXA\_SOCK\_INET** VXA\_SOCK\_INET
- #define **NXA\_SOCK\_INET6** VXA\_SOCK\_INET6
- #define **NXA\_SOCK\_PACKET** VXA\_SOCK\_PACKET
- #define **NXA\_SOCK\_OTHER** VXA\_SOCK\_OTHER

### Typedefs

- typedef uint32\_t **nid\_t**
- typedef **\_nx\_info** **nx\_info\_t**  
*Network context information.*
- typedef **\_nx\_addr** **nx\_addr\_t**  
*Network address information.*

- `typedef __nx_flags nx_flags_t`  
*Network context flags.*
- `typedef __nx_sock_stat nx_sock_stat_t`  
*Accounting data.*

## Functions

- `int nx_create (nid_t nid, nx_flags_t *data)`  
*Create network context.*
- `int nx_migrate (nid_t nid)`  
*Migrate to an existing network context.*
- `int nx_task_nid (pid_t pid)`  
*Get the network context ID of a process.*
- `int nx_info (nid_t nid, nx_info_t *data)`  
*Get network context information.*
- `int nx_addr_add (nid_t nid, nx_addr_t *data)`  
*Add network context addresses.*
- `int nx_addr_remove (nid_t nid, nx_addr_t *data)`  
*Remove network context addresses.*
- `int nx_flags_set (nid_t nid, nx_flags_t *data)`  
*Set network context flags.*
- `int nx_flags_get (nid_t nid, nx_flags_t *data)`  
*Get network context flags.*
- `int nx_caps_set (nid_t nid, nx_flags_t *data)`  
*Set network context capabilities.*
- `int nx_caps_get (nid_t nid, nx_flags_t *data)`  
*Get network context capabilities.*
- `int nx_sock_stat (nid_t nid, nx_sock_stat_t *data)`  
*Get network socket accounting data.*

### 4.7.1 Define Documentation

#### 4.7.1.1 `#define NXF_INFO_PRIVATE 0x00000008`

Network context cannot be entered

Definition at line 669 of file vserver.h.

**4.7.1.2   #define NXF\_STATE\_SETUP (1ULL<<32)**

Network context is in setup state

Definition at line 670 of file vserver.h.

**4.7.1.3   #define NXF\_STATE\_ADMIN (1ULL<<34)**

Context is in admin state

Definition at line 671 of file vserver.h.

**4.7.1.4   #define NXF\_SC\_HELPER (1ULL<<36)**

Network state change helper

Definition at line 672 of file vserver.h.

**4.7.1.5   #define NXF\_PERSISTENT (1ULL<<38)**

Make network context persistent

Definition at line 673 of file vserver.h.

**4.7.1.6   #define NXA\_TYPE\_IPV4 1**

Address is IPv4

Definition at line 675 of file vserver.h.

**4.7.1.7   #define NXA\_TYPE\_IPV6 2**

Address is IPv6

Definition at line 676 of file vserver.h.

**4.7.1.8   #define NXA\_MOD\_BCAST (1<<8)**

Address is Broadcast

Definition at line 677 of file vserver.h.

**4.7.1.9   #define NXA\_TYPE\_ANY ((uint16\_t)-1)**

Matches any address

Definition at line 678 of file vserver.h.

**4.7.1.10   #define VXA\_SOCK\_UNSPEC 0**

Definition at line 682 of file vserver.h.

**4.7.1.11 #define VXA\_\_SOCK\_\_UNIX 1**

Definition at line 683 of file vserver.h.

**4.7.1.12 #define VXA\_\_SOCK\_\_INET 2**

Definition at line 684 of file vserver.h.

**4.7.1.13 #define VXA\_\_SOCK\_\_INET6 3**

Definition at line 685 of file vserver.h.

**4.7.1.14 #define VXA\_\_SOCK\_\_PACKET 4**

Definition at line 686 of file vserver.h.

**4.7.1.15 #define VXA\_\_SOCK\_\_OTHER 5**

Definition at line 687 of file vserver.h.

**4.7.1.16 #define NXA\_\_SOCK\_\_UNSPEC VXA\_\_SOCK\_\_UNSPEC**

Definition at line 690 of file vserver.h.

**4.7.1.17 #define NXA\_\_SOCK\_\_UNIX VXA\_\_SOCK\_\_UNIX**

Definition at line 691 of file vserver.h.

**4.7.1.18 #define NXA\_\_SOCK\_\_INET VXA\_\_SOCK\_\_INET**

Definition at line 692 of file vserver.h.

**4.7.1.19 #define NXA\_\_SOCK\_\_INET6 VXA\_\_SOCK\_\_INET6**

Definition at line 693 of file vserver.h.

**4.7.1.20 #define NXA\_\_SOCK\_\_PACKET VXA\_\_SOCK\_\_PACKET**

Definition at line 694 of file vserver.h.

**4.7.1.21 #define NXA\_\_SOCK\_\_OTHER VXA\_\_SOCK\_\_OTHER**

Definition at line 695 of file vserver.h.

## 4.7.2 Typedef Documentation

### 4.7.2.1 typedef uint32\_t nid\_t

Network context ID type

Definition at line 697 of file vserver.h.

### 4.7.2.2 typedef struct \_nx\_info nx\_info\_t

Network context information.

### 4.7.2.3 typedef struct \_nx\_addr nx\_addr\_t

Network address information.

### 4.7.2.4 typedef struct \_nx\_flags nx\_flags\_t

Network context flags.

### 4.7.2.5 typedef struct \_nx\_sock\_stat nx\_sock\_stat\_t

Accounting data.

## 4.7.3 Function Documentation

### 4.7.3.1 int nx\_create (nid\_t *nid*, nx\_flags\_t \* *data*)

Create network context.

#### Parameters:

***nid*** Network context ID

***data*** Initial network context flags

Definition at line 60 of file network.c.

References `_nx_flags::flags`, and `vserver()`.

```
61 {  
62     struct vcmd_net_create kdata = {  
63         .flagword = 0,  
64     };  
65  
66     if (data)  
67         kdata.flagword = data->flags;  
68  
69     return vserver(VCMD_net_create, nid, &kdata);  
70 }
```



#### 4.7.3.2 int nx\_migrate (nid\_t *nid*)

Migrate to an existing network context.

**Parameters:**

*nid* Network context ID

Definition at line 72 of file network.c.

References vserver().

```
73 {  
74     return vserver(VCMD_net_migrate, nid, NULL);  
75 }
```

#### 4.7.3.3 int nx\_task\_nid (pid\_t *pid*)

Get the network context ID of a process.

**Parameters:**

*pid* Process ID

**Returns:**

Network context ID

Definition at line 77 of file network.c.

References vserver().

```
78 {  
79     return vserver(VCMD_task_nid, pid, NULL);  
80 }
```

#### 4.7.3.4 int nx\_info (nid\_t *nid*, nx\_info\_t \* *data*)

Get network context information.

**Parameters:**

*nid* Network context ID

*data* Empty nx\_info\_t struct to be filled

Definition at line 82 of file network.c.

References \_nx\_info::nid, and vserver().

```
83 {  
84     struct vcmd_nx_info_v0 kdata;  
85  
86     int rc = vserver(VCMD_nx_info, nid, &kdata);  
87  
88     if (rc == -1)
```

```

89         return rc;
90
91     if (data)
92         data->nid = kdata.nid;
93
94     return rc;
95 }

```

#### 4.7.3.5 int nx\_addr\_add (nid\_t *nid*, nx\_addr\_t \* *data*)

Add network context addresses.

##### Parameters:

***nid*** Network context ID

***data*** Network address information

Definition at line 97 of file network.c.

References `_nx_addr::count`, `_nx_addr::ip`, `_nx_addr::mask`, `_nx_addr::type`, and `vserver()`.

```

98 {
99     struct vcmd_net_addr_v0 kdata;
100
101     if (!data)
102         return errno = EINVAL, -1;
103
104     kdata.type = data->type;
105     kdata.count = data->count;
106
107     str_cpyn(kdata.ip, data->ip, sizeof(kdata.ip));
108     str_cpyn(kdata.mask, data->mask, sizeof(kdata.mask));
109
110     return vserver(VCMD_net_add, nid, &kdata);
111 }

```

#### 4.7.3.6 int nx\_addr\_remove (nid\_t *nid*, nx\_addr\_t \* *data*)

Remove network context addresses.

##### Parameters:

***nid*** Network context ID

***data*** Network address information

Definition at line 113 of file network.c.

References `_nx_addr::count`, `_nx_addr::ip`, `_nx_addr::mask`, `_nx_addr::type`, and `vserver()`.

```

114 {
115     struct vcmd_net_addr_v0 kdata;
116
117     if (!data)
118         return errno = EINVAL, -1;
119
120     kdata.type = data->type;
121     kdata.count = data->count;

```

```
122
123     str_cpy(kdata.ip, data->ip, sizeof(kdata.ip));
124     str_cpy(kdata.mask, data->mask, sizeof(kdata.mask));
125
126     return vserver(VCMD_net_remove, nid, &kdata);
127 }
```

#### 4.7.3.7 int nx\_flags\_set (nid\_t *nid*, nx\_flags\_t \* *data*)

Set network context flags.

##### Parameters:

*nid* Network context ID

*data* Network context flags

Definition at line 129 of file network.c.

References `_nx_flags::flags`, `_nx_flags::mask`, and `vserver()`.

```
130 {
131     struct vcmd_net_flags_v0 kdata;
132
133     if (!data)
134         return errno = EINVAL, -1;
135
136     kdata.flagword = data->flags;
137     kdata.mask      = data->mask;
138
139     return vserver(VCMD_set_nflags, nid, &kdata);
140 }
```

#### 4.7.3.8 int nx\_flags\_get (nid\_t *nid*, nx\_flags\_t \* *data*)

Get network context flags.

##### Parameters:

*nid* Network context ID

*data* Empty `nx_flags_t` struct to be filled

Definition at line 142 of file network.c.

References `_nx_flags::flags`, `_nx_flags::mask`, and `vserver()`.

```
143 {
144     struct vcmd_net_flags_v0 kdata;
145
146     if (!data)
147         return errno = EINVAL, -1;
148
149     int rc = vserver(VCMD_get_nflags, nid, &kdata);
150
151     if (rc == -1)
152         return rc;
153
154     data->flags = kdata.flagword;
```

```
155         data->mask = kdata.mask;
156
157         return rc;
158 }
```

#### 4.7.3.9 int nx\_caps\_set (nid\_t *nid*, nx\_flags\_t \* *data*)

Set network context capabilities.

##### Parameters:

***nid*** Network context ID

***data*** Network context capabilities

Definition at line 160 of file network.c.

References `_nx_flags::flags`, `_nx_flags::mask`, and `vserver()`.

```
161 {
162     struct vcmd_net_caps_v0 kdata;
163
164     if (!data)
165         return errno = EINVAL, -1;
166
167     kdata.ncaps = data->flags;
168     kdata.cmask = data->mask;
169
170     return vserver(VCMD_set_ncaps, nid, &kdata);
171 }
```

#### 4.7.3.10 int nx\_caps\_get (nid\_t *nid*, nx\_flags\_t \* *data*)

Get network context capabilities.

##### Parameters:

***nid*** Network context ID

***data*** Empty `nx_flags_t` struct to be filled

Definition at line 173 of file network.c.

References `_nx_flags::flags`, `_nx_flags::mask`, and `vserver()`.

```
174 {
175     struct vcmd_net_caps_v0 kdata;
176
177     if (!data)
178         return errno = EINVAL, -1;
179
180     int rc = vserver(VCMD_get_ncaps, nid, &kdata);
181
182     if (rc == -1)
183         return rc;
184
185     data->flags = kdata.ncaps;
186     data->mask = kdata.cmask;
187
188     return rc;
189 }
```

#### 4.7.3.11 `int nx_sock_stat (nid_t nid, nx_sock_stat_t * data)`

Get network socket accounting data.

##### Parameters:

***nid*** Network context ID

***data*** Empty `nx_sock_stat_t` struct to be filled

Definition at line 191 of file `network.c`.

References `_nx_sock_stat::count`, `_nx_sock_stat::id`, `_nx_sock_stat::total`, and `vserver()`.

```
192 {
193     struct vcmd_sock_stat_v0 kdata;
194
195     if (!data)
196         return errno = EINVAL, -1;
197
198     kdata.field = data->id;
199
200     int rc = vserver(VCMD_sock_stat, nid, &kdata);
201
202     if (rc == -1)
203         return rc;
204
205     data->count[0] = kdata.count[0];
206     data->count[1] = kdata.count[1];
207     data->count[2] = kdata.count[2];
208
209     data->total[0] = kdata.total[0];
210     data->total[1] = kdata.total[1];
211     data->total[2] = kdata.total[2];
212
213     return rc;
214 }
```



## Chapter 5

# libvserver Data Structure Documentation

### 5.1 `__dx_limit` Struct Reference

```
#include <vserver.h>
```

#### 5.1.1 Detailed Description

Disk limit values.

Definition at line 525 of file `vserver.h`.

#### Data Fields

- `const char * filename`
- `uint32_t space_used`
- `uint32_t space_total`
- `uint32_t inodes_used`
- `uint32_t inodes_total`
- `uint32_t reserved`
- `uint32_t flags`

#### 5.1.2 Field Documentation

##### 5.1.2.1 `const char* __dx_limit::filename`

Mount point

Definition at line 526 of file `vserver.h`.

Referenced by `dx_limit_add()`, `dx_limit_get()`, `dx_limit_remove()`, and `dx_limit_set()`.

##### 5.1.2.2 `uint32_t __dx_limit::space_used`

Currently used space

Definition at line 527 of file vserver.h.

Referenced by `dx_limit_get()`, and `dx_limit_set()`.

#### **5.1.2.3    `uint32_t dx_limit::space_total`**

Total space

Definition at line 528 of file vserver.h.

Referenced by `dx_limit_get()`, and `dx_limit_set()`.

#### **5.1.2.4    `uint32_t dx_limit::inodes_used`**

Currently used inodes

Definition at line 529 of file vserver.h.

Referenced by `dx_limit_get()`, and `dx_limit_set()`.

#### **5.1.2.5    `uint32_t dx_limit::inodes_total`**

Total inodes

Definition at line 530 of file vserver.h.

Referenced by `dx_limit_get()`, and `dx_limit_set()`.

#### **5.1.2.6    `uint32_t dx_limit::reserved`**

Space reserved for the root user

Definition at line 531 of file vserver.h.

Referenced by `dx_limit_get()`, and `dx_limit_set()`.

#### **5.1.2.7    `uint32_t dx_limit::flags`**

Disk limit flags

Definition at line 532 of file vserver.h.

Referenced by `dx_limit_add()`, `dx_limit_get()`, and `dx_limit_set()`.

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

- **vserver.h**



## 5.2 `_ix_attr` Struct Reference

```
#include <vserver.h>
```

### 5.2.1 Detailed Description

Inode attributes.

Definition at line 588 of file `vserver.h`.

### Data Fields

- `const char * filename`
- `xid_t xid`
- `uint32_t flags`
- `uint32_t mask`

### 5.2.2 Field Documentation

#### 5.2.2.1 `const char* _ix_attr::filename`

Filename

Definition at line 589 of file `vserver.h`.

Referenced by `ix_attr_get()`, and `ix_attr_set()`.

#### 5.2.2.2 `xid_t _ix_attr::xid`

Context ID

Definition at line 590 of file `vserver.h`.

Referenced by `ix_attr_get()`, and `ix_attr_set()`.

#### 5.2.2.3 `uint32_t _ix_attr::flags`

Inode flags

Definition at line 591 of file `vserver.h`.

Referenced by `ix_attr_get()`, and `ix_attr_set()`.

#### 5.2.2.4 `uint32_t _ix_attr::mask`

Set mask

Definition at line 592 of file `vserver.h`.

Referenced by `ix_attr_get()`, and `ix_attr_set()`.

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

- `vserver.h`

## 5.3 `__nx_addr` Struct Reference

```
#include <vserver.h>
```

### 5.3.1 Detailed Description

Network address information.

Definition at line 709 of file `vserver.h`.

### Data Fields

- `uint16_t type`
- `uint16_t count`
- `uint32_t ip` [4]
- `uint32_t mask` [4]

### 5.3.2 Field Documentation

#### 5.3.2.1 `uint16_t __nx_addr::type`

Address type

Definition at line 710 of file `vserver.h`.

Referenced by `nx_addr_add()`, and `nx_addr_remove()`.

#### 5.3.2.2 `uint16_t __nx_addr::count`

Number of addresses in `ip/mask`

Definition at line 711 of file `vserver.h`.

Referenced by `nx_addr_add()`, and `nx_addr_remove()`.

#### 5.3.2.3 `uint32_t __nx_addr::ip[4]`

Up to four addresses

Definition at line 712 of file `vserver.h`.

Referenced by `nx_addr_add()`, and `nx_addr_remove()`.

#### 5.3.2.4 `uint32_t __nx_addr::mask[4]`

Up to four netmasks

Definition at line 713 of file `vserver.h`.

Referenced by `nx_addr_add()`, and `nx_addr_remove()`.

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

- `vserver.h`

## 5.4 `_nx_flags` Struct Reference

```
#include <vserver.h>
```

### 5.4.1 Detailed Description

Network context flags.

Definition at line 719 of file `vserver.h`.

### Data Fields

- `uint64_t flags`
- `uint64_t mask`

### 5.4.2 Field Documentation

#### 5.4.2.1 `uint64_t _nx_flags::flags`

Network context flags

Definition at line 720 of file `vserver.h`.

Referenced by `nx_caps_get()`, `nx_caps_set()`, `nx_create()`, `nx_flags_get()`, and `nx_flags_set()`.

#### 5.4.2.2 `uint64_t _nx_flags::mask`

Set mask

Definition at line 721 of file `vserver.h`.

Referenced by `nx_caps_get()`, `nx_caps_set()`, `nx_flags_get()`, and `nx_flags_set()`.

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

- `vserver.h`

## 5.5 `_nx_info` Struct Reference

```
#include <vserver.h>
```

### 5.5.1 Detailed Description

Network context information.

Definition at line 702 of file `vserver.h`.

### Data Fields

- `nid_t nid`

### 5.5.2 Field Documentation

#### 5.5.2.1 `nid_t _nx_info::nid`

Network context ID

Definition at line 703 of file `vserver.h`.

Referenced by `nx_info()`.

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

- `vserver.h`

## 5.6 `_nx_sock_stat` Struct Reference

```
#include <vserver.h>
```

### 5.6.1 Detailed Description

Accounting data.

Definition at line 727 of file `vserver.h`.

### Data Fields

- `uint32_t id`
- `uint32_t count` [3]
- `uint64_t total` [3]

### 5.6.2 Field Documentation

#### 5.6.2.1 `uint32_t _nx_sock_stat::id`

Socket type ID

Definition at line 728 of file `vserver.h`.

Referenced by `nx_sock_stat()`.

#### 5.6.2.2 `uint32_t _nx_sock_stat::count[3]`

Number of packets

Definition at line 729 of file `vserver.h`.

Referenced by `nx_sock_stat()`.

#### 5.6.2.3 `uint64_t _nx_sock_stat::total[3]`

Number of bytes

Definition at line 730 of file `vserver.h`.

Referenced by `nx_sock_stat()`.

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

- `vserver.h`

## 5.7 `_vx_flags` Struct Reference

`#include <vserver.h>`

### 5.7.1 Detailed Description

Context/migration flags.

Definition at line 202 of file `vserver.h`.

### Data Fields

- `uint64_t flags`
- `uint64_t mask`

### 5.7.2 Field Documentation

#### 5.7.2.1 `uint64_t _vx_flags::flags`

Flags

Definition at line 203 of file `vserver.h`.

Referenced by `vx_bcaps_get()`, `vx_bcaps_set()`, `vx_ccaps_get()`, `vx_ccaps_set()`, `vx_create()`, `vx_flags_get()`, `vx_flags_set()`, and `vx_migrate()`.

#### 5.7.2.2 `uint64_t _vx_flags::mask`

Set mask

Definition at line 204 of file `vserver.h`.

Referenced by `vx_bcaps_get()`, `vx_bcaps_set()`, `vx_ccaps_get()`, `vx_ccaps_set()`, `vx_flags_get()`, and `vx_flags_set()`.

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

- `vserver.h`

## 5.8 `_vx_info` Struct Reference

```
#include <vserver.h>
```

### 5.8.1 Detailed Description

Context information.

Definition at line 178 of file `vserver.h`.

#### Data Fields

- `xid_t xid`
- `pid_t initpid`

### 5.8.2 Field Documentation

#### 5.8.2.1 `xid_t _vx_info::xid`

Context ID

Definition at line 179 of file `vserver.h`.

Referenced by `vx_info()`.

#### 5.8.2.2 `pid_t _vx_info::initpid`

Process ID of init

Definition at line 180 of file `vserver.h`.

Referenced by `vx_info()`.

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

- `vserver.h`

## 5.9 `__vx_limit` Struct Reference

```
#include <vserver.h>
```

### 5.9.1 Detailed Description

Resource limits.

Definition at line 453 of file `vserver.h`.

### Data Fields

- `uint32_t id`
- `uint64_t minimum`
- `uint64_t softlimit`
- `uint64_t maximum`

### 5.9.2 Field Documentation

#### 5.9.2.1 `uint32_t __vx_limit::id`

Limit ID

Definition at line 454 of file `vserver.h`.

Referenced by `vx_limit_get()`, and `vx_limit_set()`.

#### 5.9.2.2 `uint64_t __vx_limit::minimum`

Minimum

Definition at line 455 of file `vserver.h`.

Referenced by `vx_limit_get()`, `vx_limit_mask_get()`, and `vx_limit_set()`.

#### 5.9.2.3 `uint64_t __vx_limit::softlimit`

Softlimit

Definition at line 456 of file `vserver.h`.

Referenced by `vx_limit_get()`, `vx_limit_mask_get()`, and `vx_limit_set()`.

#### 5.9.2.4 `uint64_t __vx_limit::maximum`

Maximum

Definition at line 457 of file `vserver.h`.

Referenced by `vx_limit_get()`, `vx_limit_mask_get()`, and `vx_limit_set()`.

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

- `vserver.h`



## 5.10 `_vx_limit_stat` Struct Reference

```
#include <vserver.h>
```

### 5.10.1 Detailed Description

Resource limit accounting.

Definition at line 463 of file `vserver.h`.

### Data Fields

- `uint32_t id`
- `uint32_t hits`
- `uint64_t value`
- `uint64_t minimum`
- `uint64_t maximum`

### 5.10.2 Field Documentation

#### 5.10.2.1 `uint32_t _vx_limit_stat::id`

Limit ID

Definition at line 464 of file `vserver.h`.

Referenced by `vx_limit_stat()`.

#### 5.10.2.2 `uint32_t _vx_limit_stat::hits`

Number of hits

Definition at line 465 of file `vserver.h`.

Referenced by `vx_limit_stat()`.

#### 5.10.2.3 `uint64_t _vx_limit_stat::value`

Current value

Definition at line 466 of file `vserver.h`.

Referenced by `vx_limit_stat()`.

#### 5.10.2.4 `uint64_t _vx_limit_stat::minimum`

Minimum value

Definition at line 467 of file `vserver.h`.

Referenced by `vx_limit_stat()`.

#### 5.10.2.5 `uint64_t _vx_limit_stat::maximum`

Maximum value

Definition at line 468 of file `vserver.h`.

Referenced by `vx_limit_stat()`.

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

- `vserver.h`

## 5.11 \_\_vx\_sched Struct Reference

```
#include <vserver.h>
```

### 5.11.1 Detailed Description

Scheduler values.

Definition at line 375 of file vserver.h.

### Data Fields

- `uint32_t mask`
- `int32_t cpu_id`
- `int32_t bucket_id`
- `int32_t fill_rate [2]`
- `int32_t interval [2]`
- `int32_t tokens`
- `int32_t tokens_min`
- `int32_t tokens_max`
- `int32_t prio_bias`

### 5.11.2 Field Documentation

#### 5.11.2.1 `uint32_t __vx_sched::mask`

Set mask

Definition at line 376 of file vserver.h.

Referenced by `vx_sched_get()`, and `vx_sched_set()`.

#### 5.11.2.2 `int32_t __vx_sched::cpu_id`

CPU ID (for SMP machines)

Definition at line 377 of file vserver.h.

Referenced by `vx_sched_get()`, and `vx_sched_set()`.

#### 5.11.2.3 `int32_t __vx_sched::bucket_id`

Token Bucket ID

Definition at line 378 of file vserver.h.

Referenced by `vx_sched_get()`.

#### 5.11.2.4 `int32_t __vx_sched::fill_rate[2]`

Fill rate

Definition at line 379 of file vserver.h.

Referenced by `vx_sched_get()`, and `vx_sched_set()`.

#### **5.11.2.5 `int32_t vx_sched::interval[2]`**

Interval between fills

Definition at line 380 of file `vserver.h`.

Referenced by `vx_sched_get()`, and `vx_sched_set()`.

#### **5.11.2.6 `int32_t vx_sched::tokens`**

Number of tokens in the bucket

Definition at line 381 of file `vserver.h`.

Referenced by `vx_sched_get()`, and `vx_sched_set()`.

#### **5.11.2.7 `int32_t vx_sched::tokens_min`**

Minimum tokens to unhold the context

Definition at line 382 of file `vserver.h`.

Referenced by `vx_sched_get()`, and `vx_sched_set()`.

#### **5.11.2.8 `int32_t vx_sched::tokens_max`**

Maximum number of tokens in the bucket

Definition at line 383 of file `vserver.h`.

Referenced by `vx_sched_get()`, and `vx_sched_set()`.

#### **5.11.2.9 `int32_t vx_sched::prio_bias`**

Priority bias

Definition at line 384 of file `vserver.h`.

Referenced by `vx_sched_get()`, and `vx_sched_set()`.

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

- **`vserver.h`**

## 5.12 `_vx_sched_info` Struct Reference

```
#include <vserver.h>
```

### 5.12.1 Detailed Description

Scheduler information.

Definition at line 390 of file `vserver.h`.

### Data Fields

- `int32_t` **cpu\_id**
- `int32_t` **bucket\_id**
- `uint64_t` **user\_msec**
- `uint64_t` **sys\_msec**
- `uint64_t` **hold\_msec**
- `uint32_t` **token\_usec**
- `int32_t` **vavavoom**

### 5.12.2 Field Documentation

#### 5.12.2.1 `int32_t _vx_sched_info::cpu_id`

Definition at line 391 of file `vserver.h`.

Referenced by `vx_sched_info()`.

#### 5.12.2.2 `int32_t _vx_sched_info::bucket_id`

Definition at line 392 of file `vserver.h`.

Referenced by `vx_sched_info()`.

#### 5.12.2.3 `uint64_t _vx_sched_info::user_msec`

Definition at line 393 of file `vserver.h`.

Referenced by `vx_sched_info()`.

#### 5.12.2.4 `uint64_t _vx_sched_info::sys_msec`

Definition at line 394 of file `vserver.h`.

Referenced by `vx_sched_info()`.

#### 5.12.2.5 `uint64_t _vx_sched_info::hold_msec`

Definition at line 395 of file `vserver.h`.

Referenced by `vx_sched_info()`.

**5.12.2.6 uint32\_t \_vx\_sched\_info::token\_usec**

Definition at line 396 of file vserver.h.

Referenced by vx\_sched\_info().

**5.12.2.7 int32\_t \_vx\_sched\_info::vavavoom**

Definition at line 397 of file vserver.h.

Referenced by vx\_sched\_info().

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

- **vserver.h**

## 5.13 `_vx_stat` Struct Reference

```
#include <vserver.h>
```

### 5.13.1 Detailed Description

Context statistics.

Definition at line 186 of file `vserver.h`.

#### Data Fields

- `uint32_t usecnt`
- `uint32_t tasks`
- `uint32_t nr_threads`
- `uint32_t nr_running`
- `uint32_t nr_unintr`
- `uint32_t nr_onhold`
- `uint32_t nr_forks`
- `uint32_t load [3]`
- `uint64_t offset`
- `uint64_t uptime`

### 5.13.2 Field Documentation

#### 5.13.2.1 `uint32_t _vx_stat::usecnt`

Number of context references

Definition at line 187 of file `vserver.h`.

Referenced by `vx_stat()`.

#### 5.13.2.2 `uint32_t _vx_stat::tasks`

Number of tasks

Definition at line 188 of file `vserver.h`.

Referenced by `vx_stat()`.

#### 5.13.2.3 `uint32_t _vx_stat::nr_threads`

Total number of threads

Definition at line 189 of file `vserver.h`.

Referenced by `vx_stat()`.

**5.13.2.4   uint32\_t \_\_vx\_stat::nr\_running**

Number of running threads

Definition at line 190 of file vserver.h.

Referenced by vx\_stat().

**5.13.2.5   uint32\_t \_\_vx\_stat::nr\_unintr**

Number of uninterruptible threads

Definition at line 191 of file vserver.h.

Referenced by vx\_stat().

**5.13.2.6   uint32\_t \_\_vx\_stat::nr\_onhold**

Number of threads being held

Definition at line 192 of file vserver.h.

Referenced by vx\_stat().

**5.13.2.7   uint32\_t \_\_vx\_stat::nr\_forks**

Total number of forks since context startup

Definition at line 193 of file vserver.h.

Referenced by vx\_stat().

**5.13.2.8   uint32\_t \_\_vx\_stat::load[3]**

Load average

Definition at line 194 of file vserver.h.

Referenced by vx\_stat().

**5.13.2.9   uint64\_t \_\_vx\_stat::offset**

Offset to the system time

Definition at line 195 of file vserver.h.

Referenced by vx\_stat().

**5.13.2.10   uint64\_t \_\_vx\_stat::uptime**

Context uptime

Definition at line 196 of file vserver.h.

Referenced by vx\_stat().

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



- `vserver.h`

## 5.14 `_vx_uname` Struct Reference

```
#include <vserver.h>
```

### 5.14.1 Detailed Description

Virtual system information data.

Definition at line 210 of file `vserver.h`.

### Data Fields

- `uint32_t id`
- `char value [65]`

### 5.14.2 Field Documentation

#### 5.14.2.1 `uint32_t _vx_uname::id`

Name ID

Definition at line 211 of file `vserver.h`.

Referenced by `vx_uname_get()`, and `vx_uname_set()`.

#### 5.14.2.2 `char _vx_uname::value[65]`

Name value

Definition at line 212 of file `vserver.h`.

Referenced by `vx_uname_get()`, and `vx_uname_set()`.

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

- `vserver.h`

## 5.15 `_vx_wait` Struct Reference

```
#include <vserver.h>
```

### 5.15.1 Detailed Description

Wait results.

Definition at line 218 of file `vserver.h`.

### Data Fields

- `int32_t reboot_cmd`
- `int32_t exit_code`

### 5.15.2 Field Documentation

#### 5.15.2.1 `int32_t _vx_wait::reboot_cmd`

Context reboot command

Definition at line 219 of file `vserver.h`.

Referenced by `vx_wait()`.

#### 5.15.2.2 `int32_t _vx_wait::exit_code`

Context exit code

Definition at line 220 of file `vserver.h`.

Referenced by `vx_wait()`.

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

- `vserver.h`



## Chapter 6

# libvserver File Documentation

### 6.1 context.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/context_cmd.h"
#include "linux/vserver/cvirt_cmd.h"
#include "linux/vserver/signal_cmd.h"
#include "vserver.h"
```

Include dependency graph for context.c:

#### Functions

- **int vx\_create (xid\_t xid, vx\_flags\_t \*data)**  
*Create a new context.*
- **int vx\_migrate (xid\_t xid, vx\_flags\_t \*data)**  
*Migrate to an existing context.*
- **int vx\_task\_xid (pid\_t pid)**  
*Get the context ID of a process.*
- **int vx\_info (xid\_t xid, vx\_info\_t \*data)**  
*Get context information.*
- **int vx\_stat (xid\_t xid, vx\_stat\_t \*data)**  
*Get context statistics.*
- **int vx\_bcaps\_set (xid\_t xid, vx\_flags\_t \*data)**

*Set system capabilities.*

- `int vx_bcaps_get (xid_t xid, vx_flags_t *data)`

*Get system capabilities.*

- `int vx_ccaps_set (xid_t xid, vx_flags_t *data)`

*Set context capabilities.*

- `int vx_ccaps_get (xid_t xid, vx_flags_t *data)`

*Get context capabilities.*

- `int vx_flags_set (xid_t xid, vx_flags_t *data)`

*Set context flags.*

- `int vx_flags_get (xid_t xid, vx_flags_t *data)`

*Get context flags.*

- `int vx_uname_set (xid_t xid, vx_uname_t *data)`

*Set virtual system information.*

- `int vx_uname_get (xid_t xid, vx_uname_t *data)`

*Get virtual system information.*

- `int vx_kill (xid_t xid, pid_t pid, int sig)`

*Kill one or more processes.*

- `int vx_wait (xid_t xid, vx_wait_t *data)`

*Wait for context death.*

## 6.2 dlimit.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/dlimit_cmd.h"
#include "vserver.h"
```

Include dependency graph for dlimit.c:

### Functions

- `int dx_limit_add (xid_t xid, dx_limit_t *data)`  
*Add disk limit entry.*
- `int dx_limit_remove (xid_t xid, dx_limit_t *data)`  
*Remove disk limit.*
- `int dx_limit_set (xid_t xid, dx_limit_t *data)`  
*Set disk limit values.*
- `int dx_limit_get (xid_t xid, dx_limit_t *data)`  
*Get disk limit values.*

## 6.3 inode.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/inode_cmd.h"
#include "vserver.h"
```

Include dependency graph for inode.c:

### Functions

- `int ix_attr_set (ix_attr_t *data)`  
*Set inode attributes.*
- `int ix_attr_get (ix_attr_t *data)`  
*Get inode attributes.*



## 6.4 limit.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/limit_cmd.h"
#include "vserver.h"
```

Include dependency graph for limit.c:

### Functions

- `int vx_limit_mask_get (vx_limit_t *data)`  
*Get resource limits mask.*
- `int vx_limit_set (xid_t xid, vx_limit_t *data)`  
*Set resource limit.*
- `int vx_limit_get (xid_t xid, vx_limit_t *data)`  
*Get resource limits.*
- `int vx_limit_stat (xid_t xid, vx_limit_stat_t *data)`  
*Get resource limit accounting data.*
- `int vx_limit_reset (xid_t xid)`  
*Reset resource limit accounting data.*

## 6.5 namespace.c File Reference

```
#include <stdint.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/space_cmd.h"
#include "vserver.h"
```

Include dependency graph for namespace.c:

### Functions

- int **ns\_clone** (int flags, void \*child\_stack)  
*Clone the current namespace (FS/IPC/UTS).*
- int **ns\_enter** (**xid\_t** xid, uint64\_t mask)  
*Enter namespace.*
- int **ns\_set** (**xid\_t** xid, uint64\_t mask)  
*Set namespace.*

## 6.6 network.c File Reference

```
#include <stdint.h>
#include <errno.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/cacct_cmd.h"
#include "linux/vserver/network_cmd.h"
#include "vserver.h"
```

Include dependency graph for network.c:

### Functions

- `int nx_create (nid_t nid, nx_flags_t *data)`  
*Create network context.*
- `int nx_migrate (nid_t nid)`  
*Migrate to an existing network context.*
- `int nx_task_nid (pid_t pid)`  
*Get the network context ID of a process.*
- `int nx_info (nid_t nid, nx_info_t *data)`  
*Get network context information.*
- `int nx_addr_add (nid_t nid, nx_addr_t *data)`  
*Add network context addresses.*
- `int nx_addr_remove (nid_t nid, nx_addr_t *data)`  
*Remove network context addresses.*
- `int nx_flags_set (nid_t nid, nx_flags_t *data)`  
*Set network context flags.*
- `int nx_flags_get (nid_t nid, nx_flags_t *data)`  
*Get network context flags.*
- `int nx_caps_set (nid_t nid, nx_flags_t *data)`  
*Set network context capabilities.*
- `int nx_caps_get (nid_t nid, nx_flags_t *data)`  
*Get network context capabilities.*
- `int nx_sock_stat (nid_t nid, nx_sock_stat_t *data)`  
*Get network socket accounting data.*

## 6.7 sched.c File Reference

```
#include <errno.h>
#include "vserver.h"
#include "linux/vserver/switch.h"
#include "linux/vserver/sched_cmd.h"
```

Include dependency graph for sched.c:

### Functions

- `int vx_sched_get (xid_t xid, vx_sched_t *data)`  
*Get scheduler values.*
- `int vx_sched_set (xid_t xid, vx_sched_t *data)`  
*Set scheduler values.*
- `int vx_sched_info (xid_t xid, vx_sched_info_t *data)`  
*Get scheduler information.*

## 6.8 switch.c File Reference

```
#include <stdint.h>
#include "linux/vserver/switch.h"
#include "linux/vserver/debug_cmd.h"
#include "vserver.h"
```

Include dependency graph for switch.c:

### Functions

- `int vs__get__version (void)`  
*Get vserver version of running kernel.*
- `int vs__get__config (void)`  
*Get vserver configuration of running kernel.*

### 6.8.1 Function Documentation

#### 6.8.1.1 `int vs__get__version (void)`

Get vserver version of running kernel.

##### Returns:

Kernel version

Definition at line 25 of file switch.c.

References `vserver()`.

```
26 {
27     return vserver(VCMD_get_version, 0, NULL);
28 }
```

#### 6.8.1.2 `int vs__get__config (void)`

Get vserver configuration of running kernel.

##### Returns:

Kernel configuration

Definition at line 30 of file switch.c.

References `vserver()`.

```
31 {
32     return vserver(VCMD_get_vci, 0, NULL);
33 }
```

## 6.9 syscall.c File Reference

```
#include <stdint.h>
```

```
#include <errno.h>
```

```
#include "syscall.h"
```

```
#include "vserver.h"
```

Include dependency graph for syscall.c:

## 6.10 vserver.h File Reference

### 6.10.1 Detailed Description

Interface to the vserver syscalls.

Definition in file **vserver.h**.

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

```
#include <stdint.h>
```

Include dependency graph for vserver.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct **\_vx\_info**  
*Context information.*
- struct **\_vx\_stat**  
*Context statistics.*
- struct **\_vx\_flags**  
*Context/migration flags.*
- struct **\_vx\_uname**  
*Virtual system information data.*
- struct **\_vx\_wait**  
*Wait results.*
- struct **\_vx\_sched**  
*Scheduler values.*
- struct **\_vx\_sched\_info**  
*Scheduler information.*
- struct **\_vx\_limit**  
*Resource limits.*
- struct **\_vx\_limit\_stat**  
*Resource limit accounting.*
- struct **\_dx\_limit**  
*Disk limit values.*

- struct **\_\_ix\_attr**  
*Inode attributes.*
- struct **\_\_nx\_info**  
*Network context information.*
- struct **\_\_nx\_addr**  
*Network address information.*
- struct **\_\_nx\_flags**  
*Network context flags.*
- struct **\_\_nx\_sock\_stat**  
*Accounting data.*

## Defines

- #define **LIBVSERVER\_API\_MAJOR** 2
- #define **LIBVSERVER\_API\_MINOR** 0
- #define **CAP\_CHOWN** 0
- #define **CAP\_DAC\_OVERRIDE** 1
- #define **CAP\_DAC\_READ\_SEARCH** 2
- #define **CAP\_FOWNER** 3
- #define **CAP\_FSETID** 4
- #define **CAP\_KILL** 5
- #define **CAP\_SETGID** 6
- #define **CAP\_SETUID** 7
- #define **CAP\_SETPCAP** 8
- #define **CAP\_LINUX\_IMMUTABLE** 9
- #define **CAP\_NET\_BIND\_SERVICE** 10
- #define **CAP\_NET\_BROADCAST** 11
- #define **CAP\_NET\_ADMIN** 12
- #define **CAP\_NET\_RAW** 13
- #define **CAP\_IPC\_LOCK** 14
- #define **CAP\_IPC\_OWNER** 15
- #define **CAP\_SYS\_MODULE** 16
- #define **CAP\_SYS\_RAWIO** 17
- #define **CAP\_SYS\_CHROOT** 18
- #define **CAP\_SYS\_PTRACE** 19
- #define **CAP\_SYS\_PACCT** 20
- #define **CAP\_SYS\_ADMIN** 21
- #define **CAP\_SYS\_BOOT** 22
- #define **CAP\_SYS\_NICE** 23
- #define **CAP\_SYS\_RESOURCE** 24
- #define **CAP\_SYS\_TIME** 25
- #define **CAP\_SYS\_TTY\_CONFIG** 26
- #define **CAP\_MKNOD** 27
- #define **CAP\_LEASE** 28
- #define **CAP\_AUDIT\_WRITE** 29



- #define CAP\_AUDIT\_CONTROL 30
- #define CAP\_CONTEXT 31
- #define VXC\_SET\_UTSNAME 0x00000001
- #define VXC\_SET\_RLIMIT 0x00000002
- #define VXC\_RAW\_ICMP 0x00000100
- #define VXC\_SYSLOG 0x00001000
- #define VXC\_SECURE\_MOUNT 0x00010000
- #define VXC\_SECURE\_REMOUNT 0x00020000
- #define VXC\_BINARY\_MOUNT 0x00040000
- #define VXC\_QUOTA\_CTL 0x00100000
- #define VXC\_ADMIN\_MAPPER 0x00200000
- #define VXC\_ADMIN\_CLOOP 0x00400000
- #define VXF\_INFO\_SCHED 0x00000002
- #define VXF\_INFO\_NPROC 0x00000004
- #define VXF\_INFO\_PRIVATE 0x00000008
- #define VXF\_INFO\_INIT 0x00000010
- #define VXF\_INFO\_HIDE 0x00000020
- #define VXF\_INFO\_ULIMIT 0x00000040
- #define VXF\_INFO\_NSPACE 0x00000080
- #define VXF\_SCHED\_HARD 0x00000100
- #define VXF\_SCHED\_PRIO 0x00000200
- #define VXF\_SCHED\_PAUSE 0x00000400
- #define VXF\_VIRT\_MEM 0x00010000
- #define VXF\_VIRT\_UPTIME 0x00020000
- #define VXF\_VIRT\_CPU 0x00040000
- #define VXF\_VIRT\_LOAD 0x00080000
- #define VXF\_VIRT\_TIME 0x00100000
- #define VXF\_HIDE\_MOUNT 0x01000000
- #define VXF\_HIDE\_NETIF 0x02000000
- #define VXF\_HIDE\_VINFO 0x04000000
- #define VXF\_STATE\_SETUP (1ULL<<32)
- #define VXF\_STATE\_INIT (1ULL<<33)
- #define VXF\_STATE\_ADMIN (1ULL<<34)
- #define VXF\_SC\_HELPER (1ULL<<36)
- #define VXF\_REBOOT\_KILL (1ULL<<37)
- #define VXF\_PERSISTENT (1ULL<<38)
- #define VXF\_FORK\_RSS (1ULL<<48)
- #define VXF\_PROLIFIC (1ULL<<49)
- #define VXF\_IGNEG\_NICE (1ULL<<52)
- #define VXM\_SET\_INIT 0x00000001
- #define VXM\_SET\_REAPER 0x00000002
- #define VHIN\_CONTEXT 0
- #define VHIN\_SYSNAME 1
- #define VHIN\_NODENAME 2
- #define VHIN\_RELEASE 3
- #define VHIN\_VERSION 4
- #define VHIN\_MACHINE 5
- #define VHIN\_DOMAINNAME 6
- #define VXSM\_FILL\_RATE 0x0001
- #define VXSM\_INTERVAL 0x0002

---

- `#define VXSM_FILL_RATE2 0x0004`
- `#define VXSM_INTERVAL2 0x0008`
- `#define VXSM_TOKENS 0x0010`
- `#define VXSM_TOKENS_MIN 0x0020`
- `#define VXSM_TOKENS_MAX 0x0040`
- `#define VXSM_PRIO_BIAS 0x0100`
- `#define VXSM_IDLE_TIME 0x0200`
- `#define VXSM_FORCE 0x0400`
- `#define VXSM_CPU_ID 0x1000`
- `#define VXSM_BUCKET_ID 0x2000`
- `#define VXSM_MSEC 0x4000`
- `#define VXSM_V3_MASK 0x0173`
- `#define VXSM_SET_MASK 0x01FF`
- `#define CRLIM_UNSET (0ULL)`
- `#define CRLIM_INFINITY (~0ULL)`
- `#define CRLIM_KEEP (~1ULL)`
- `#define VLIMIT_NSOK 16`
- `#define VLIMIT_OPENFD 17`
- `#define VLIMIT_ANON 18`
- `#define VLIMIT_SHMEM 19`
- `#define VLIMIT_SEMARI 20`
- `#define VLIMIT_NSEMS 21`
- `#define VLIMIT_DENTRY 22`
- `#define VLIMIT_MAPPED 23`
- `#define CDLIM_UNSET ((uint32_t)0UL)`
- `#define CDLIM_INFINITY ((uint32_t)~0UL)`
- `#define CDLIM_KEEP ((uint32_t)~1UL)`
- `#define IATTR_TAG 0x01000000`
- `#define IATTR_ADMIN 0x00000001`
- `#define IATTR_WATCH 0x00000002`
- `#define IATTR_HIDE 0x00000004`
- `#define IATTR_FLAGS 0x00000007`
- `#define IATTR_BARRIER 0x00010000`
- `#define IATTR_IUNLINK 0x00020000`
- `#define IATTR_IMMUTABLE 0x00040000`
- `#define CLONE_VM 0x00000100`
- `#define CLONE_FS 0x00000200`
- `#define CLONE_FILES 0x00000400`
- `#define CLONE_SIGHAND 0x00000800`
- `#define CLONE_PTRACE 0x00002000`
- `#define CLONE_VFORK 0x00004000`
- `#define CLONE_PARENT 0x00008000`
- `#define CLONE_THREAD 0x00010000`
- `#define CLONE_NEWNS 0x00020000`
- `#define CLONE_SYSVSEM 0x00040000`
- `#define CLONE_SETTLS 0x00080000`
- `#define CLONE_PARENT_SETTID 0x00100000`
- `#define CLONE_CHILD_CLEARPID 0x00200000`
- `#define CLONE_DETACHED 0x00400000`
- `#define CLONE_UNTRACED 0x00800000`

---

- `#define CLONE_CHILD_SETTID 0x01000000`
- `#define CLONE_STOPPED 0x02000000`
- `#define CLONE_NEWUTS 0x04000000`
- `#define CLONE_NEWIPC 0x08000000`
- `#define CLONE_KTHREAD 0x10000000`
- `#define NXF_INFO_PRIVATE 0x00000008`
- `#define NXF_STATE_SETUP (1ULL<<32)`
- `#define NXF_STATE_ADMIN (1ULL<<34)`
- `#define NXF_SC_HELPER (1ULL<<36)`
- `#define NXF_PERSISTENT (1ULL<<38)`
- `#define NXA_TYPE_IPV4 1`
- `#define NXA_TYPE_IPV6 2`
- `#define NXA_MOD_BCAST (1<<8)`
- `#define NXA_TYPE_ANY ((uint16_t)-1)`
- `#define VXA_SOCKET_UNSPEC 0`
- `#define VXA_SOCKET_UNIX 1`
- `#define VXA_SOCKET_INET 2`
- `#define VXA_SOCKET_INET6 3`
- `#define VXA_SOCKET_PACKET 4`
- `#define VXA_SOCKET_OTHER 5`
- `#define NXA_SOCKET_UNSPEC VXA_SOCKET_UNSPEC`
- `#define NXA_SOCKET_UNIX VXA_SOCKET_UNIX`
- `#define NXA_SOCKET_INET VXA_SOCKET_INET`
- `#define NXA_SOCKET_INET6 VXA_SOCKET_INET6`
- `#define NXA_SOCKET_PACKET VXA_SOCKET_PACKET`
- `#define NXA_SOCKET_OTHER VXA_SOCKET_OTHER`

## Typedefs

- `typedef uint32_t xid_t`
- `typedef __vx_info vx_info_t`  
*Context information.*
- `typedef __vx_stat vx_stat_t`  
*Context statistics.*
- `typedef __vx_flags vx_flags_t`  
*Context/migration flags.*
- `typedef __vx_uname vx_uname_t`  
*Virtual system information data.*
- `typedef __vx_wait vx_wait_t`  
*Wait results.*
- `typedef __vx_sched vx_sched_t`  
*Scheduler values.*
- `typedef __vx_sched_info vx_sched_info_t`

*Scheduler information.*

- typedef **\_vx\_limit vx\_limit\_t**  
*Resource limits.*
- typedef **\_vx\_limit\_stat vx\_limit\_stat\_t**  
*Resource limit accounting.*
- typedef **\_dx\_limit dx\_limit\_t**  
*Disk limit values.*
- typedef **\_ix\_attr ix\_attr\_t**  
*Inode attributes.*
- typedef **uint32\_t nid\_t**
- typedef **\_nx\_info nx\_info\_t**  
*Network context information.*
- typedef **\_nx\_addr nx\_addr\_t**  
*Network address information.*
- typedef **\_nx\_flags nx\_flags\_t**  
*Network context flags.*
- typedef **\_nx\_sock\_stat nx\_sock\_stat\_t**  
*Accounting data.*

## Functions

- int **vserver** (uint32\_t cmd, uint32\_t id, void \*data)  
*Main vserver syscall interface.*
- int **clone** (int flags, void \*child\_stack)  
*Clone system call.*
- int **vs\_get\_version** (void)  
*Get vserver version of running kernel.*
- int **vs\_get\_config** (void)  
*Get vserver configuration of running kernel.*
- int **vx\_create** (xid\_t xid, vx\_flags\_t \*data)  
*Create a new context.*
- int **vx\_migrate** (xid\_t xid, vx\_flags\_t \*data)  
*Migrate to an existing context.*
- int **vx\_task\_xid** (pid\_t pid)  
*Get the context ID of a process.*

- `int vx_info (xid_t xid, vx_info_t *data)`  
*Get context information.*
- `int vx_stat (xid_t xid, vx_stat_t *data)`  
*Get context statistics.*
- `int vx_bcaps_set (xid_t xid, vx_flags_t *data)`  
*Set system capabilities.*
- `int vx_bcaps_get (xid_t xid, vx_flags_t *data)`  
*Get system capabilities.*
- `int vx_ccaps_set (xid_t xid, vx_flags_t *data)`  
*Set context capabilities.*
- `int vx_ccaps_get (xid_t xid, vx_flags_t *data)`  
*Get context capabilities.*
- `int vx_flags_set (xid_t xid, vx_flags_t *data)`  
*Set context flags.*
- `int vx_flags_get (xid_t xid, vx_flags_t *data)`  
*Get context flags.*
- `int vx_uname_set (xid_t xid, vx_uname_t *data)`  
*Set virtual system information.*
- `int vx_uname_get (xid_t xid, vx_uname_t *data)`  
*Get virtual system information.*
- `int vx_kill (xid_t xid, pid_t pid, int sig)`  
*Kill one or more processes.*
- `int vx_wait (xid_t xid, vx_wait_t *data)`  
*Wait for context death.*
- `int vx_sched_set (xid_t xid, vx_sched_t *data)`  
*Set scheduler values.*
- `int vx_sched_get (xid_t xid, vx_sched_t *data)`  
*Get scheduler values.*
- `int vx_sched_info (xid_t xid, vx_sched_info_t *data)`  
*Get scheduler information.*
- `int vx_limit_mask_get (vx_limit_t *data)`  
*Get resource limits mask.*
- `int vx_limit_set (xid_t xid, vx_limit_t *data)`

*Set resource limit.*

- `int vx_limit_get (xid_t xid, vx_limit_t *data)`  
*Get resource limits.*
- `int vx_limit_stat (xid_t xid, vx_limit_stat_t *data)`  
*Get resource limit accounting data.*
- `int vx_limit_reset (xid_t xid)`  
*Reset resource limit accounting data.*
- `int dx_limit_add (xid_t xid, dx_limit_t *data)`  
*Add disk limit entry.*
- `int dx_limit_remove (xid_t xid, dx_limit_t *data)`  
*Remove disk limit.*
- `int dx_limit_set (xid_t xid, dx_limit_t *data)`  
*Set disk limit values.*
- `int dx_limit_get (xid_t xid, dx_limit_t *data)`  
*Get disk limit values.*
- `int ix_attr_set (ix_attr_t *data)`  
*Set inode attributes.*
- `int ix_attr_get (ix_attr_t *data)`  
*Get inode attributes.*
- `int ns_clone (int flags, void *child_stack)`  
*Clone the current namespace (FS/IPC/UTS).*
- `int ns_enter (xid_t xid, uint64_t mask)`  
*Enter namespace.*
- `int ns_set (xid_t xid, uint64_t mask)`  
*Set namespace.*
- `int nx_create (nid_t nid, nx_flags_t *data)`  
*Create network context.*
- `int nx_migrate (nid_t nid)`  
*Migrate to an existing network context.*
- `int nx_task_nid (pid_t pid)`  
*Get the network context ID of a process.*
- `int nx_info (nid_t nid, nx_info_t *data)`  
*Get network context information.*

- `int nx_addr_add (nid_t nid, nx_addr_t *data)`  
*Add network context addresses.*
- `int nx_addr_remove (nid_t nid, nx_addr_t *data)`  
*Remove network context addresses.*
- `int nx_flags_set (nid_t nid, nx_flags_t *data)`  
*Set network context flags.*
- `int nx_flags_get (nid_t nid, nx_flags_t *data)`  
*Get network context flags.*
- `int nx_caps_set (nid_t nid, nx_flags_t *data)`  
*Set network context capabilities.*
- `int nx_caps_get (nid_t nid, nx_flags_t *data)`  
*Get network context capabilities.*
- `int nx_sock_stat (nid_t nid, nx_sock_stat_t *data)`  
*Get network socket accounting data.*

## 6.10.2 Define Documentation

### 6.10.2.1 `#define LIBVSERVER_API_MAJOR 2`

API major version

Definition at line 28 of file vserver.h.

### 6.10.2.2 `#define LIBVSERVER_API_MINOR 0`

API minor version

Definition at line 29 of file vserver.h.

## 6.10.3 Function Documentation

### 6.10.3.1 `int vserver (uint32_t cmd, uint32_t id, void * data)`

Main vserver syscall interface.

#### Parameters:

*cmd* Command number

*id* Context ID (sometimes process id)

*data* Data structures

#### Returns:

Syscall return code

See also:

Referenced by `dx_limit_add()`, `dx_limit_get()`, `dx_limit_remove()`, `dx_limit_set()`, `ix_attr_get()`, `ix_attr_set()`, `ns_enter()`, `ns_set()`, `nx_addr_add()`, `nx_addr_remove()`, `nx_caps_get()`, `nx_caps_set()`, `nx_create()`, `nx_flags_get()`, `nx_flags_set()`, `nx_info()`, `nx_migrate()`, `nx_sock_stat()`, `nx_task_nid()`, `vs_get_config()`, `vs_get_version()`, `vx_bcaps_get()`, `vx_bcaps_set()`, `vx_ccaps_get()`, `vx_ccaps_set()`, `vx_create()`, `vx_flags_get()`, `vx_flags_set()`, `vx_info()`, `vx_kill()`, `vx_limit_get()`, `vx_limit_mask_get()`, `vx_limit_reset()`, `vx_limit_set()`, `vx_limit_stat()`, `vx_migrate()`, `vx_sched_get()`, `vx_sched_info()`, `vx_sched_set()`, `vx_stat()`, `vx_task_xid()`, `vx_undefine_get()`, `vx_undefine_set()`, and `vx_wait()`.

### 6.10.3.2 `int clone (int flags, void * child_stack)`

Clone system call.

**Parameters:**

*flags* Clone flags  
*child\_stack* Child stack

**Returns:**

Process ID in parent, 0 in child, -1 on error

See also:

`clone(2)`

Referenced by `ns_clone()`.

### 6.10.3.3 `int vs_get_version (void)`

Get vserver version of running kernel.

**Returns:**

Kernel version

Definition at line 25 of file `switch.c`.

References `vserver()`.

```
26 {
27     return vserver(VCMD_get_version, 0, NULL);
28 }
```

### 6.10.3.4 `int vs_get_config (void)`

Get vserver configuration of running kernel.

**Returns:**

Kernel configuration



Definition at line 30 of file switch.c.

References `vserver()`.

```
31 {  
32     return vserver(VCMD_get_vci, 0, NULL);  
33 }
```

# Index

- `_dx_limit`, 59
  - filename, 59
  - flags, 60
  - inodes\_total, 60
  - inodes\_used, 60
  - reserved, 60
  - space\_total, 60
  - space\_used, 59
- `_ix_attr`, 61
  - filename, 61
  - flags, 61
  - mask, 61
  - xid, 61
- `_nx_addr`, 62
  - count, 62
  - ip, 62
  - mask, 62
  - type, 62
- `_nx_flags`, 63
  - flags, 63
  - mask, 63
- `_nx_info`, 64
  - nid, 64
- `_nx_sock_stat`, 65
  - count, 65
  - id, 65
  - total, 65
- `_vx_flags`, 66
  - flags, 66
  - mask, 66
- `_vx_info`, 67
  - initpid, 67
  - xid, 67
- `_vx_limit`, 68
  - id, 68
  - maximum, 68
  - minimum, 68
  - softlimit, 68
- `_vx_limit_stat`, 69
  - hits, 69
  - id, 69
  - maximum, 69
  - minimum, 69
  - value, 69
- `_vx_sched`, 71
  - bucket\_id, 71
  - cpu\_id, 71
  - fill\_rate, 71
  - interval, 72
  - mask, 71
  - prio\_bias, 72
  - tokens, 72
  - tokens\_max, 72
  - tokens\_min, 72
- `_vx_sched_info`, 73
  - bucket\_id, 73
  - cpu\_id, 73
  - hold\_msec, 73
  - sys\_msec, 73
  - token\_usec, 73
  - user\_msec, 73
  - vavavoom, 74
- `_vx_stat`, 75
  - load, 76
  - nr\_forks, 76
  - nr\_onhold, 76
  - nr\_running, 75
  - nr\_threads, 75
  - nr\_unintr, 76
  - offset, 76
  - tasks, 75
  - uptime, 76
  - usecnt, 75
- `_vx_uname`, 78
  - id, 78
  - value, 78
- `_vx_wait`, 79
  - exit\_code, 79
  - reboot\_cmd, 79
- `bucket_id`
  - `_vx_sched`, 71
  - `_vx_sched_info`, 73
- `CAP_AUDIT_CONTROL`
  - `syscall_context`, 13
- `CAP_AUDIT_WRITE`
  - `syscall_context`, 13
- `CAP_CHOWN`
  - `syscall_context`, 10

- CAP\_CONTEXT
  - syscall\_context, 13
- CAP\_DAC\_OVERRIDE
  - syscall\_context, 10
- CAP\_DAC\_READ\_SEARCH
  - syscall\_context, 10
- CAP\_FOWNER
  - syscall\_context, 10
- CAP\_FSETID
  - syscall\_context, 11
- CAP\_IPC\_LOCK
  - syscall\_context, 11
- CAP\_IPC\_OWNER
  - syscall\_context, 11
- CAP\_KILL
  - syscall\_context, 11
- CAP\_LEASE
  - syscall\_context, 13
- CAP\_LINUX\_IMMUTABLE
  - syscall\_context, 11
- CAP\_MKNOD
  - syscall\_context, 12
- CAP\_NET\_ADMIN
  - syscall\_context, 11
- CAP\_NET\_BIND\_SERVICE
  - syscall\_context, 11
- CAP\_NET\_BROADCAST
  - syscall\_context, 11
- CAP\_NET\_RAW
  - syscall\_context, 11
- CAP\_SETGID
  - syscall\_context, 11
- CAP\_SETPCAP
  - syscall\_context, 11
- CAP\_SETUID
  - syscall\_context, 11
- CAP\_SYS\_ADMIN
  - syscall\_context, 12
- CAP\_SYS\_BOOT
  - syscall\_context, 12
- CAP\_SYS\_CHROOT
  - syscall\_context, 12
- CAP\_SYS\_MODULE
  - syscall\_context, 12
- CAP\_SYS\_NICE
  - syscall\_context, 12
- CAP\_SYS\_PACCT
  - syscall\_context, 12
- CAP\_SYS\_PTRACE
  - syscall\_context, 12
- CAP\_SYS\_RAWIO
  - syscall\_context, 12
- CAP\_SYS\_RESOURCE
  - syscall\_context, 12
- CAP\_SYS\_TIME
  - syscall\_context, 12
- CAP\_SYS\_TTY\_CONFIG
  - syscall\_context, 12
- CDLIM\_INFINITY
  - syscall\_dlimit, 37
- CDLIM\_KEEP
  - syscall\_dlimit, 37
- CDLIM\_UNSET
  - syscall\_dlimit, 37
- clone
  - vserver.h, 100
- CLONE\_CHILD\_CLEARTID
  - syscall\_space, 45
- CLONE\_CHILD\_SETTID
  - syscall\_space, 46
- CLONE\_DETACHED
  - syscall\_space, 45
- CLONE\_FILES
  - syscall\_space, 44
- CLONE\_FS
  - syscall\_space, 44
- CLONE\_KTHREAD
  - syscall\_space, 46
- CLONE\_NEWIPC
  - syscall\_space, 46
- CLONE\_NEWNS
  - syscall\_space, 45
- CLONE\_NEWUTS
  - syscall\_space, 46
- CLONE\_PARENT
  - syscall\_space, 45
- CLONE\_PARENT\_SETTID
  - syscall\_space, 45
- CLONE\_PTRACE
  - syscall\_space, 45
- CLONE\_SETTLS
  - syscall\_space, 45
- CLONE\_SIGHAND
  - syscall\_space, 45
- CLONE\_STOPPED
  - syscall\_space, 46
- CLONE\_SYSVSEM
  - syscall\_space, 45
- CLONE\_THREAD
  - syscall\_space, 45
- CLONE\_UNTRACED
  - syscall\_space, 46
- CLONE\_VFORK
  - syscall\_space, 45
- CLONE\_VM
  - syscall\_space, 44
- Context commands, 7
- context.c, 81

- count
  - \_nx\_addr, 62
  - \_nx\_sock\_stat, 65
- CPU scheduler commands, 27
- cpu\_id
  - \_vx\_sched, 71
  - \_vx\_sched\_info, 73
- CRLIM\_INFINITY
  - syscall\_rlimit, 33
- CRLIM\_KEEP
  - syscall\_rlimit, 33
- CRLIM\_UNSET
  - syscall\_rlimit, 33
- Disk limit commands, 37
- dlimit.c, 83
- dx\_limit\_add
  - syscall\_dlimit, 38
- dx\_limit\_get
  - syscall\_dlimit, 39
- dx\_limit\_remove
  - syscall\_dlimit, 38
- dx\_limit\_set
  - syscall\_dlimit, 39
- dx\_limit\_t
  - syscall\_dlimit, 38
- exit\_code
  - \_vx\_wait, 79
- filename
  - \_dx\_limit, 59
  - \_ix\_attr, 61
- fill\_rate
  - \_vx\_sched, 71
- flags
  - \_dx\_limit, 60
  - \_ix\_attr, 61
  - \_nx\_flags, 63
  - \_vx\_flags, 66
- hits
  - \_vx\_limit\_stat, 69
- hold\_msec
  - \_vx\_sched\_info, 73
- IATTR\_ADMIN
  - syscall\_inode, 41
- IATTR\_BARRIER
  - syscall\_inode, 42
- IATTR\_FLAGS
  - syscall\_inode, 42
- IATTR\_HIDE
  - syscall\_inode, 42
- IATTR\_IMMUTABLE
  - syscall\_inode, 42
- IATTR\_IUNLINK
  - syscall\_inode, 42
- IATTR\_TAG
  - syscall\_inode, 41
- IATTR\_WATCH
  - syscall\_inode, 41
- id
  - \_nx\_sock\_stat, 65
  - \_vx\_limit, 68
  - \_vx\_limit\_stat, 69
  - \_vx\_uname, 78
- initpid
  - \_vx\_info, 67
- Inode attribute commands, 41
- inode.c, 84
- inodes\_total
  - \_dx\_limit, 60
- inodes\_used
  - \_dx\_limit, 60
- interval
  - \_vx\_sched, 72
- ip
  - \_nx\_addr, 62
- ix\_attr\_get
  - syscall\_inode, 43
- ix\_attr\_set
  - syscall\_inode, 42
- ix\_attr\_t
  - syscall\_inode, 42
- LIBVSERVER\_API\_MAJOR
  - vserver.h, 99
- LIBVSERVER\_API\_MINOR
  - vserver.h, 99
- limit.c, 85
- load
  - \_vx\_stat, 76
- mask
  - \_ix\_attr, 61
  - \_nx\_addr, 62
  - \_nx\_flags, 63
  - \_vx\_flags, 66
  - \_vx\_sched, 71
- maximum
  - \_vx\_limit, 68
  - \_vx\_limit\_stat, 69
- minimum
  - \_vx\_limit, 68
  - \_vx\_limit\_stat, 69
- Namespace commands, 44
- namespace.c, 86

- Network context commands, 48
- network.c, 87
- nid
  - \_nx\_info, 64
- nid\_t
  - syscall\_network, 52
- nr\_forks
  - \_vx\_stat, 76
- nr\_onhold
  - \_vx\_stat, 76
- nr\_running
  - \_vx\_stat, 75
- nr\_threads
  - \_vx\_stat, 75
- nr\_unintr
  - \_vx\_stat, 76
- ns\_clone
  - syscall\_space, 46
- ns\_enter
  - syscall\_space, 46
- ns\_set
  - syscall\_space, 47
- nx\_addr\_add
  - syscall\_network, 54
- nx\_addr\_remove
  - syscall\_network, 54
- nx\_addr\_t
  - syscall\_network, 52
- nx\_caps\_get
  - syscall\_network, 56
- nx\_caps\_set
  - syscall\_network, 56
- nx\_create
  - syscall\_network, 52
- nx\_flags\_get
  - syscall\_network, 55
- nx\_flags\_set
  - syscall\_network, 55
- nx\_flags\_t
  - syscall\_network, 52
- nx\_info
  - syscall\_network, 53
- nx\_info\_t
  - syscall\_network, 52
- nx\_migrate
  - syscall\_network, 52
- nx\_sock\_stat
  - syscall\_network, 56
- nx\_sock\_stat\_t
  - syscall\_network, 52
- nx\_task\_nid
  - syscall\_network, 53
- NXA\_MOD\_BCAST
  - syscall\_network, 50
- NXA\_SOCKET\_INET
  - syscall\_network, 51
- NXA\_SOCKET\_INET6
  - syscall\_network, 51
- NXA\_SOCKET\_OTHER
  - syscall\_network, 51
- NXA\_SOCKET\_PACKET
  - syscall\_network, 51
- NXA\_SOCKET\_UNIX
  - syscall\_network, 51
- NXA\_SOCKET\_UNSPEC
  - syscall\_network, 51
- NXA\_TYPE\_ANY
  - syscall\_network, 50
- NXA\_TYPE\_IPV4
  - syscall\_network, 50
- NXA\_TYPE\_IPV6
  - syscall\_network, 50
- NXF\_INFO\_PRIVATE
  - syscall\_network, 49
- NXF\_PERSISTENT
  - syscall\_network, 50
- NXF\_SC\_HELPER
  - syscall\_network, 50
- NXF\_STATE\_ADMIN
  - syscall\_network, 50
- NXF\_STATE\_SETUP
  - syscall\_network, 49
- offset
  - \_vx\_stat, 76
- prio\_bias
  - \_vx\_sched, 72
- reboot\_cmd
  - \_vx\_wait, 79
- reserved
  - \_dx\_limit, 60
- Resource limit commands, 32
- sched.c, 88
- softlimit
  - \_vx\_limit, 68
- space\_total
  - \_dx\_limit, 60
- space\_used
  - \_dx\_limit, 59
- switch.c, 89
  - vs\_get\_config, 89
  - vs\_get\_version, 89
- sys\_msec
  - \_vx\_sched\_info, 73
- syscall.c, 90

## syscall\_context

CAP\_AUDIT\_CONTROL, 13  
 CAP\_AUDIT\_WRITE, 13  
 CAP\_CHOWN, 10  
 CAP\_CONTEXT, 13  
 CAP\_DAC\_OVERRIDE, 10  
 CAP\_DAC\_READ\_SEARCH, 10  
 CAP\_FOWNER, 10  
 CAP\_FSETID, 11  
 CAP\_IPC\_LOCK, 11  
 CAP\_IPC\_OWNER, 11  
 CAP\_KILL, 11  
 CAP\_LEASE, 13  
 CAP\_LINUX\_IMMUTABLE, 11  
 CAP\_MKNOD, 12  
 CAP\_NET\_ADMIN, 11  
 CAP\_NET\_BIND\_SERVICE, 11  
 CAP\_NET\_BROADCAST, 11  
 CAP\_NET\_RAW, 11  
 CAP\_SETGID, 11  
 CAP\_SETPCAP, 11  
 CAP\_SETUID, 11  
 CAP\_SYS\_ADMIN, 12  
 CAP\_SYS\_BOOT, 12  
 CAP\_SYS\_CHROOT, 12  
 CAP\_SYS\_MODULE, 12  
 CAP\_SYS\_NICE, 12  
 CAP\_SYS\_PACCT, 12  
 CAP\_SYS\_PTRACE, 12  
 CAP\_SYS\_RAWIO, 12  
 CAP\_SYS\_RESOURCE, 12  
 CAP\_SYS\_TIME, 12  
 CAP\_SYS\_TTY\_CONFIG, 12  
 VHIN\_CONTEXT, 17  
 VHIN\_DOMAINNAME, 18  
 VHIN\_MACHINE, 18  
 VHIN\_NODENAME, 17  
 VHIN\_RELEASE, 18  
 VHIN\_SYSNAME, 17  
 VHIN\_VERSION, 18  
 vx\_bcaps\_get, 21  
 vx\_bcaps\_set, 21  
 vx\_ccaps\_get, 22  
 vx\_ccaps\_set, 22  
 vx\_create, 19  
 vx\_flags\_get, 23  
 vx\_flags\_set, 23  
 vx\_flags\_t, 18  
 vx\_info, 20  
 vx\_info\_t, 18  
 vx\_kill, 25  
 vx\_migrate, 19  
 vx\_stat, 20  
 vx\_stat\_t, 18

## vx\_task\_xid, 19

vx\_uname\_get, 24  
 vx\_uname\_set, 24  
 vx\_uname\_t, 18  
 vx\_wait, 25  
 vx\_wait\_t, 18  
 VXC\_ADMIN\_CLOOP, 14  
 VXC\_ADMIN\_MAPPER, 14  
 VXC\_BINARY\_MOUNT, 14  
 VXC\_QUOTA\_CTL, 14  
 VXC\_RAW\_ICMP, 13  
 VXC\_SECURE\_MOUNT, 13  
 VXC\_SECURE\_REMOUNT, 13  
 VXC\_SET\_RLIMIT, 13  
 VXC\_SET\_UTSNAME, 13  
 VXC\_SYSLOG, 13  
 VXF\_FORK\_RSS, 17  
 VXF\_HIDE\_MOUNT, 16  
 VXF\_HIDE\_NETIF, 16  
 VXF\_HIDE\_VINFO, 16  
 VXF\_IGNEG\_NICE, 17  
 VXF\_INFO\_HIDE, 14  
 VXF\_INFO\_INIT, 14  
 VXF\_INFO\_NPROC, 14  
 VXF\_INFO\_NSPACE, 15  
 VXF\_INFO\_PRIVATE, 14  
 VXF\_INFO\_SCHED, 14  
 VXF\_INFO\_ULIMIT, 15  
 VXF\_PERSISTENT, 17  
 VXF\_PROLIFIC, 17  
 VXF\_REBOOT\_KILL, 16  
 VXF\_SC\_HELPER, 16  
 VXF\_SCHED\_HARD, 15  
 VXF\_SCHED\_PAUSE, 15  
 VXF\_SCHED\_PRIO, 15  
 VXF\_STATE\_ADMIN, 16  
 VXF\_STATE\_INIT, 16  
 VXF\_STATE\_SETUP, 16  
 VXF\_VIRT\_CPU, 15  
 VXF\_VIRT\_LOAD, 15  
 VXF\_VIRT\_MEM, 15  
 VXF\_VIRT\_TIME, 16  
 VXF\_VIRT\_UPTIME, 15  
 VXM\_SET\_INIT, 17  
 VXM\_SET\_REAPER, 17  
 xid\_t, 18

## syscall\_dlimit

CDLIM\_INFINITY, 37  
 CDLIM\_KEEP, 37  
 CDLIM\_UNSET, 37  
 dx\_limit\_add, 38  
 dx\_limit\_get, 39  
 dx\_limit\_remove, 38  
 dx\_limit\_set, 39

- dx\_limit\_t, 38
- syscall\_inode
  - IATTR\_ADMIN, 41
  - IATTR\_BARRIER, 42
  - IATTR\_FLAGS, 42
  - IATTR\_HIDE, 42
  - IATTR\_IMMUTABLE, 42
  - IATTR\_IUNLINK, 42
  - IATTR\_TAG, 41
  - IATTR\_WATCH, 41
  - ix\_attr\_get, 43
  - ix\_attr\_set, 42
  - ix\_attr\_t, 42
- syscall\_network
  - nid\_t, 52
  - nx\_addr\_add, 54
  - nx\_addr\_remove, 54
  - nx\_addr\_t, 52
  - nx\_caps\_get, 56
  - nx\_caps\_set, 56
  - nx\_create, 52
  - nx\_flags\_get, 55
  - nx\_flags\_set, 55
  - nx\_flags\_t, 52
  - nx\_info, 53
  - nx\_info\_t, 52
  - nx\_migrate, 52
  - nx\_sock\_stat, 56
  - nx\_sock\_stat\_t, 52
  - nx\_task\_nid, 53
  - NXA\_MOD\_BCAST, 50
  - NXA\_SOCKET\_INET, 51
  - NXA\_SOCKET\_INET6, 51
  - NXA\_SOCKET\_OTHER, 51
  - NXA\_SOCKET\_PACKET, 51
  - NXA\_SOCKET\_UNIX, 51
  - NXA\_SOCKET\_UNSPEC, 51
  - NXA\_TYPE\_ANY, 50
  - NXA\_TYPE\_IPV4, 50
  - NXA\_TYPE\_IPV6, 50
  - NXF\_INFO\_PRIVATE, 49
  - NXF\_PERSISTENT, 50
  - NXF\_SC\_HELPER, 50
  - NXF\_STATE\_ADMIN, 50
  - NXF\_STATE\_SETUP, 49
  - VXA\_SOCKET\_INET, 51
  - VXA\_SOCKET\_INET6, 51
  - VXA\_SOCKET\_OTHER, 51
  - VXA\_SOCKET\_PACKET, 51
  - VXA\_SOCKET\_UNIX, 50
  - VXA\_SOCKET\_UNSPEC, 50
- syscall\_rlimit
  - CRLIM\_INFINITY, 33
  - CRLIM\_KEEP, 33
  - CRLIM\_UNSET, 33
  - VLIMIT\_ANON, 33
  - VLIMIT\_DENTRY, 34
  - VLIMIT\_MAPPED, 34
  - VLIMIT\_NSEMS, 33
  - VLIMIT\_NSOCK, 33
  - VLIMIT\_OPENFD, 33
  - VLIMIT\_SEMARY, 33
  - VLIMIT\_SHMEM, 33
  - vx\_limit\_get, 35
  - vx\_limit\_mask\_get, 34
  - vx\_limit\_reset, 36
  - vx\_limit\_set, 34
  - vx\_limit\_stat, 35
  - vx\_limit\_stat\_t, 34
  - vx\_limit\_t, 34
- syscall\_sched
  - vx\_sched\_get, 30
  - vx\_sched\_info, 31
  - vx\_sched\_info\_t, 29
  - vx\_sched\_set, 30
  - vx\_sched\_t, 29
  - VXSM\_BUCKET\_ID, 29
  - VXSM\_CPU\_ID, 29
  - VXSM\_FILL\_RATE, 28
  - VXSM\_FILL\_RATE2, 28
  - VXSM\_FORCE, 29
  - VXSM\_IDLE\_TIME, 28
  - VXSM\_INTERVAL, 28
  - VXSM\_INTERVAL2, 28
  - VXSM\_MSEC, 29
  - VXSM\_PRIO\_BIAS, 28
  - VXSM\_SET\_MASK, 29
  - VXSM\_TOKENS, 28
  - VXSM\_TOKENS\_MAX, 28
  - VXSM\_TOKENS\_MIN, 28
  - VXSM\_V3\_MASK, 29
- syscall\_space
  - CLONE\_CHILD\_CLEARPID, 45
  - CLONE\_CHILD\_SETTID, 46
  - CLONE\_DETACHED, 45
  - CLONE\_FILES, 44
  - CLONE\_FS, 44
  - CLONE\_KTHREAD, 46
  - CLONE\_NEWIPC, 46
  - CLONE\_NEWNS, 45
  - CLONE\_NEWUTS, 46
  - CLONE\_PARENT, 45
  - CLONE\_PARENT\_SETTID, 45
  - CLONE\_PTRACE, 45
  - CLONE\_SETTID, 45
  - CLONE\_SIGHAND, 45
  - CLONE\_STOPPED, 46
  - CLONE\_SYSVSEM, 45

- CLONE\_THREAD, 45
- CLONE\_UNTRACED, 46
- CLONE\_VFORK, 45
- CLONE\_VM, 44
- ns\_clone, 46
- ns\_enter, 46
- ns\_set, 47
- tasks
  - \_vx\_stat, 75
- token\_usec
  - \_vx\_sched\_info, 73
- tokens
  - \_vx\_sched, 72
- tokens\_max
  - \_vx\_sched, 72
- tokens\_min
  - \_vx\_sched, 72
- total
  - \_nx\_sock\_stat, 65
- type
  - \_nx\_addr, 62
- uptime
  - \_vx\_stat, 76
- usecnt
  - \_vx\_stat, 75
- user\_msec
  - \_vx\_sched\_info, 73
- value
  - \_vx\_limit\_stat, 69
  - \_vx\_uname, 78
- vavavoom
  - \_vx\_sched\_info, 74
- VHIN\_CONTEXT
  - syscall\_context, 17
- VHIN\_DOMAINNAME
  - syscall\_context, 18
- VHIN\_MACHINE
  - syscall\_context, 18
- VHIN\_NODENAME
  - syscall\_context, 17
- VHIN\_RELEASE
  - syscall\_context, 18
- VHIN\_SYSNAME
  - syscall\_context, 17
- VHIN\_VERSION
  - syscall\_context, 18
- VLIMIT\_ANON
  - syscall\_rlimit, 33
- VLIMIT\_DENTRY
  - syscall\_rlimit, 34
- VLIMIT\_MAPPED
  - syscall\_rlimit, 34
- VLIMIT\_NSEMS
  - syscall\_rlimit, 33
- VLIMIT\_NSOCK
  - syscall\_rlimit, 33
- VLIMIT\_OPENFD
  - syscall\_rlimit, 33
- VLIMIT\_SEMAY
  - syscall\_rlimit, 33
- VLIMIT\_SHMEM
  - syscall\_rlimit, 33
- vs\_get\_config
  - switch.c, 89
  - vserver.h, 100
- vs\_get\_version
  - switch.c, 89
  - vserver.h, 100
- vserver
  - vserver.h, 99
- vserver.h, 91
  - clone, 100
  - LIBVSERVER\_API\_MAJOR, 99
  - LIBVSERVER\_API\_MINOR, 99
  - vs\_get\_config, 100
  - vs\_get\_version, 100
  - vserver, 99
- vx\_bcaps\_get
  - syscall\_context, 21
- vx\_bcaps\_set
  - syscall\_context, 21
- vx\_ccaps\_get
  - syscall\_context, 22
- vx\_ccaps\_set
  - syscall\_context, 22
- vx\_create
  - syscall\_context, 19
- vx\_flags\_get
  - syscall\_context, 23
- vx\_flags\_set
  - syscall\_context, 23
- vx\_flags\_t
  - syscall\_context, 18
- vx\_info
  - syscall\_context, 20
- vx\_info\_t
  - syscall\_context, 18
- vx\_kill
  - syscall\_context, 25
- vx\_limit\_get
  - syscall\_rlimit, 35
- vx\_limit\_mask\_get
  - syscall\_rlimit, 34
- vx\_limit\_reset
  - syscall\_rlimit, 36



- vx\_limit\_set
  - syscall\_rlimit, 34
- vx\_limit\_stat
  - syscall\_rlimit, 35
- vx\_limit\_stat\_t
  - syscall\_rlimit, 34
- vx\_limit\_t
  - syscall\_rlimit, 34
- vx\_migrate
  - syscall\_context, 19
- vx\_sched\_get
  - syscall\_sched, 30
- vx\_sched\_info
  - syscall\_sched, 31
- vx\_sched\_info\_t
  - syscall\_sched, 29
- vx\_sched\_set
  - syscall\_sched, 30
- vx\_sched\_t
  - syscall\_sched, 29
- vx\_stat
  - syscall\_context, 20
- vx\_stat\_t
  - syscall\_context, 18
- vx\_task\_xid
  - syscall\_context, 19
- vx\_uname\_get
  - syscall\_context, 24
- vx\_uname\_set
  - syscall\_context, 24
- vx\_uname\_t
  - syscall\_context, 18
- vx\_wait
  - syscall\_context, 25
- vx\_wait\_t
  - syscall\_context, 18
- VXA\_SOCKET\_INET
  - syscall\_network, 51
- VXA\_SOCKET\_INET6
  - syscall\_network, 51
- VXA\_SOCKET\_OTHER
  - syscall\_network, 51
- VXA\_SOCKET\_PACKET
  - syscall\_network, 51
- VXA\_SOCKET\_UNIX
  - syscall\_network, 50
- VXA\_SOCKET\_UNSPEC
  - syscall\_network, 50
- VXC\_ADMIN\_CLOOP
  - syscall\_context, 14
- VXC\_ADMIN\_MAPPER
  - syscall\_context, 14
- VXC\_BINARY\_MOUNT
  - syscall\_context, 14
- VXC\_QUOTA\_CTL
  - syscall\_context, 14
- VXC\_RAW\_ICMP
  - syscall\_context, 13
- VXC\_SECURE\_MOUNT
  - syscall\_context, 13
- VXC\_SECURE\_REMOUNT
  - syscall\_context, 13
- VXC\_SET\_RLIMIT
  - syscall\_context, 13
- VXC\_SET\_UTSNAME
  - syscall\_context, 13
- VXC\_SYSLOG
  - syscall\_context, 13
- VXF\_FORK\_RSS
  - syscall\_context, 17
- VXF\_HIDE\_MOUNT
  - syscall\_context, 16
- VXF\_HIDE\_NETIF
  - syscall\_context, 16
- VXF\_HIDE\_VINFO
  - syscall\_context, 16
- VXF\_IGNEG\_NICE
  - syscall\_context, 17
- VXF\_INFO\_HIDE
  - syscall\_context, 14
- VXF\_INFO\_INIT
  - syscall\_context, 14
- VXF\_INFO\_NPROC
  - syscall\_context, 14
- VXF\_INFO\_NSPACE
  - syscall\_context, 15
- VXF\_INFO\_PRIVATE
  - syscall\_context, 14
- VXF\_INFO\_SCHED
  - syscall\_context, 14
- VXF\_INFO\_ULIMIT
  - syscall\_context, 15
- VXF\_PERSISTENT
  - syscall\_context, 17
- VXF\_PROLIFIC
  - syscall\_context, 17
- VXF\_REBOOT\_KILL
  - syscall\_context, 16
- VXF\_SC\_HELPER
  - syscall\_context, 16
- VXF\_SCHED\_HARD
  - syscall\_context, 15
- VXF\_SCHED\_PAUSE
  - syscall\_context, 15
- VXF\_SCHED\_PRIO
  - syscall\_context, 15
- VXF\_STATE\_ADMIN
  - syscall\_context, 16

---

VXF\_STATE\_INIT  
    syscall\_context, 16

VXF\_STATE\_SETUP  
    syscall\_context, 16

VXF\_VIRT\_CPU  
    syscall\_context, 15

VXF\_VIRT\_LOAD  
    syscall\_context, 15

VXF\_VIRT\_MEM  
    syscall\_context, 15

VXF\_VIRT\_TIME  
    syscall\_context, 16

VXF\_VIRT\_UPTIME  
    syscall\_context, 15

VXM\_SET\_INIT  
    syscall\_context, 17

VXM\_SET\_REAPER  
    syscall\_context, 17

VXSM\_BUCKET\_ID  
    syscall\_sched, 29

VXSM\_CPU\_ID  
    syscall\_sched, 29

VXSM\_FILL\_RATE  
    syscall\_sched, 28

VXSM\_FILL\_RATE2  
    syscall\_sched, 28

VXSM\_FORCE  
    syscall\_sched, 29

VXSM\_IDLE\_TIME  
    syscall\_sched, 28

VXSM\_INTERVAL  
    syscall\_sched, 28

VXSM\_INTERVAL2  
    syscall\_sched, 28

VXSM\_MSEC  
    syscall\_sched, 29

VXSM\_PRIO\_BIAS  
    syscall\_sched, 28

VXSM\_SET\_MASK  
    syscall\_sched, 29

VXSM\_TOKENS  
    syscall\_sched, 28

VXSM\_TOKENS\_MAX  
    syscall\_sched, 28

VXSM\_TOKENS\_MIN  
    syscall\_sched, 28

VXSM\_V3\_MASK  
    syscall\_sched, 29

xid  
    \_ix\_attr, 61  
    \_vx\_info, 67

xid\_t  
    syscall\_context, 18