NAME

ost::UDPSocket −

UDP sockets implement the TCP SOCK_DGRAM UDP protocol.

SYNOPSIS

```
#include <socket.h>
```

Inherits **ost::Socket**.

Inherited by **ost::UDPBroadcast**, **ost::UDPReceive** `[protected]`, and **ost::UDPTransmit** `[protected]`.

Public Member Functions

UDPSocket (**Family family**=IPV4)
Create an unbound UDP socket, mostly for internal use.

UDPSocket (const char *name, **Family family**=IPV4)
Create a UDP socket bound by a service name.

UDPSocket (const **IPV4Address** &bind, **tpport_t** port)
Create a UDP socket and bind it to a specific interface and port address so that other UDP sockets on remote machines (or the same host) may find and send UDP messages to it.

UDPSocket (const **IPV6Address** &bind, **tpport_t** port)

virtual **~UDPSocket** ()
Destroy a UDP socket as a socket.

Error setLoopback (bool enable)
Set the loopback.

Error setMulticast (bool enable)
Set the multicast.

Error setTimeToLive (char **ttl**)
Set time to live.

void **setPeer** (const **IPV4Host** &host, **tpport_t** port)
set the peer address to send message packets to.

void **connect** (const **IPV4Host** &host, **tpport_t** port)

void **setPeer** (const **IPV6Host** &host, **tpport_t** port)

void **connect** (const **IPV6Host** &host, **tpport_t** port)

Socket::Error getInterfaceIndex (const char *ethX, int &InterfaceIndex)
get the interface index for a named network device

Socket::Error join (const **IPV4Multicast** &ia, int InterfaceIndex)
join a multicast group on a particular interface

ssize_t **send** (const void *buf, size_t len)
Send a message packet to a peer host.

ssize_t **receive** (void *buf, size_t len, bool reply=false)
Receive a message from any host.

IPV4Host getIPV4Peer (**tpport_t** *port=NULL) const
Examine address of sender of next waiting packet.

IPV4Host getPeer (**tpport_t** *port=NULL) const

IPV6Host getIPV6Peer (**tpport_t** *port=NULL) const

ssize_t **peek** (void *buf, size_t len)
Examine contents of next waiting packet.

void **setPeer** (const char *service)
Associate socket with a named connection.

void **connect** (const char *service)

Error disconnect (void)
Disassociate this socket from any host connection.

Protected Attributes

union {
  struct sockaddr_in6 **ipv6**
  struct sockaddr_in **ipv4**
} **peer**

Family family

Detailed Description

UDP sockets implement the TCP SOCK_DGRAM UDP protocol.

They can be used to pass unverified messages between hosts, or to broadcast a specific message to an entire subnet. Please note that Streaming of realtime data commonly use **UDPDuplex** related classes rather than **UDPSocket**.

In addition to connected TCP sessions, Common C++ supports UDP sockets and these also cover a range of functionality. Like a **TCPSocket**, A **UDPSocket** can be created bound to a specific network interface and/or port address, though this is not required. UDP sockets also are usually either connected or otherwise 'associated' with a specific 'peer' UDP socket. Since UDP sockets operate through discreet packets, there are no streaming operators used with UDP sockets.

In addition to the UDP 'socket' class, there is a 'UDPBroadcast' class. The **UDPBroadcast** is a socket that is set to send messages to a subnet as a whole rather than to an individual peer socket that it may be associated with.

UDP sockets are often used for building 'realtime' media streaming protocols and full duplex messaging services. When used in this manner, typically a pair of UDP sockets are used together; one socket is used to send and the other to receive data with an associated pair of UDP sockets on a 'peer' host. This concept is represented through the Common C++ **UDPDuplex** object, which is a pair of sockets that communicate with another **UDPDuplex** pair.

Author:

David Sugar <dyfet AT ostel DOT com> Unreliable Datagram Protocol sockets.

Constructor & Destructor Documentation

ost::UDPSocket::UDPSocket (Family family = `IPV4`**)**

Create an unbound UDP socket, mostly for internal use.

ost::UDPSocket::UDPSocket (const char * name, Family family = `IPV4`**)**

Create a UDP socket bound by a service name.

ost::UDPSocket::UDPSocket (const IPV4Address & bind, tpport_t port)

Create a UDP socket and bind it to a specific interface and port address so that other UDP sockets on remote machines (or the same host) may find and send UDP messages to it. On failure to bind, an exception is thrown.

Parameters:

bind address to bind this socket to.
port number to bind this socket to.

ost::UDPSocket::UDPSocket (const IPV6Address & bind, tpport_t port)

virtual ost::UDPSocket::~UDPSocket () `[virtual]`

Destroy a UDP socket as a socket.

Member Function Documentation

void ost::UDPSocket::connect (const char * service)

void ost::UDPSocket::connect (const IPV6Host & host, tpport_t port)

Reimplemented in **ost::UDPReceive**, and **ost::UDPDuplex**.

void ost::UDPSocket::connect (const IPV4Host & host, tpport_t port)

Reimplemented in **ost::UDPTransmit**, **ost::UDPReceive**, and **ost::UDPDuplex**.

Error ost::UDPSocket::disconnect (void)

Disassociate this socket from any host connection. No data should be read or written until a connection is established.

Reimplemented in **ost::UDPDuplex**.

Socket::Error ost::UDPSocket::getInterfaceIndex (const char * ethX, int & InterfaceIndex)

get the interface index for a named network device **Parameters:**

ethX is device name, like 'eth0' or 'eth1'
InterfaceIndex is the index value returned by os

IPV4Host ost::UDPSocket::getIPV4Peer (tpport_t * port = `NULL`**) const**

Examine address of sender of next waiting packet. This also sets 'peer' address to the sender so that the next 'send' message acts as a 'reply'. This additional behavior overides the standard socket getSender

behavior.

Parameters:
port pointer to hold port number.

Reimplemented from **ost::Socket**.

IPV6Host ost::UDPSocket::getIPV6Peer (tpport_t * port = `NULL`**) const**
Reimplemented from **ost::Socket**.

IPV4Host ost::UDPSocket::getPeer (tpport_t * port = `NULL`**) const** `[inline]`
Reimplemented from **ost::Socket**.

Socket::Error ost::UDPSocket::join (const IPV4Multicast & ia, int InterfaceIndex)
join a multicast group on a particular interface **Parameters:**
ia is the multicast address to use
InterfaceIndex is the index value returned by getInterfaceIndex

ssize_t ost::UDPSocket::peek (void * buf, size_t len) `[inline]`
Examine contents of next waiting packet. **Parameters:**
buf pointer to packet buffer for contents.
len of packet buffer.

Returns:
number of bytes examined.

ssize_t ost::UDPSocket::receive (void * buf, size_t len, bool reply = `false`**)**
Receive a message from any host. **Parameters:**
buf pointer to packet buffer to receive.
len of packet buffer to receive.
reply save sender address for reply if true.

Returns:
number of bytes received.

ssize_t ost::UDPSocket::send (const void * buf, size_t len)
Send a message packet to a peer host. **Parameters:**
buf pointer to packet buffer to send.
len of packet buffer to send.

Returns:
number of bytes sent.

Reimplemented in **ost::UDPTransmit**.

Error ost::UDPSocket::setLoopback (bool enable) `[inline]`
Set the loopback.

References ost::Socket::setLoopbackByFamily().

Error ost::UDPSocket::setMulticast (bool enable) `[inline]`
Set the multicast.

Reimplemented in **ost::UDPTransmit**, and **ost::UDPReceive**.

References ost::Socket::setMulticastByFamily().

void ost::UDPSocket::setPeer (const char * service)
Associate socket with a named connection.

void ost::UDPSocket::setPeer (const IPV6Host & host, tpport_t port)
void ost::UDPSocket::setPeer (const IPV4Host & host, tpport_t port)
set the peer address to send message packets to. This can be set before every **send()** call if nessisary.

Parameters:
host address to send packets to.
port number to deliver packets to.

Error ost::UDPSocket::setTimeToLive (char ttl) `[inline]`
Set time to live.

References ost::Socket::setTimeToLiveByFamily().

Member Data Documentation

Family ost::UDPSocket::family [protected]

struct sockaddr_in ost::UDPSocket::ipv4

struct sockaddr_in6 ost::UDPSocket::ipv6

union { ... } ost::UDPSocket::peer [protected]

Author

Generated automatically by Doxygen for GNU CommonC++ from the source code.