

NAME

vsock – Linux VSOCK address family

SYNOPSIS

```
#include <sys/socket.h>
#include <linux/vm_sockets.h>

stream_socket = socket(AF_VSOCK, SOCK_STREAM, 0);
datagram_socket = socket(AF_VSOCK, SOCK_DGRAM, 0);
```

DESCRIPTION

The VSOCK address family facilitates communication between virtual machines and the host they are running on. This address family is used by guest agents and hypervisor services that need a communications channel that is independent of virtual machine network configuration.

Valid socket types are **SOCK_STREAM** and **SOCK_DGRAM**. **SOCK_STREAM** provides connection-oriented byte streams with guaranteed, in-order delivery. **SOCK_DGRAM** provides a connectionless datagram packet service with best-effort delivery and best-effort ordering. Availability of these socket types is dependent on the underlying hypervisor.

A new socket is created with

```
socket(AF_VSOCK, socket_type, 0);
```

When a process wants to establish a connection, it calls **connect(2)** with a given destination socket address. The socket is automatically bound to a free port if unbound.

A process can listen for incoming connections by first binding to a socket address using **bind(2)** and then calling **listen(2)**.

Data is transmitted using the **send(2)** or **write(2)** families of system calls and data is received using the **recv(2)** or **read(2)** families of system calls.

Address format

A socket address is defined as a combination of a 32-bit Context Identifier (CID) and a 32-bit port number. The CID identifies the source or destination, which is either a virtual machine or the host. The port number differentiates between multiple services running on a single machine.

```
struct sockaddr_vm {
    sa_family_t    svm_family;    /* Address family: AF_VSOCK */
    unsigned short svm_reserved1;
    unsigned int    svm_port;      /* Port # in host byte order */
    unsigned int    svm_cid;      /* Address in host byte order */
    unsigned char   svm_zero[sizeof(struct sockaddr) -
                               sizeof(sa_family_t) -
                               sizeof(unsigned short) -
                               sizeof(unsigned int) -
                               sizeof(unsigned int)];
};
```

svm_family is always set to **AF_VSOCK**. *svm_reserved1* is always set to 0. *svm_port* contains the port number in host byte order. The port numbers below 1024 are called *privileged ports*. Only a process with the **CAP_NET_BIND_SERVICE** capability may **bind(2)** to these port numbers. *svm_zero* must be zero-filled.

There are several special addresses: **VMADDR_CID_ANY** (-1U) means any address for binding; **VMADDR_CID_HYPERVISOR** (0) is reserved for services built into the hypervisor; **VMADDR_CID_LOCAL** (1) is the well-known address for local communication (loopback); **VMADDR_CID_HOST** (2) is the well-known address of the host.

The special constant **VMADDR_PORT_ANY** (-1U) means any port number for binding.

Live migration

Sockets are affected by live migration of virtual machines. Connected **SOCK_STREAM** sockets become disconnected when the virtual machine migrates to a new host. Applications must reconnect when this happens.

The local CID may change across live migration if the old CID is not available on the new host. Bound sockets are automatically updated to the new CID.

Ioctls

The following ioctls are available on the `/dev/vsock` device.

IOCTL_VM_SOCKETS_GET_LOCAL_CID

Get the CID of the local machine. The argument is a pointer to an *unsigned int*.

```
ioctl(fd, IOCTL_VM_SOCKETS_GET_LOCAL_CID, &cid);
```

Consider using **VMADDR_CID_ANY** when binding instead of getting the local CID with **IOCTL_VM_SOCKETS_GET_LOCAL_CID**.

Local communication

VMADDR_CID_LOCAL (1) directs packets to the same host that generated them. This is useful for testing applications on a single host and for debugging.

The local CID obtained with **IOCTL_VM_SOCKETS_GET_LOCAL_CID** can be used for the same purpose, but it is preferable to use **VMADDR_CID_LOCAL**.

ERRORS**EACCES**

Unable to bind to a privileged port without the **CAP_NET_BIND_SERVICE** capability.

EADDRINUSE

Unable to bind to a port that is already in use.

EADDRNOTAVAIL

Unable to find a free port for binding or unable to bind to a nonlocal CID.

EINVAL

Invalid parameters. This includes: attempting to bind a socket that is already bound, providing an invalid struct *sockaddr_vm*, and other input validation errors.

ENOPROTOPT

Invalid socket option in **setsockopt(2)** or **getsockopt(2)**.

ENOTCONN

Unable to perform operation on an unconnected socket.

EOPNOTSUPP

Operation not supported. This includes: the **MSG_OOB** flag that is not implemented for the **send(2)** family of syscalls and **MSG_PEEK** for the **recv(2)** family of syscalls.

EPROTONOSUPPORT

Invalid socket protocol number. The protocol should always be 0.

ESOCKTNOSUPPORT

Unsupported socket type in **socket(2)**. Only **SOCK_STREAM** and **SOCK_DGRAM** are valid.

VERSIONS

Support for VMware (VMCI) has been available since Linux 3.9. KVM (virtio) is supported since Linux 4.8. Hyper-V is supported since Linux 4.14.

VMADDR_CID_LOCAL is supported since Linux 5.6. Local communication in the guest and on the host is available since Linux 5.6. Previous versions supported only local communication within a guest (not on the host), and with only some transports (VMCI and virtio).

SEE ALSO

bind(2), connect(2), listen(2), recv(2), send(2), socket(2), capabilities(7)