NAME

ost::IPV4Address −

The network name and address objects are all derived from a common **IPV4Address** base class.

SYNOPSIS

```
#include <address.h>
```

Inherited by **ost::IPV4Broadcast**, **ost::IPV4Host**, **ost::IPV4Mask**, and **ost::IPV4Multicast**.

Public Member Functions

IPV4Address (const **IPV4Validator** *validator=NULL)
Create an Internet Address object with an empty (0.0.0.0) address.

IPV4Address (struct in_addr addr, const **IPV4Validator** *validator=NULL)
Convert the system internet address data type (struct in_addr) into a Common C++ **IPV4Address** *object.*

IPV4Address (const char *address, const **IPV4Validator** *validator=NULL)
Convert a null terminated ASCII host address string (example: '127.0.0.1') or host address name (example: 'www.voxilla.org') directly into a Common C++ **IPV4Address** *object.*

IPV4Address (const **IPV4Address** &rhs)
Copy constructor.

virtual **~IPV4Address** ()
Destructor.

const char * **getHostname** (void) const
Provide a string representation of the value (Internet Address) held in the **IPV4Address** *object.*

bool **isInetAddress** (void) const
May be used to verify if a given **IPV4Address** *returned by another function contains a 'valid' address, or '0.0.0.0' which is often used to mark 'invalid'* **IPV4Address** *values.*

struct in_addr **getAddress** (void) const
Provide a low level system usable struct in_addr object from the contents of **IPV4Address**.

struct in_addr **getAddress** (size_t i) const
Provide a low level system usable struct in_addr object from the contents of **IPV4Address**.

size_t **getAddressCount** () const
Returns the number of internet addresses that an **IPV4Address** *object contains.*

IPV4Address & **operator=** (const char *str)

IPV4Address & **operator=** (struct in_addr addr)

IPV4Address & **operator=** (const **IPV4Address** &rhs)

IPV4Address & **operator=** (unsigned long addr)
Allows assignment from the return of functions like inet_addr() or htonl().

IPV4Address & **operator=** (unsigned int addr)

bool **operator!** () const

bool **operator==** (const **IPV4Address** &a) const
Compare two internet addresses to see if they are equal (if they specify the physical address of the same internet host).

bool **operator!=** (const **IPV4Address** &a) const
Compare two internet addresses to see if they are not equal (if they each refer to unique and different physical ip addresses).

Protected Member Functions

bool **setIPAddress** (const char *host)
Sets the IP address from a string representation of the numeric address, ie '127.0.0.1'.

void **setAddress** (const char *host)
Used to specify a host name or numeric internet address.

Protected Attributes

struct in_addr * **ipaddr**

size_t **addr_count**

char * **hostname**

Static Protected Attributes

static **Mutex mutex**

Detailed Description

The network name and address objects are all derived from a common **IPV4Address** base class.

Specific classes, such as **IPV4Host**, **IPV4Mask**, etc, are defined from **IPV4Address** entirely so that the manner a network address is being used can easily be documented and understood from the code and to avoid common errors and accidental misuse of the wrong address object. For example, a 'connection' to something that is declared as a 'IPV4Host' can be kept type-safe from a 'connection' accidently being made to something that was declared a 'IPV4Broadcast'.

Author:

David Sugar <dyfet AT ostel DOT com> Internet Address binary data type.

Examples:

tcpthread.cpp.

Constructor & Destructor Documentation

ost::IPV4Address::IPV4Address (const IPV4Validator * validator = `NULL`**)**

Create an Internet Address object with an empty (0.0.0.0) address. **Parameters:**

validator optional validator function object, intended for derived classes.

ost::IPV4Address::IPV4Address (struct in_addr addr, const IPV4Validator * validator = `NULL`**)**

Convert the system internet address data type (struct in_addr) into a Common C++ **IPV4Address** object. **Parameters:**

addr struct of system used binary internet address.
validator optional validator function object, intended for derived classes.

ost::IPV4Address::IPV4Address (const char * address, const IPV4Validator * validator = `NULL`**)**

Convert a null terminated ASCII host address string (example: '127.0.0.1') or host address name (example: 'www.voxilla.org') directly into a Common C++ **IPV4Address** object. **Parameters:**

address null terminated C string.
validator optional validator function object, intended for derived classes.

ost::IPV4Address::IPV4Address (const IPV4Address & rhs)

Copy constructor.

virtual ost::IPV4Address::~IPV4Address () `[virtual]`

Destructor.

Member Function Documentation

struct in_addr ost::IPV4Address::getAddress (size_t i) const `[read]`

Provide a low level system usable struct in_addr object from the contents of **IPV4Address**. This is needed for services such as bind() and connect().

Parameters:

i for IPV4Addresses with multiple addresses, returns the address at this index. User should call **getAddressCount()** to determine the number of address the object contains.

Returns:

system binary coded internet address. If parameter i is out of range, the first address is returned.

struct in_addr ost::IPV4Address::getAddress (void) const `[read]`

Provide a low level system usable struct in_addr object from the contents of **IPV4Address**. This is needed for services such as bind() and connect().

Returns:

system binary coded internet address.

size_t ost::IPV4Address::getAddressCount () const `[inline]`

Returns the number of internet addresses that an **IPV4Address** object contains. This usually only happens with **IPV4Host** objects where multiple IP addresses are returned for a DNS lookup

const char* ost::IPV4Address::getHostname (void) const

Provide a string representation of the value (Internet Address) held in the **IPV4Address** object.

Returns:

string representation of **IPV4Address**.

Examples:

tcpthread.cpp.

bool ost::IPV4Address::isInetAddress (void) const

May be used to verify if a given **IPV4Address** returned by another function contains a 'valid' address, or '0.0.0.0' which is often used to mark 'invalid' **IPV4Address** values. **Returns:**

true if address != 0.0.0.0.

bool ost::IPV4Address::operator! () const `[inline]`

bool ost::IPV4Address::operator!= (const IPV4Address & a) const

Compare two internet addresses to see if they are not equal (if they each refer to unique and different physical ip addresses). This is implimented in terms of operator==

IPV4Address& ost::IPV4Address::operator= (unsigned int addr) `[inline]`

IPV4Address& ost::IPV4Address::operator= (unsigned long addr)

Allows assignment from the return of functions like inet_addr() or htonl().

Reimplemented in **ost::IPV4Mask**, and **ost::IPV4Host**.

IPV4Address& ost::IPV4Address::operator= (const IPV4Address & rhs)

IPV4Address& ost::IPV4Address::operator= (struct in_addr addr)

IPV4Address& ost::IPV4Address::operator= (const char * str)

Referenced by ost::IPV4Host::operator=(), and ost::IPV4Mask::operator=().

bool ost::IPV4Address::operator== (const IPV4Address & a) const

Compare two internet addresses to see if they are equal (if they specify the physical address of the same internet host). If there is more than one IP address in either **IPV4Address** object, this will return true if all of the IP addresses in the smaller are in the larger in any order.

void ost::IPV4Address::setAddress (const char * host) `[protected]`

Used to specify a host name or numeric internet address. **Parameters:**

host The string representation of the IP address or a hostname, , if NULL, it will default to INADDR_ANY

bool ost::IPV4Address::setIPAddress (const char * host) `[protected]`

Sets the IP address from a string representation of the numeric address, ie '127.0.0.1'. **Parameters:**

host The string representation of the IP address

Returns:

true if successful

Member Data Documentation

size_t ost::IPV4Address::addr_count `[protected]`

char* ost::IPV4Address::hostname `[mutable, protected]`

struct in_addr* ost::IPV4Address::ipaddr `[protected]`

Mutex ost::IPV4Address::mutex `[static, protected]`

Author

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