n2nc

0.1

Generated by Doxygen 1.5.5

Mon Oct 6 19:32:15 2008

Contents

Chapter 1

Nat 2 Nat Connector Project

1.1 Introduction

The n2nc project aims to connects 2 host behind differents nats gateway together using standard streams or vdeplug library.

The architecture is server-client on which the client works in 2 modes: active and passive, the active client want to connect to the passive endpoint and the passive client expects a connection request by active client. Each client(passive and active) first registers itself to the server by sending a couple of data: the ID of the client (that simply is a substring of md5sum of the own rsa public key) and a port number on which the client binds a socket for transport comunication.

Once both clients are registered, the active client send a message to the passive client trough the server which act a message bouncer, this message tell the wish of connect; the passive peer receives the message and then ask the server which service port has registered the active client. At this point each client sends to the other garbage UDP datagram asincroniously, this way each NAT gateway binds a mapping so that the connection can forward in both senses from/to each client. this method is called UDP Hole Punching. when each client sees the datagram arrives, the active client generates a key of 128 bit lenght to identify the session and send it to their other party in a secured rsa message. the ability of sending secured message are guaranted by the RSA cipher family where each client ID associates a unique rsa public key. secured message are encrypted using the public key of the other party.

The n2nc has the so called filter module extension, which work in this way:

the service data(datagrams) is processed in a chain of filters that one a time does datagram inspection end/or edit, at this time i write this document, are implemented 4 filters: dummy filter that does nothing, LZO filter that act as LZO link compression, BZ2 link compressor, and the last, BlowFish filter cipher module which uses the previously geretated and shared session key.

1.2 Compiling

Simply execute:

./configure

make

make install #(optional)

There are a couple of dependencies to satisfy. n2nc needs the libssl header (libssl-dev on debian) and bzip2

headers (libbz2-dev).

1.3 Testing

On each side(clients passive/active) create a file in ∼/.n2nc/loadfilters and put the filename of filter you want to load on each line:

e.g.

cat ∼/.n2nc/loadfilters

libbz2filter

libbffilter

in this case, libz2filter will act as link compressor, then the compressed data will be processed by blowfish crypto module. straighforwardly the data's coming path will traverses these filters in reverse order.

Now it's time to generate a rsa key pair with the script "n2nc_newrsa.sh", both public and private keys will be generated to keyring directory (∼/.n2nc/keyring), the created keys should have a filename starting with a 4 digit hexadecimal number, this is the own peer's ID. Now share the public key between the 2 clients through an arbitrary secure way you mind.

1.3.1 On the server side:

#./server -p 5555 -bindip 0.0.0.0

1.3.2 On the passive client:

#./n2nc -p -saddr n2ncservice.org -bindip 192.168.0.55 -myid 0x245e -filterdir filters/.libs/

1.3.3 On the active client:

#./n2nc -saddr n2ncservice.org -bindip 192.168.1.4 -myid 0x0ad9 -otherid 0x245e -filterdir filters/.libs/

-saddr is the server hostname or ip

-bindip is the IP to bind for udp service comunication (should be the address for which there is the default gateway)

-myid is my ID (first 4 digit of my rsa publick key's filename)

-otherid the ID of the other party

-filterdir is the direcrory where plugins modules are located

1.3.4 VDEPLUG BUILTIN MODE

Simply add the "-vdesock path.to.vde_switch.ctl.dir" paramaeter and n2nc will connect to that switch.

e.g.:

1.3.5 On the passive client:

#./n2nc -p -saddr n2ncservice.org -bindip 192.168.0.55 -myid 0x245e -filterdir filters/.libs/ -vdesock /tmp/vde.ctl/

1.3.6 On the active client:

#./n2nc -saddr n2ncservice.org -bindip 192.168.1.4 -myid 0x0ad9 -otherid 0x245e -filterdir filters/.libs/ -vdesock /tmp/vde.ctl/

PLease use the sourceforge system to track bugs/wishes and so on.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 n2nc Namespace Reference

Default project namespace.

Classes

- class ClientInfo
- class ConfigurationOpt

  Class to provides configuration option pair.

- class ConfigurationCtx

  Class to provides configuration context. A context is ConfigurationOpt container.

- class Configuration

  Class to provides configuration handler and storage. This class is a ConfigurationCtx container.

- class Credential

  This class provides the storage for information about peer/user credential.

- class Direct_Traversal

  This class implements direct traversal(dummy).

- class Filter

  Base class for plugin-based traffic filter.

- class FilterAdapter

  This class provides the filter chain to be applied to a Socket.

- class Filter_dummy

  Dummy transport filter which does NOTHING.

- class FilterBlowFish
- class FilterBZ2
- class FilterLZO

- struct **myconf_t**
- class Network

  This class represents a network.

- class NetworkChannel

  This class represents the comunication channel between 2 networks.

- class PacketManager
- class Relay_Traversal

  This class provides the implementation of tracker's relay/bouncer.

- class Server
- class TrackerClient

  This class proviedes the code to connect and exchange commands to and from a TrackerServer.

- class TrackerServer

  This class provides the server tracker implementation.

- class Traversal

  This class provides the base interface to transport NAT traversal procedures.

- class UdpHolePunching_Traversal

  This class provides Udp Hole Punching NAT traversal method.

- class Upnp_Traversal

  This class provides the UPNP implementation.

Namespaces

- namespace net

  Networking related classes.

- namespace sync

  Synchronization related classes.

- namespace system

  System related classes.

Functions

- Filter * **get_istance** ()
- int **free_istance** (Filter *filter)
- int **stdinoutAttach** (net::Socket *srv_sock, FilterAdapter &fa)
- int **vdeAttach** (net::Socket *srv_sock, FilterAdapter &fa)
- int **loadArgsHelper** (int argc, char **argv)
- int **passivemode** (net::Socket *ctl_sock, net::Socket &srv_sock, FilterAdapter &fa)
- int **activemode** (net::Socket *ctl_sock, net::Socket &srv_sock, FilterAdapter &fa)

- int **punching** (net::Socket &srv_sock)
- Server::packet * **sendMSG** (net::Socket *ctl_sock, Server::server_command cmd)
- Server::packet * **awaitMSG** (net::Socket *ctl_sock, Server::server_command cmd, Server::packet *recvpk)

Variables

- struct myconf_t **myconf**

5.1.1 Detailed Description

Default project namespace.

n2nc namespace description.

5.2 n2nc::net Namespace Reference

Networking related classes.

Classes

- class Address

  Provides the abstraction to implements various address family addresses.

- class Host
- class IP4Address

  Provides the IP version 4 address abstraction.

- class IP6Address

  Provides the IP version 6 address abstraction.

- class RawSocket

  Raw socket implementation.

- class Resolver

  This class provides various utilities to help about dns name resolver.

- class Socket

  The socket abstaction class.

- class SocketAddress

  socket address base class (similar to sockaddr_storage)

- class SocketEvents

  Socket events to avoid explicits callbacks.

- class SocketEventsHandler

  Provides asynchronus socket managment. this class provides a method to handle all events generated by Socket. Basically it starts a new thread which poll on any Socket by using select(), it generates a callback event defined in SocketEvents interface. This way avoid the plenty use of select() and move programmer to a event-based socket management.

- class TcpSocket

  Tcp Socket implementation.

- class UdpSocket

  Udp Socket implementation.

- class UnixAddress

  Unix Address family(AF_UNIX) abstaction.

5.2.1 Detailed Description

Networking related classes.

n2nc::net namespace description.

5.3 n2nc::sync Namespace Reference

Synchronization related classes.

Classes

- class Condition

  Conditional variable implementation.

- class Mutex

  Mutex implementation.

- class Semaphore

  This class implements the semaphore paradigm.

- class Thread

  Base Threads interface.

- class MainThread

  Workaround to emulate the main thread. Do not use this.

- class InlineThread

  C-Style threading model.

5.3.1 Detailed Description

Synchronization related classes.

n2nc::sync namespace description.

5.4 n2nc::system Namespace Reference

System related classes.

Classes

- class Logger

  Provides a thead-safe and reentrant logging facility.

Functions

- void **operator**<< (std::ostream &os, Logger::endl_flush_t dummy)
- void **operator**<< (std::ostream &os, Logger::flush_t dummy)

5.4.1 Detailed Description

System related classes.

n2nc::system namespace description.

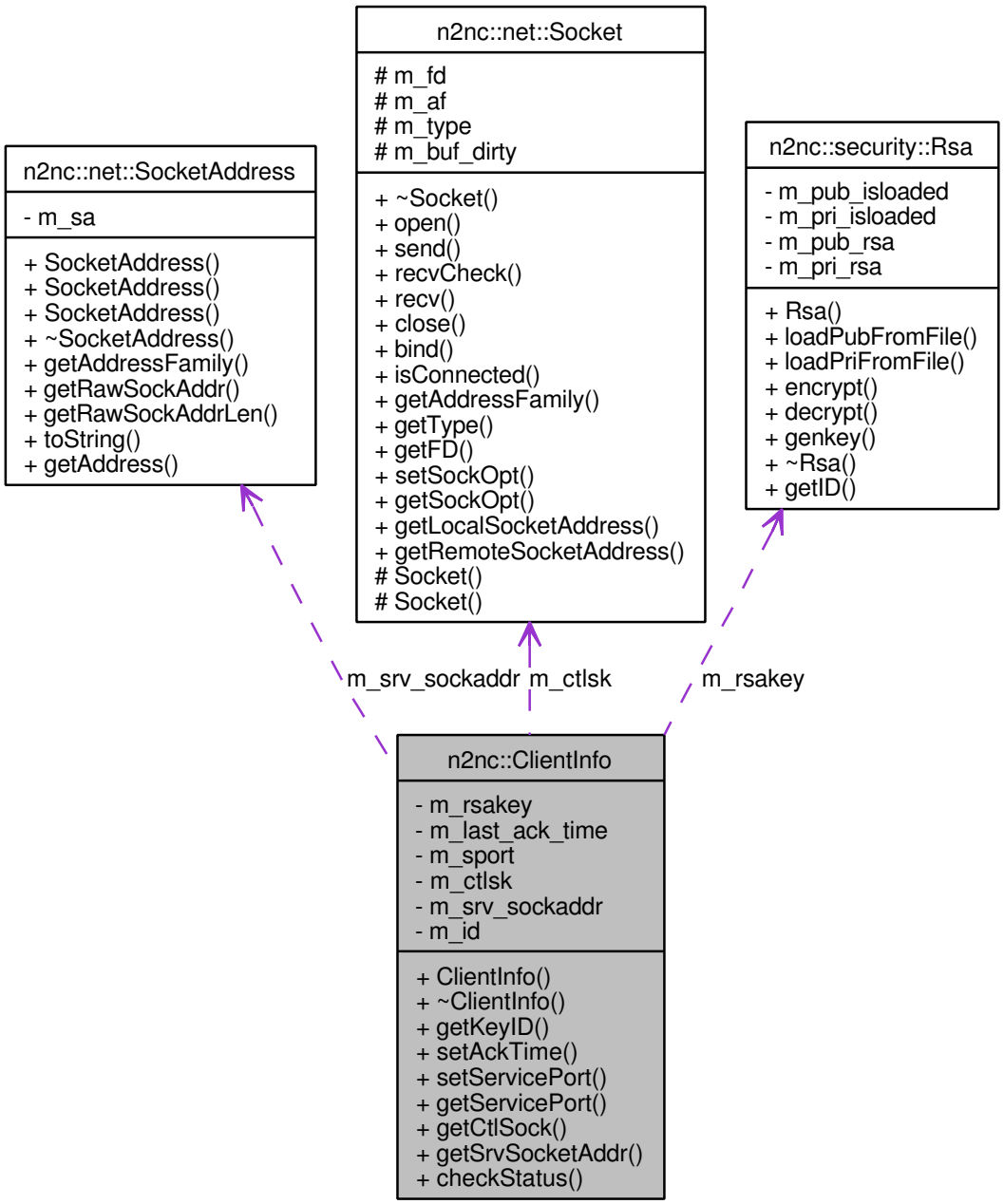
Chapter 6

Class Documentation

6.1 n2nc::ClientInfo Class Reference

`#include <clientinfo.h>`

Collaboration diagram for n2nc::ClientInfo:



Public Types

- typedef u_int32_t **ClientID**

Public Member Functions

- **ClientInfo** (ClientID id, net::Socket *sk)
- ClientID **getKeyID** ()

- time_t setAckTime ()
- int **setServicePort** (uint16_t port)
- uint16_t **getServicePort** ()
- net::Socket * **getCtlSock** ()
- net::SocketAddress * **getSrvSocketAddr** ()
- bool checkStatus ()

6.1.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 19 of file clientinfo.h.

6.1.2 Member Function Documentation

6.1.2.1 time_t n2nc::ClientInfo::setAckTime ()

Refresh the ack timestamp

Definition at line 22 of file clientinfo.cpp.

6.1.2.2 bool n2nc::ClientInfo::checkStatus ()

Returns:

true if the client is currently online, false if the client timeout is expired

Definition at line 26 of file clientinfo.cpp.

The documentation for this class was generated from the following files:

- clientinfo.h
- clientinfo.cpp

6.2 n2nc::Configuration Class Reference

Class to provides configuration handler and storage. This class is a ConfigurationCtx container.

`#include <configuration.h>`

Public Member Functions

- std::string toString ()
- bool fromString (std::string &str)
- bool addContext (const ConfigurationCtx &ctx)
- bool delContext (ConfigurationCtx &ctx)
- ConfigurationCtx & getCtx (std::string ctxname)
- ConfigurationCtx & operator[ ] (std::string ctxname)

6.2.1 Detailed Description

Class to provides configuration handler and storage. This class is a ConfigurationCtx container.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

configuration_ex.cpp.

Definition at line 69 of file configuration.h.

6.2.2 Member Function Documentation

6.2.2.1 std::string n2nc::Configuration::toString ()

To string.

Returns:

A string containing all contexts and their options

6.2.2.2 bool n2nc::Configuration::fromString (std::string & *str*)

Loads configuration from string

6.2.2.3 bool n2nc::Configuration::addContext (const ConfigurationCtx & *ctx*)

Adds a Context

Examples:

configuration_ex.cpp.

Definition at line 54 of file configuration.cpp.

6.2.2.4 bool n2nc::Configuration::delContext (ConfigurationCtx & *ctx*)

Deletes a Context

6.2.2.5 ConfigurationCtx& n2nc::Configuration::getCtx (std::string *ctxname*)

Returns:

The Context by its name

6.2.2.6 ConfigurationCtx & n2nc::Configuration::operator[ ] (std::string *ctxname*)

Returns:

The Context by its name.

Definition at line 25 of file configuration.cpp.

The documentation for this class was generated from the following files:

- configuration.h
- configuration.cpp

6.3 n2nc::ConfigurationCtx Class Reference

Class to provides configuration context. A context is ConfigurationOpt container.

`#include <configuration.h>`

Public Member Functions

- **ConfigurationCtx** (std::string name)
- std::string toString ()
- bool addOption (const ConfigurationOpt &opt)
- bool delOption (ConfigurationOpt &opt)
- ConfigurationOpt & getOpt (std::string optname)
- ConfigurationOpt & operator[ ] (std::string optname)

6.3.1 Detailed Description

Class to provides configuration context. A context is ConfigurationOpt container.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

configuration_ex.cpp.

Definition at line 43 of file configuration.h.

6.3.2 Member Function Documentation

6.3.2.1 std::string n2nc::ConfigurationCtx::toString ()

To string

6.3.2.2 bool n2nc::ConfigurationCtx::addOption (const ConfigurationOpt & *opt*)

Adds an Option in this context

Definition at line 31 of file configuration.cpp.

6.3.2.3 bool n2nc::ConfigurationCtx::delOption (ConfigurationOpt & *opt*)

Deletes an Option in this context

6.3.2.4 ConfigurationOpt& n2nc::ConfigurationCtx::getOpt (std::string *optname*)

Returns:

the option value associated to optname

6.3.2.5 ConfigurationOpt& n2nc::ConfigurationCtx::operator[ ] (std::string *optname*)

Returns:

the option value associated to optname

The documentation for this class was generated from the following files:

- configuration.h
- configuration.cpp

6.4 n2nc::ConfigurationOpt Class Reference

Class to provides configuration option pair.

`#include <configuration.h>`

Public Member Functions

- ConfigurationOpt (std::string first, std::string second)
- ConfigurationOpt (const ConfigurationOpt &opt)
- std::string **toString** ()
- std::string **value** ()
- void setValue (std::string value)
- void operator= (std::string value)

6.4.1 Detailed Description

Class to provides configuration option pair.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

configuration_ex.cpp.

Definition at line 12 of file configuration.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 n2nc::ConfigurationOpt::ConfigurationOpt (std::string *first*, std::string *second*)

Parameters:

first The option name.

second The option value.

Definition at line 7 of file configuration.cpp.

6.4.2.2 n2nc::ConfigurationOpt::ConfigurationOpt (const ConfigurationOpt & *opt*)

Copy constructor

Definition at line 11 of file configuration.cpp.

References m_first, and m_second.

6.4.3 Member Function Documentation

6.4.3.1 void n2nc::ConfigurationOpt::setValue (std::string *value*)

sets the associated value for this Config

Definition at line 17 of file configuration.cpp.

6.4.3.2 void n2nc::ConfigurationOpt::operator= (std::string *value*)

sets the associated value for this Config. eg: ConfigurationOpt opt("key","value"); opt = "newvalue" ;

The documentation for this class was generated from the following files:

- configuration.h
- configuration.cpp

6.5 n2nc::Credential Class Reference

This class provides the storage for information about peer/user credential.

`#include <credential.h>`

6.5.1 Detailed Description

This class provides the storage for information about peer/user credential.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 12 of file credential.h.

The documentation for this class was generated from the following files:

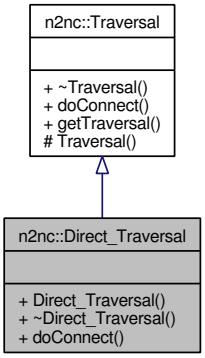
- credential.h
- credential.cpp

6.6 n2nc::Direct_Traversal Class Reference

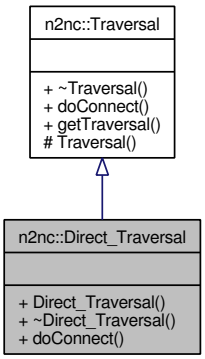
This class implements direct traversal(dummy).

`#include <direct_traversal.h>`

Inheritance diagram for n2nc::Direct_Traversal:



Collaboration diagram for n2nc::Direct_Traversal:



Public Member Functions

- virtual net::Socket & doConnect (Network &net)

6.6.1 Detailed Description

This class implements direct traversal(dummy).

This dummy traversal method aim to be used whereas one peer has a public access service, in other words has its own public IP or forwarded port to it through the nat service.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 16 of file direct_traversal.h.

6.6.2 Member Function Documentation

6.6.2.1 net::Socket & n2nc::Direct_Traversal::doConnect (Network & *net*) [virtual]

Performs a connection to Network *net* FIXME if error ?

Implements n2nc::Traversal.

Definition at line 14 of file direct_traversal.cpp.

The documentation for this class was generated from the following files:

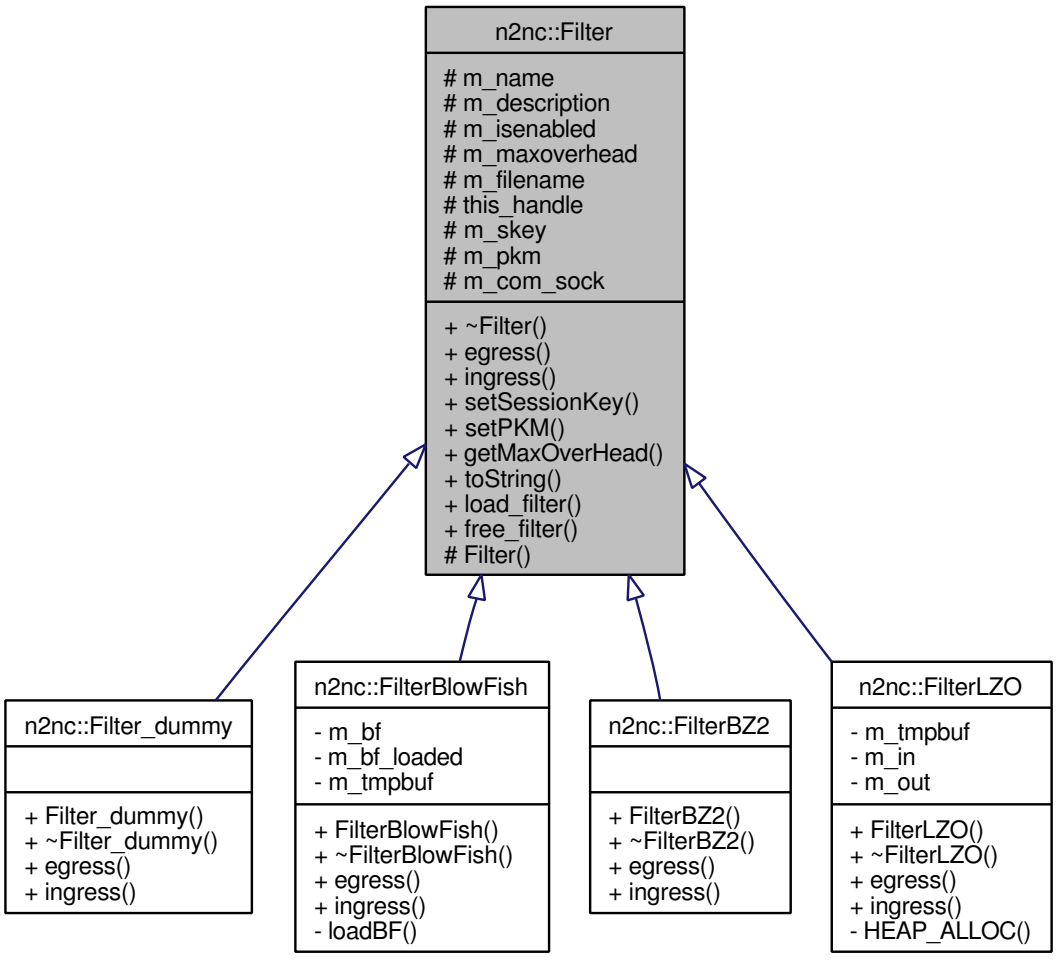
- direct_traversal.h
- direct_traversal.cpp

6.7 n2nc::Filter Class Reference

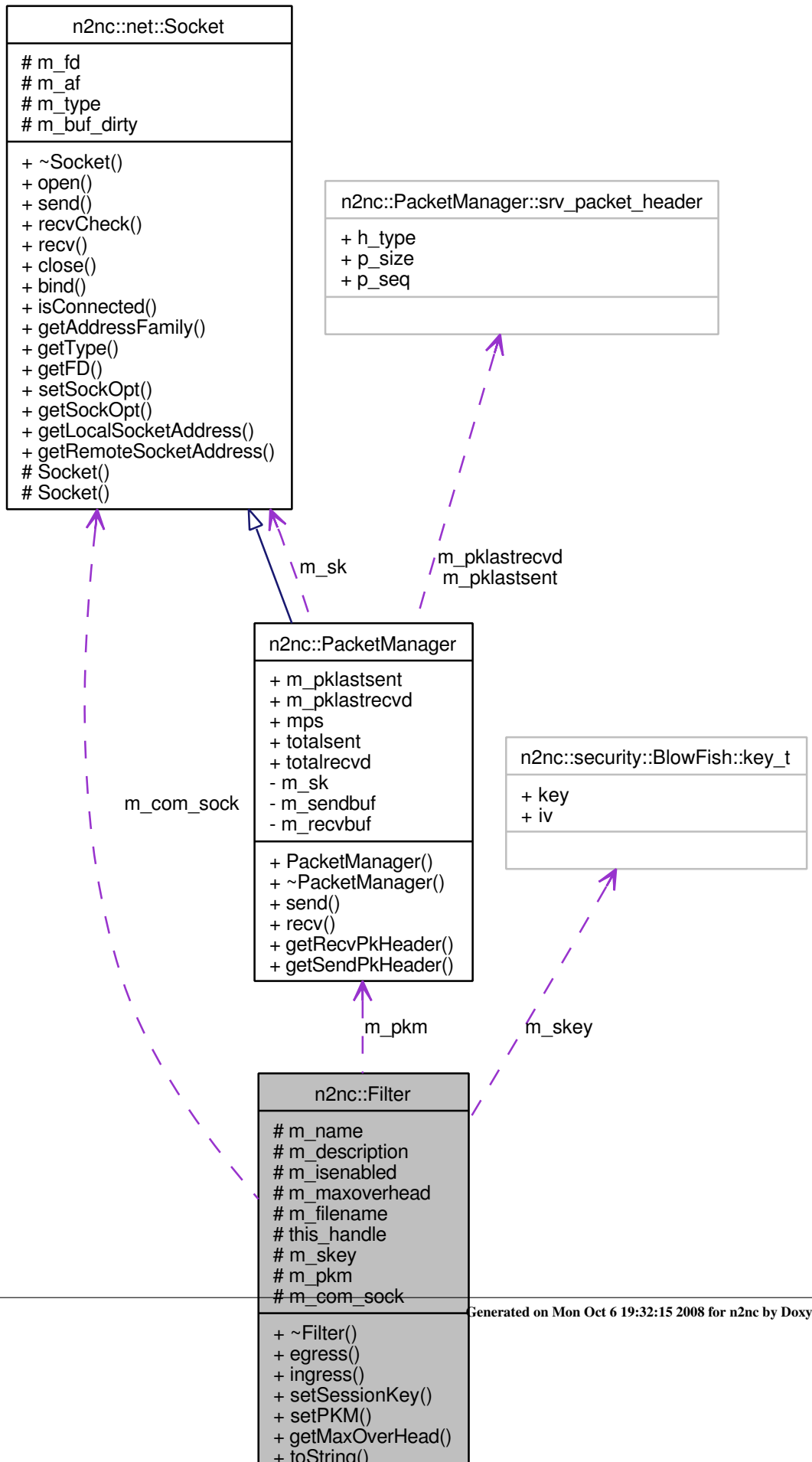
Base class for plugin-based traffic filter.

`#include <filter.h>`

Inheritance diagram for n2nc::Filter:



Collaboration diagram for n2nc::Filter:



Public Types

- enum status_t { FILTER_CONTINUE = 0, FILTER_RETURN, FILTER_DROP }

Public Member Functions

- virtual status_t egress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)=0
- virtual status_t ingress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)=0
- virtual int setSessionKey (security::BlowFish::key_t *key)
- virtual int setPKM (n2nc::PacketManager *pkm)
- virtual size_t **getMaxOverHead** ()
- std::string toString ()

Static Public Member Functions

- static Filter * load_filter (const std::string filename)
- static int free_filter (Filter *filter)

Protected Member Functions

- Filter (std::string name, std::string description)

Protected Attributes

- std::string **m_name**
- std::string **m_description**
- bool **m_isenabled**
- size_t **m_maxoverhead**
- std::string **m_filename**
- void * **this_handle**
- security::BlowFish::key_t * **m_skey**
- n2nc::PacketManager * **m_pkm**
- n2nc::net::Socket * **m_com_sock**

6.7.1 Detailed Description

Base class for plugin-based traffic filter.

Author:

fabsoft <fabsoft@gmail.com>

Note:

Plugin has to define these factory/cleaner with

```
extern "C"
static Filter* get_istance(void *argv,int argc);
extern "C"
static int free_istance();
```

Examples:

filter_ex.cpp.

Definition at line 28 of file filter.h.

6.7.2 Member Enumeration Documentation

6.7.2.1 enum n2nc::Filter::status_t

Defines the packet behaviour in the chain

Enumerator:

FILTER_CONTINUE The packet follows the filter chain

FILTER_RETURN The packet exit the filter chain

FILTER_DROP The packet is discarded

Definition at line 59 of file filter.h.

6.7.3 Constructor & Destructor Documentation

6.7.3.1 n2nc::Filter::Filter (std::string *name*, std::string *description*) [inline, protected]

Each filter has to set its name and description by constructor call.

Note:

Use Filter::load_filter to creates a new Filter

Definition at line 48 of file filter.h.

6.7.4 Member Function Documentation

6.7.4.1 virtual status_t n2nc::Filter::egress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [pure virtual]

Outgoing traffic goes through egress hook.

Implemented in n2nc::Filter_dummy, n2nc::FilterBlowFish, n2nc::FilterBZ2, and n2nc::FilterLZO.

Examples:

filter_ex.cpp.

Referenced by n2nc::FilterAdapter::send().

Here is the caller graph for this function:

6.7.4.2 virtual status_t n2nc::Filter::ingress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [pure virtual]

Ingoing traffic goes through ingress hook

Implemented in n2nc::Filter_dummy, n2nc::FilterBlowFish, n2nc::FilterBZ2, and n2nc::FilterLZO.

Referenced by n2nc::FilterAdapter::recv().

Here is the caller graph for this function:

6.7.4.3 virtual int n2nc::Filter::setSessionKey (security::BlowFish::key_t * *key*) [inline, virtual]

Sets the session key. usually 128 bit

Definition at line 80 of file filter.h.

6.7.4.4 virtual int n2nc::Filter::setPKM (n2nc::PacketManager * *pkm*) [inline, virtual]

Set the packet manager to the filter, used in certain circumstances

Definition at line 82 of file filter.h.

6.7.4.5 std::string n2nc::Filter::toString () [inline]

Prints the filter name and description..

Examples:

filter_ex.cpp.

Definition at line 85 of file filter.h.

6.7.4.6 static Filter* n2nc::Filter::load_filter (const std::string *filename*) [inline, static]

Public factory method to istanciates a new Filter module by *filename*

Definition at line 94 of file filter.h.

References m_filename, and this_handle.

6.7.4.7 static int n2nc::Filter::free_filter (Filter * *filter*) [inline, static]

Public denstructor method.

Note:

The library will be unloaded only if there are no symbols linked against it.

Definition at line 112 of file filter.h.

References this_handle.

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

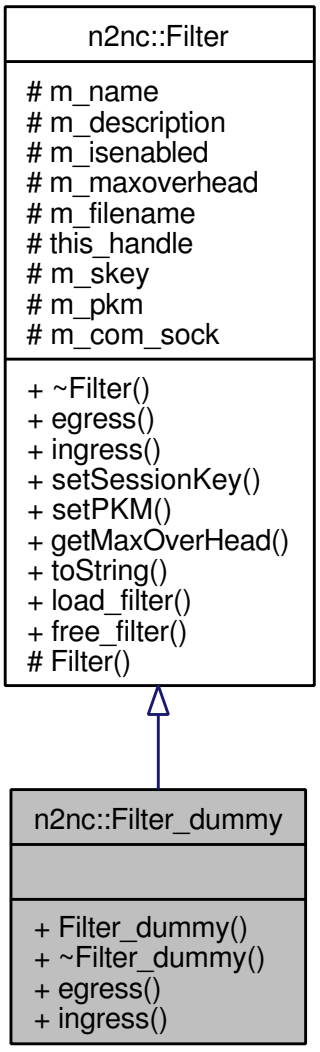
- filter.h

6.8 n2nc::Filter_dummy Class Reference

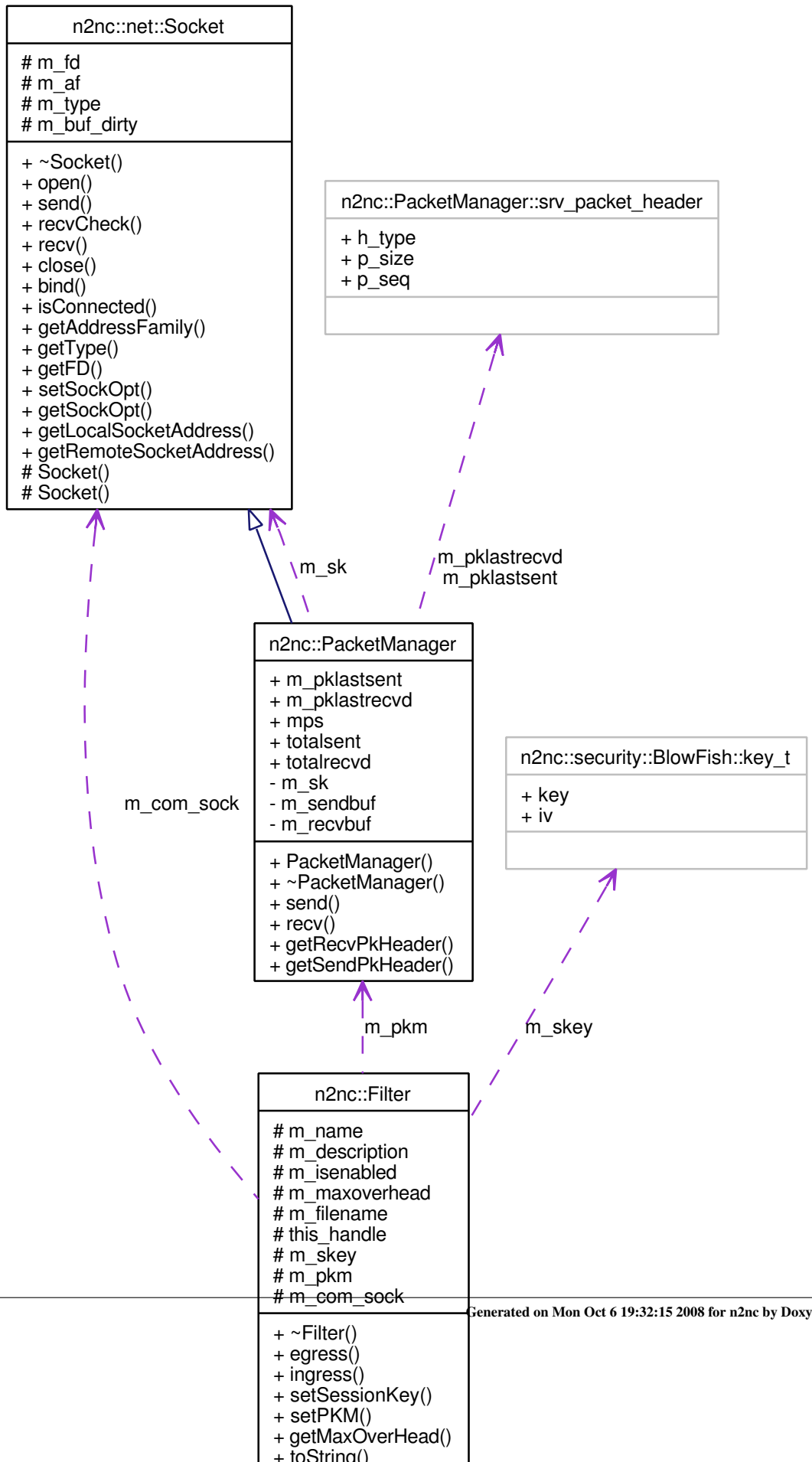
Dummy transport filter which does NOTHING.

`#include <filter_dummy.h>`

Inheritance diagram for n2nc::Filter_dummy:



Collaboration diagram for n2nc::Filter_dummy:



Public Member Functions

- virtual status_t egress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)
- virtual status_t ingress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)

6.8.1 Detailed Description

Dummy transport filter which does NOTHING.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 12 of file filter_dummy.h.

6.8.2 Member Function Documentation

6.8.2.1 Filter::status_t n2nc::Filter_dummy::egress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [virtual]

Outgoing traffic goes through egress hook.

Implements n2nc::Filter.

Definition at line 15 of file filter_dummy.cpp.

References n2nc::Filter::FILTER_CONTINUE.

6.8.2.2 Filter::status_t n2nc::Filter_dummy::ingress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [virtual]

Ingoing traffic goes through ingress hook

Implements n2nc::Filter.

Definition at line 22 of file filter_dummy.cpp.

References n2nc::Filter::FILTER_CONTINUE.

The documentation for this class was generated from the following files:

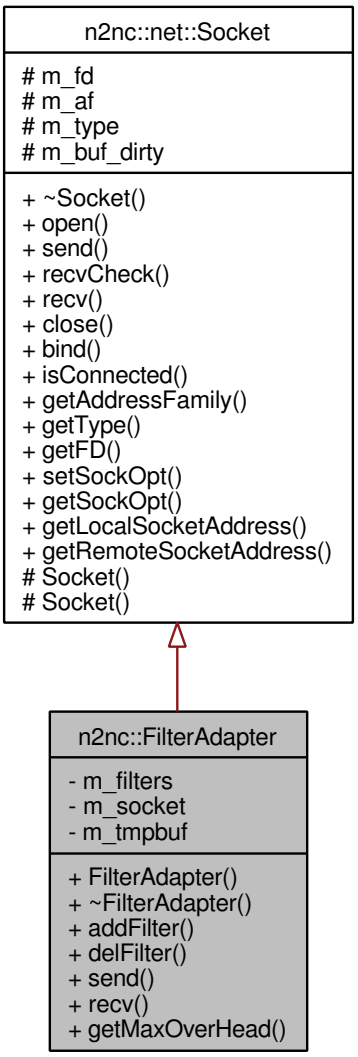
- filter_dummy.h
- filter_dummy.cpp

6.9 n2nc::FilterAdapter Class Reference

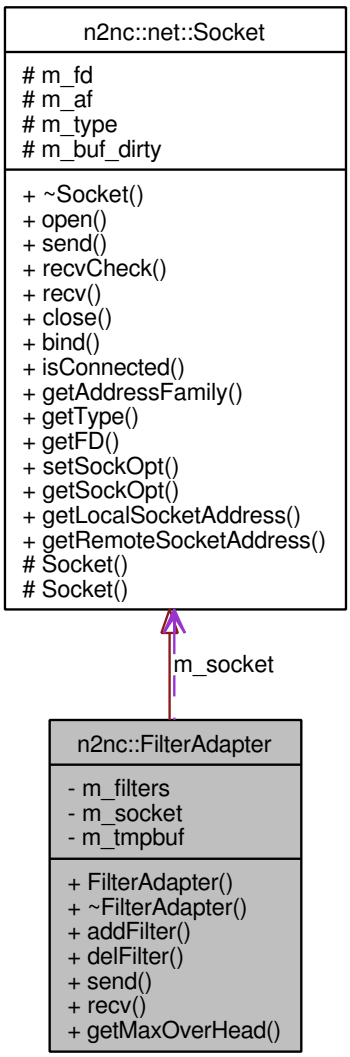
This class provides the filter chain to be applied to a Socket.

`#include <filteradapter.h>`

Inheritance diagram for n2nc::FilterAdapter:



Collaboration diagram for n2nc::FilterAdapter:



Public Member Functions

- **FilterAdapter** (net::Socket &socket)
- bool **addFilter** (Filter ∗filter)
- bool **delFilter** (Filter ∗filter)
- virtual int send (void ∗buf, size_t len)
- virtual int recv (void ∗buf, size_t len)
- size_t **getMaxOverHead** ()

6.9.1 Detailed Description

This class provides the filter chain to be applied to a Socket.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 14 of file filteradapter.h.

6.9.2 Member Function Documentation

6.9.2.1 int n2nc::FilterAdapter::send (void * *buf*, size_t *len*) [virtual]

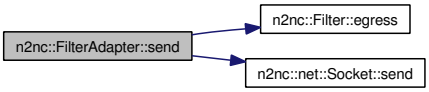
Send data buf to remote peer

Reimplemented from n2nc::net::Socket.

Definition at line 27 of file filteradapter.cpp.

References n2nc::Filter::egress(), n2nc::Filter::FILTER_CONTINUE, n2nc::Filter::FILTER_DROP, n2nc::Filter::FILTER_RETURN, and n2nc::net::Socket::send().

Here is the call graph for this function:



6.9.2.2 int n2nc::FilterAdapter::recv (void * *buf*, size_t *len*) [virtual]

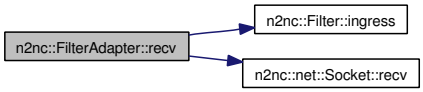
Receive len leght data from remote peer and store it to buf

Reimplemented from n2nc::net::Socket.

Definition at line 70 of file filteradapter.cpp.

References n2nc::Filter::FILTER_CONTINUE, n2nc::Filter::FILTER_DROP, n2nc::Filter::FILTER_RETURN, n2nc::Filter::ingress(), and n2nc::net::Socket::recv().

Here is the call graph for this function:



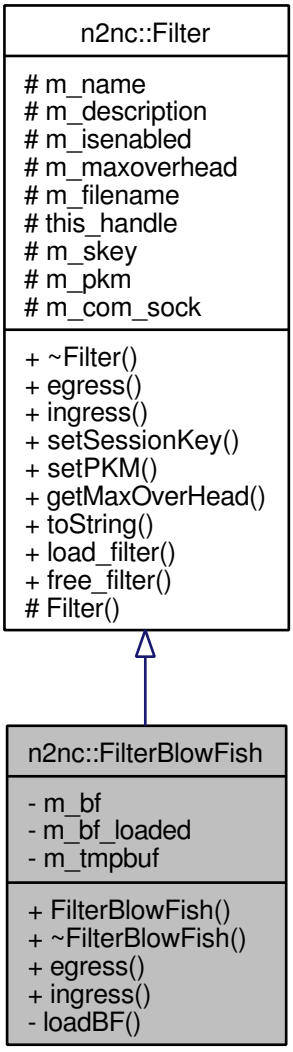
The documentation for this class was generated from the following files:

- filteradapter.h
- filteradapter.cpp

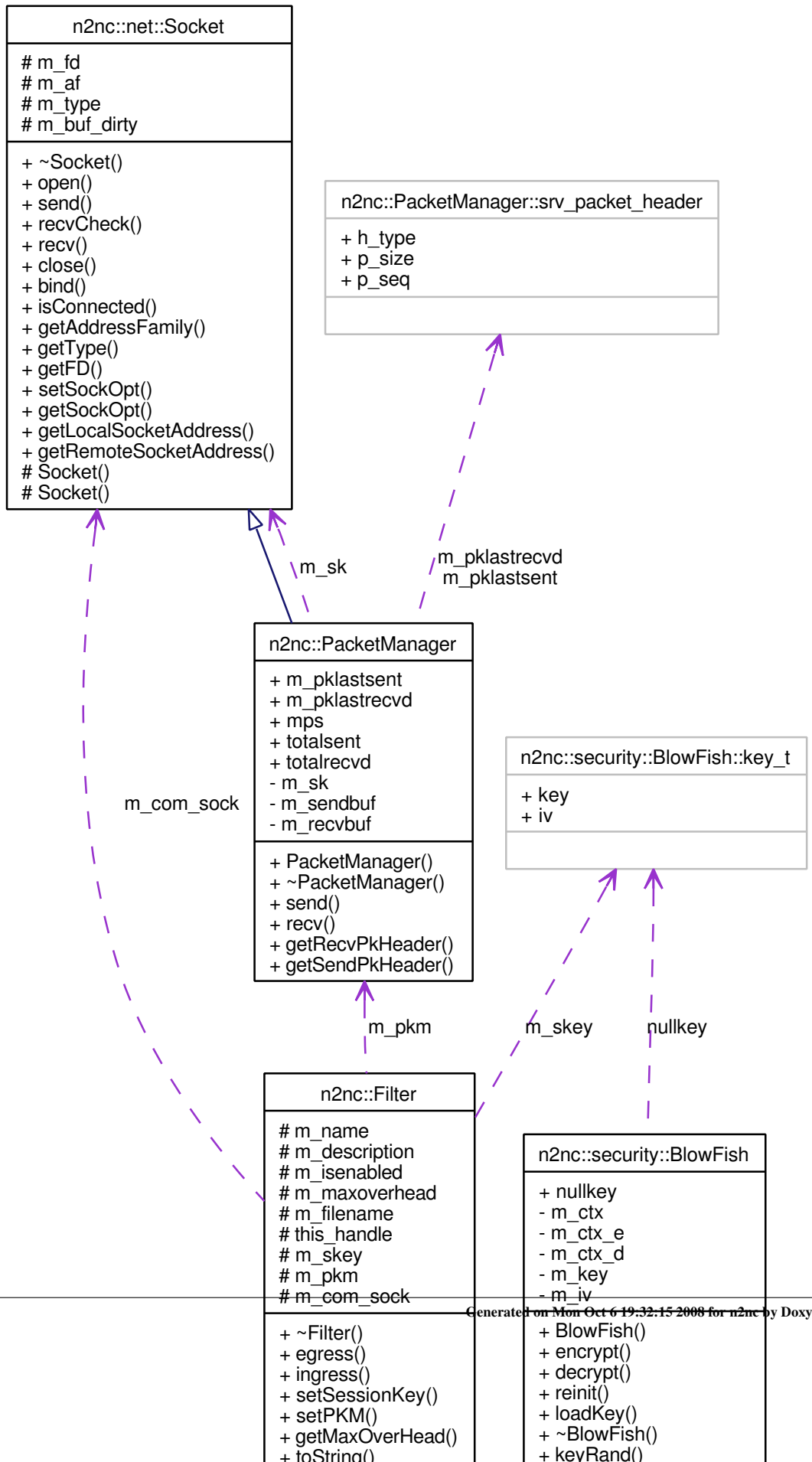
6.10 n2nc::FilterBlowFish Class Reference

`#include <filterblowfish.h>`

Inheritance diagram for n2nc::FilterBlowFish:



Collaboration diagram for n2nc::FilterBlowFish:



Public Member Functions

- virtual Filter::status_t egress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)
- virtual Filter::status_t ingress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)

Classes

- struct **bfheader_t**

6.10.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 15 of file filterblowfish.h.

6.10.2 Member Function Documentation

6.10.2.1 Filter::status_t n2nc::FilterBlowFish::egress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [virtual]

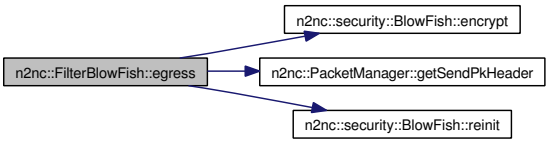
Outgoing traffic goes through egress hook.

Implements n2nc::Filter.

Definition at line 22 of file filterblowfish.cpp.

References n2nc::security::BlowFish::encrypt(), n2nc::Filter::FILTER_CONTINUE, n2nc::Filter::FILTER_DROP, n2nc::PacketManager::getSendPkHeader(), and n2nc::security::BlowFish::reinit().

Here is the call graph for this function:



6.10.2.2 Filter::status_t n2nc::FilterBlowFish::ingress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [virtual]

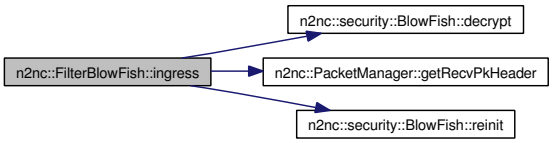
Ingoing traffic goes through ingress hook

Implements n2nc::Filter.

Definition at line 48 of file filterblowfish.cpp.

References n2nc::security::BlowFish::decrypt(), n2nc::Filter::FILTER_CONTINUE, n2nc::Filter::FILTER_DROP, n2nc::PacketManager::getRecvPkHeader(), and n2nc::security::BlowFish::reinit().

Here is the call graph for this function:



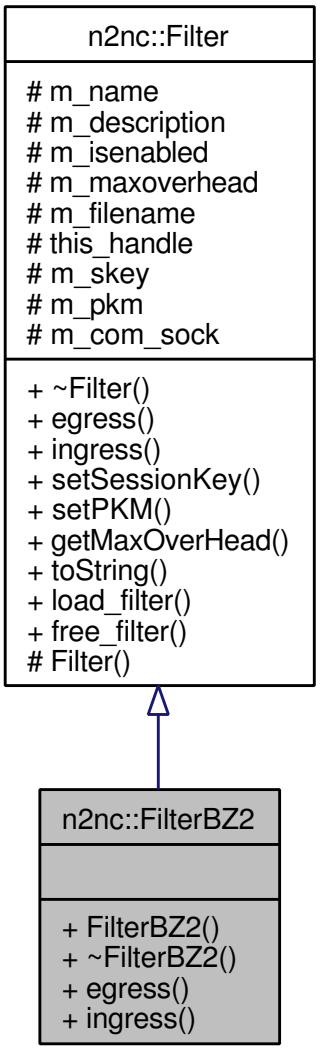
The documentation for this class was generated from the following files:

- filterblowfish.h
- filterblowfish.cpp

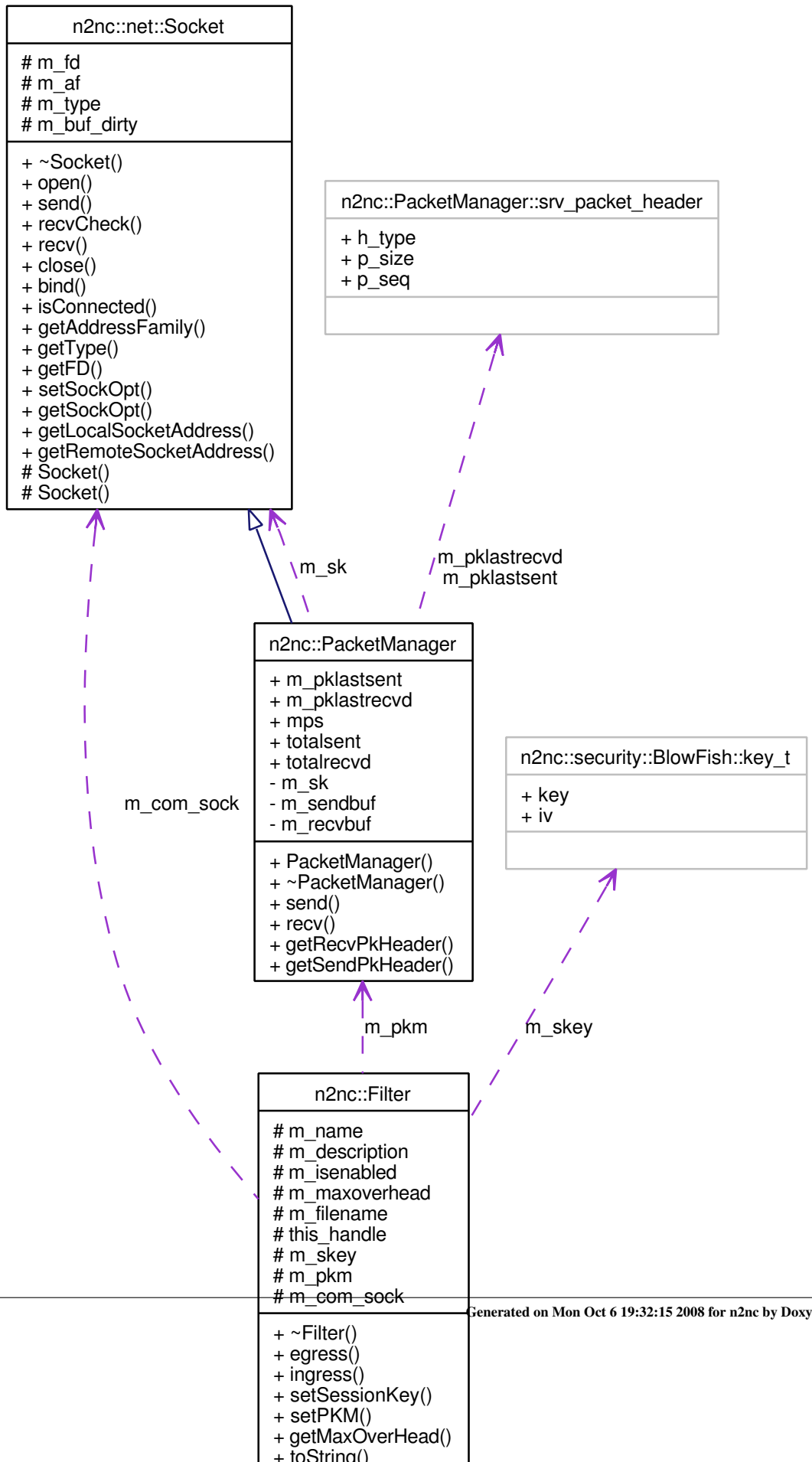
6.11 n2nc::FilterBZ2 Class Reference

`#include <filterbz2.h>`

Inheritance diagram for n2nc::FilterBZ2:



Collaboration diagram for n2nc::FilterBZ2:



Public Member Functions

- virtual status_t egress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)
- virtual status_t ingress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)

Classes

- struct **bz2header_t**

6.11.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 13 of file filterbz2.h.

6.11.2 Member Function Documentation

6.11.2.1 Filter::status_t n2nc::FilterBZ2::egress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [virtual]

Outgoing traffic goes through egress hook.

Implements n2nc::Filter.

Definition at line 13 of file filterbz2.cpp.

References n2nc::Filter::FILTER_CONTINUE.

6.11.2.2 Filter::status_t n2nc::FilterBZ2::ingress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [virtual]

Ingoing traffic goes through ingress hook

Implements n2nc::Filter.

Definition at line 35 of file filterbz2.cpp.

References n2nc::Filter::FILTER_CONTINUE, and n2nc::Filter::FILTER_DROP.

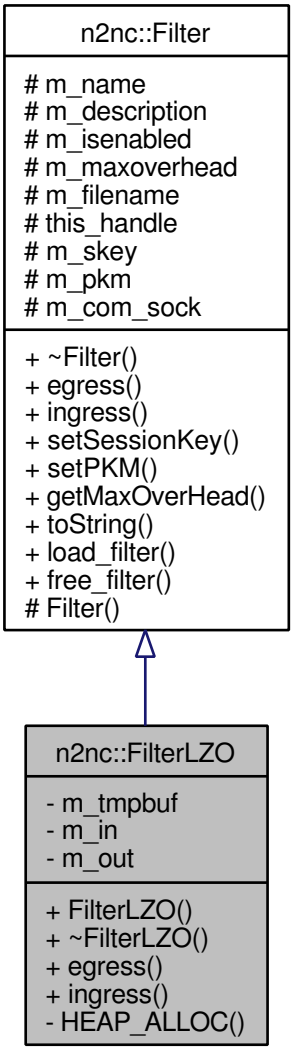
The documentation for this class was generated from the following files:

- filterbz2.h
- filterbz2.cpp

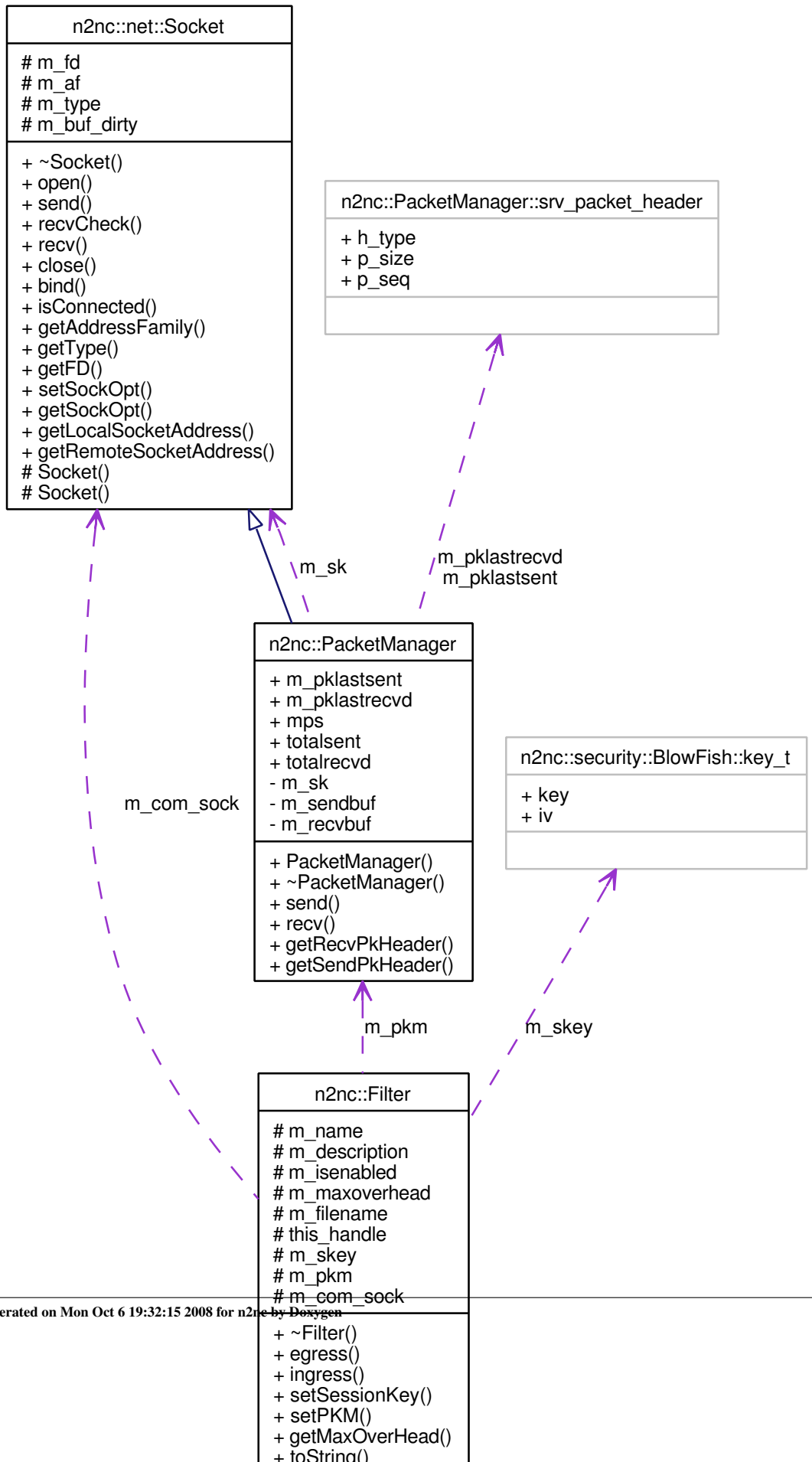
6.12 n2nc::FilterLZO Class Reference

`#include <filterlzo.h>`

Inheritance diagram for n2nc::FilterLZO:



Collaboration diagram for n2nc::FilterLZO:



Public Member Functions

- virtual Filter::status_t egress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)
- virtual Filter::status_t ingress (void *inbuf, void *outbuf, size_t inlen, size_t *outlen)

Classes

- struct **lzoheader_t**

6.12.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 12 of file filterlzo.h.

6.12.2 Member Function Documentation

6.12.2.1 Filter::status_t n2nc::FilterLZO::egress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [virtual]

Outgoing traffic goes through egress hook.

Implements n2nc::Filter.

Definition at line 20 of file filterlzo.cpp.

References n2nc::Filter::FILTER_CONTINUE.

6.12.2.2 Filter::status_t n2nc::FilterLZO::ingress (void * *inbuf*, void * *outbuf*, size_t *inlen*, size_t * *outlen*) [virtual]

Ingoing traffic goes through ingress hook

Implements n2nc::Filter.

Definition at line 44 of file filterlzo.cpp.

References n2nc::Filter::FILTER_CONTINUE.

The documentation for this class was generated from the following files:

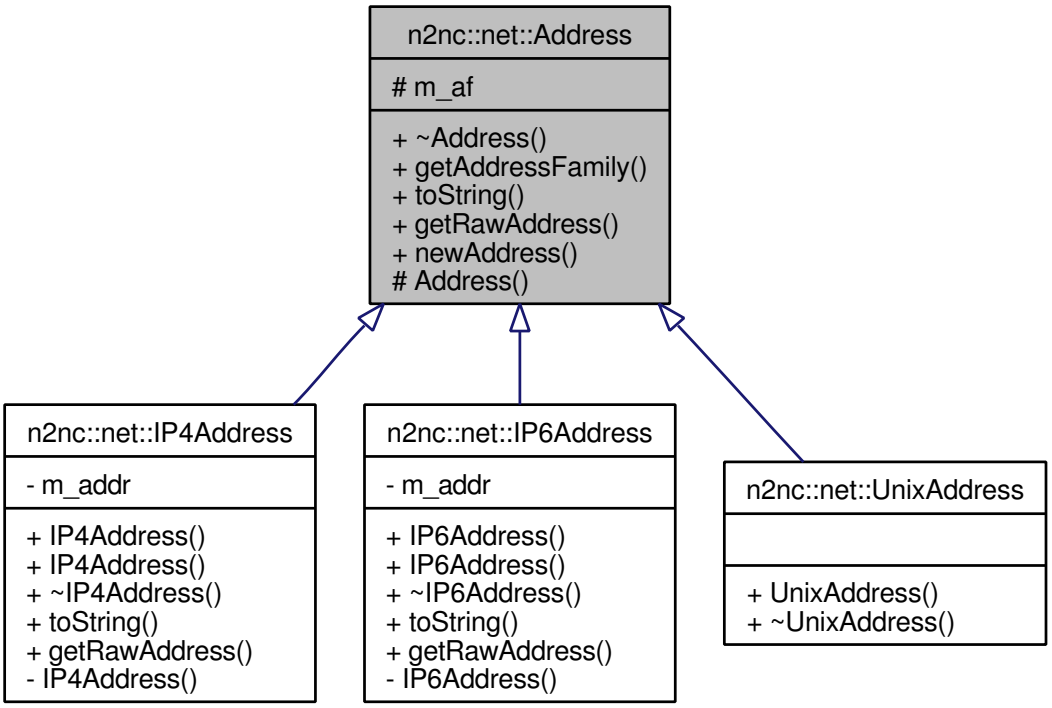
- filterlzo.h
- filterlzo.cpp

6.13 n2nc::net::Address Class Reference

Provides the abstraction to implements various address family addresses.

`#include <address.h>`

Inheritance diagram for n2nc::net::Address:



Public Member Functions

- int getAddressFamily () const
- virtual std::string toString () const =0
- virtual void getRawAddress (void **addr, size_t *len)=0

Static Public Member Functions

- static Address * newAddress (std::string &addr)

  (Factory method)

Protected Attributes

- int **m_af**

6.13.1 Detailed Description

Provides the abstraction to implements various address family addresses.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 18 of file address.h.

6.13.2 Member Function Documentation

6.13.2.1 Address * n2nc::net::Address::newAddress (std::string & *addr*) [static]

(Factory method)

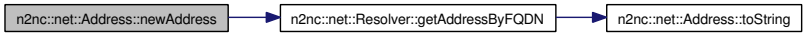
Returns:

A new Address by specified *addr* in string format

Definition at line 20 of file address.cpp.

References n2nc::net::Resolver::getAddressByFQDN().

Here is the call graph for this function:



6.13.2.2 int n2nc::net::Address::getAddressFamily () const

Returns:

the address family in OS defined format

Definition at line 16 of file address.cpp.

6.13.2.3 virtual std::string n2nc::net::Address::toString () const [pure virtual]

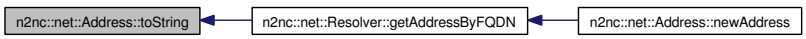
Returns:

the address in string format

Implemented in n2nc::net::IP4Address, and n2nc::net::IP6Address.

Referenced by n2nc::net::Resolver::getAddressByFQDN().

Here is the caller graph for this function:



6.13.2.4 virtual void n2nc::net::Address::getRawAddress (void ** *addr*, size_t * *len*) [pure virtual]

Returns:

a pointer to the internal sockaddr* and sets its lenght

Implemented in n2nc::net::IP4Address, and n2nc::net::IP6Address.

The documentation for this class was generated from the following files:

- address.h
- address.cpp

6.14 n2nc::net::Host Class Reference

`#include <host.h>`

Public Member Functions

- **Host** (std::string &)

6.14.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 14 of file host.h.

The documentation for this class was generated from the following files:

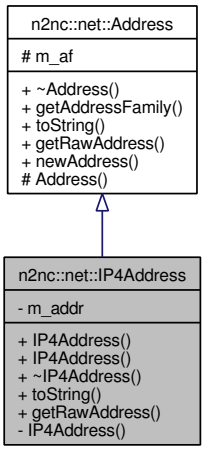
- host.h
- host.cpp

6.15 n2nc::net::IP4Address Class Reference

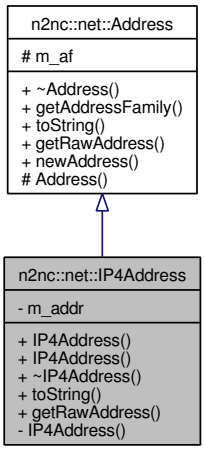
Provides the IP version 4 address abstraction.

`#include <ip4address.h>`

Inheritance diagram for n2nc::net::IP4Address:



Collaboration diagram for n2nc::net::IP4Address:



Public Member Functions

- **IP4Address** (std::string &addr)
- **IP4Address** (struct in_addr *addr)
- virtual std::string toString () const
- virtual void getRawAddress (void **addr, size_t *len)

Friends

- class **Resolver**

6.15.1 Detailed Description

Provides the IP version 4 address abstraction.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 14 of file ip4address.h.

6.15.2 Member Function Documentation

6.15.2.1 std::string n2nc::net::IP4Address::toString () const [virtual]

Returns:

the address in string format

Implements n2nc::net::Address.

Definition at line 36 of file ip4address.cpp.

6.15.2.2 void n2nc::net::IP4Address::getRawAddress (void ** *addr*, size_t * *len*) [virtual]

Returns:

a pointer to the internal sockaddr* and sets its lenght

Implements n2nc::net::Address.

Definition at line 48 of file ip4address.cpp.

The documentation for this class was generated from the following files:

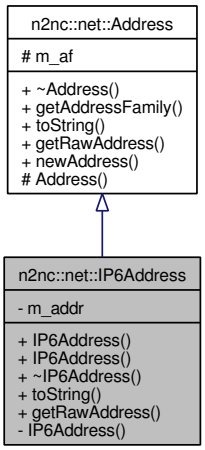
- ip4address.h
- ip4address.cpp

6.16 n2nc::net::IP6Address Class Reference

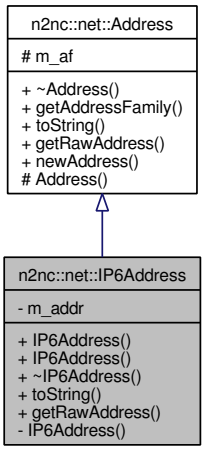
Provides the IP version 6 address abstraction.

`#include <ip6address.h>`

Inheritance diagram for n2nc::net::IP6Address:



Collaboration diagram for n2nc::net::IP6Address:



Public Member Functions

- **IP6Address** (std::string &addr)
- **IP6Address** (struct in6_addr *addr)
- virtual std::string toString () const
- virtual void getRawAddress (void **addr, size_t *len)

Friends

- class **Resolver**

6.16.1 Detailed Description

Provides the IP version 6 address abstraction.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 15 of file ip6address.h.

6.16.2 Member Function Documentation

6.16.2.1 std::string n2nc::net::IP6Address::toString () const [virtual]

Returns:

the address in string format

Implements n2nc::net::Address.

Definition at line 33 of file ip6address.cpp.

6.16.2.2 void n2nc::net::IP6Address::getRawAddress (void ** *addr*, size_t * *len*) [virtual]

Returns:

a pointer to the internal sockaddr* and sets its lenght

Implements n2nc::net::Address.

Definition at line 45 of file ip6address.cpp.

The documentation for this class was generated from the following files:

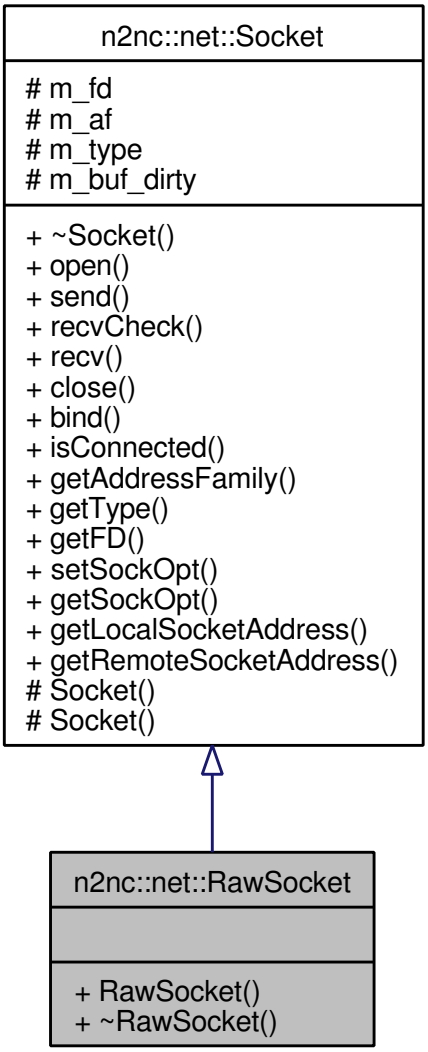
- ip6address.h
- ip6address.cpp

6.17 n2nc::net::RawSocket Class Reference

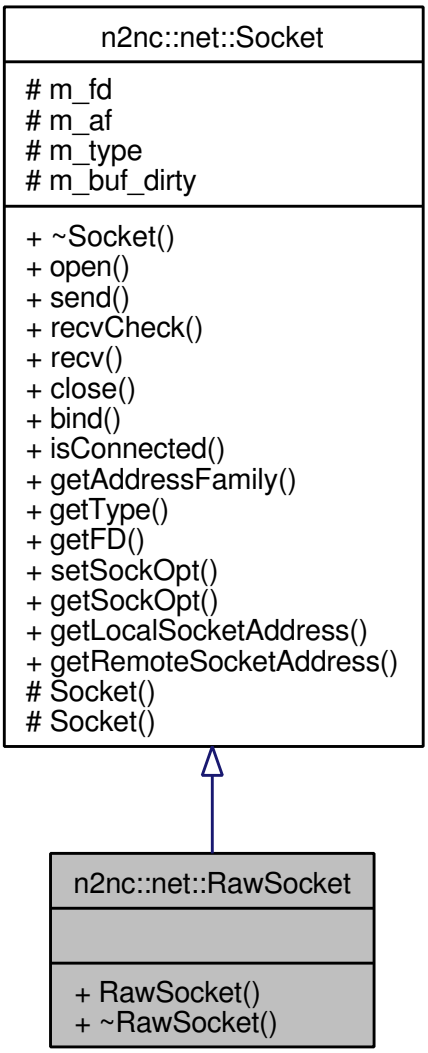
Raw socket implementation.

`#include <rawsocket.h>`

Inheritance diagram for n2nc::net::RawSocket:



Collaboration diagram for n2nc::net::RawSocket:



Public Member Functions

- **RawSocket** (int af)

6.17.1 Detailed Description

Raw socket implementation.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 14 of file rawsocket.h.

The documentation for this class was generated from the following files:

- rawsocket.h
- rawsocket.cpp

6.18 n2nc::net::Resolver Class Reference

This class provides various utilities to help about dns name resolver.

`#include <resolver.h>`

Static Public Member Functions

- static Address * getAddressByFQDN (std::string &addr, bool resolve=true)
- static SocketAddress * getSocketAddressByService (const std::string &fqdn, const std::string &port, int type=SOCK_STREAM, int af=AF_UNSPEC, bool resolve=false)
- static SocketAddress * **getSocketAddressByService** (const std::string &fqdn, const int port, int type=SOCK_STREAM, int af=AF_UNSPEC, bool resolve=false)
- static SocketAddress * **getSocketAddressByService** (const Address &addr, const int port, int type=SOCK_STREAM)
- static std::string & **getFQDNbyAddress** (Address &addr)
- static Address & **getAddressByName** (std::string &name, bool resolve=false)

6.18.1 Detailed Description

This class provides various utilities to help about dns name resolver.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 18 of file resolver.h.

6.18.2 Member Function Documentation

6.18.2.1 Address * n2nc::net::Resolver::getAddressByFQDN (std::string & *addr*, bool *resolve* = true) [static]

a new Address

Definition at line 12 of file resolver.cpp.

References n2nc::net::Address::toString().

Referenced by n2nc::net::Address::newAddress().

Here is the call graph for this function:

Here is the caller graph for this function:

6.18.2.2 SocketAddress * n2nc::net::Resolver::getSocketAddressByService (const std::string & *fqdn*, const std::string & *port*, int *type* = SOCK_STREAM, int *af* = AF_UNSPEC, bool *resolve* = false) [static]

FIXME address family can be detected by address

Definition at line 93 of file resolver.cpp.

Referenced by ThreadTest::entry_point(), and n2nc::Server::listen().

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

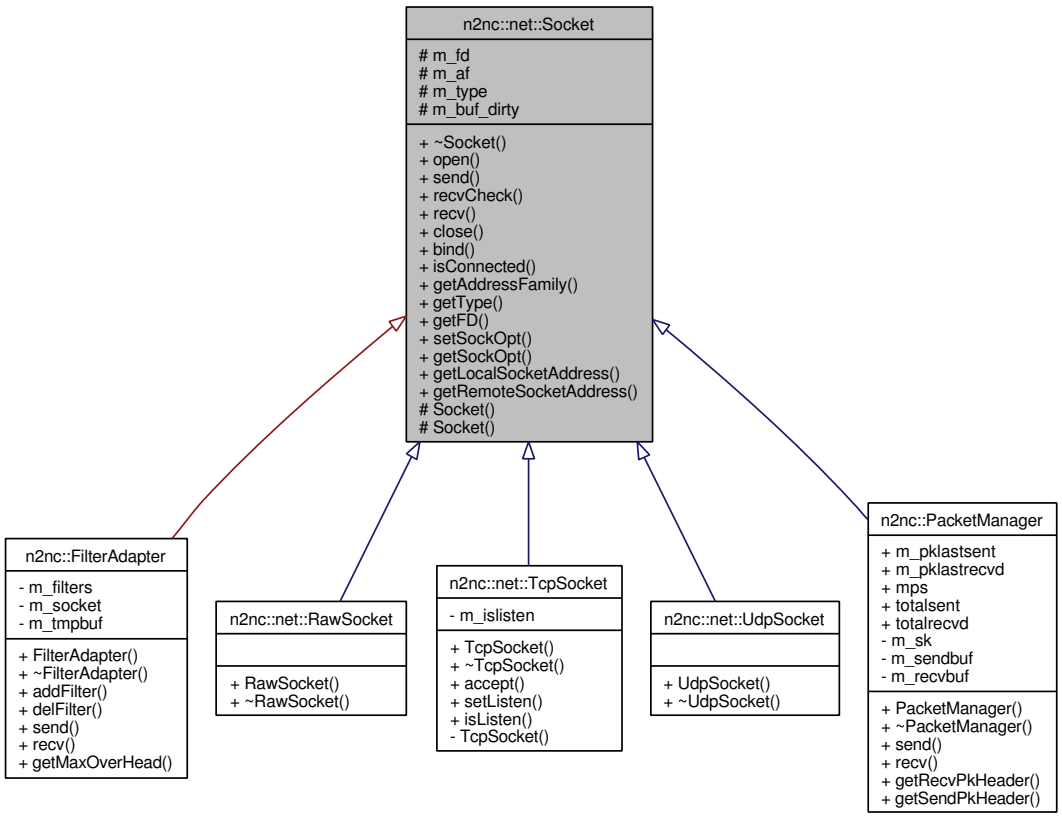
- resolver.h
- resolver.cpp

6.19 n2nc::net::Socket Class Reference

The socket abstaction class.

`#include <socket.h>`

Inheritance diagram for n2nc::net::Socket:



Public Types

- typedef int **fd_t**

Public Member Functions

- virtual int open (SocketAddress &addr)
- virtual int send (void *buf, size_t len)
- virtual size_t recvCheck ()
- virtual int recv (void *buf, size_t len)
- virtual int close ()
- virtual int bind (SocketAddress &addr)
- virtual bool isConnected ()
- int getAddressFamily ()
- int getType ()
- fd_t getFD () const

- int setSockOpt (int optname, bool value, int level=SOL_SOCKET)
- bool getSockOpt (int optname, int level=SOL_SOCKET)
- SocketAddress * getLocalSocketAddress () const
- SocketAddress * getRemoteSocketAddress () const

Protected Member Functions

- Socket (int af, int type)

Protected Attributes

- int m_fd
- int **m_af**
- int **m_type**
- void * **m_buf_dirty**

6.19.1 Detailed Description

The socket abstaction class.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

socket_ex.cpp.

Definition at line 16 of file socket.h.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 n2nc::net::Socket::Socket (int *af*, int *type*) [protected]

Register a new socket based on address family and type.

Note:

Use the derived class Constructor

Definition at line 6 of file socket.cpp.

References m_fd.

6.19.3 Member Function Documentation

6.19.3.1 int n2nc::net::Socket::open (SocketAddress & *addr*) [virtual]

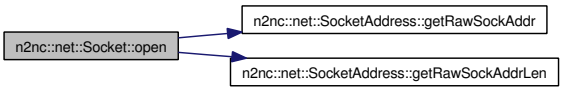
Opens a connection to a remote peer service defined in addr. Once using open() function this Socket will be treated as connection-oriented socket and by the way this is mandatory on TcpSocket (SOCK_STREAM type). NOTE: most functions returns native values.

Definition at line 60 of file socket.cpp.

References n2nc::net::SocketAddress::getRawSockAddr(), n2nc::net::SocketAddress::getRawSockAddrLen(), and m_fd.

Referenced by ThreadTest::entry_point().

Here is the call graph for this function:



Here is the caller graph for this function:

6.19.3.2 int n2nc::net::Socket::send (void * *buf*, size_t *len*) [virtual]

Send data buf to remote peer

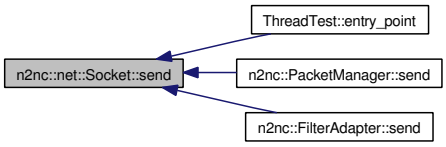
Reimplemented in n2nc::FilterAdapter, and n2nc::PacketManager.

Definition at line 68 of file socket.cpp.

References m_fd.

Referenced by ThreadTest::entry_point(), n2nc::PacketManager::send(), and n2nc::FilterAdapter::send().

Here is the caller graph for this function:



6.19.3.3 size_t n2nc::net::Socket::recvCheck () [virtual]

Check wheter the socket has Received data.Return the lenght of data in the kernel buffer

Definition at line 77 of file socket.cpp.

References m_fd, and recv().

Here is the call graph for this function:

6.19.3.4 int n2nc::net::Socket::recv (void * *buf*, size_t *len*) [virtual]

Receive len leght data from remote peer and store it to buf

Reimplemented in n2nc::FilterAdapter, and n2nc::PacketManager.

Examples:

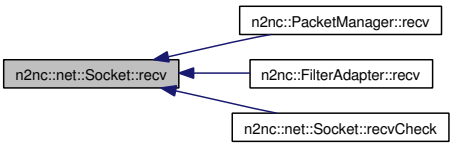
socket_ex.cpp.

Definition at line 87 of file socket.cpp.

References m_fd.

Referenced by n2nc::PacketManager::recv(), n2nc::FilterAdapter::recv(), and recvCheck().

Here is the caller graph for this function:



6.19.3.5 int n2nc::net::Socket::close () [virtual]

Closes the socket

Definition at line 95 of file socket.cpp.

References m_fd.

6.19.3.6 int n2nc::net::Socket::bind (SocketAddress & *addr*) [virtual]

Bind the socket to addr

Examples:

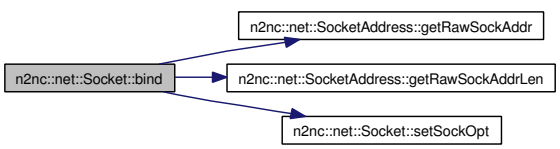
socket_ex.cpp.

Definition at line 51 of file socket.cpp.

References n2nc::net::SocketAddress::getRawSockAddr(), n2nc::net::SocketAddress::getRawSockAddrLen(), m_fd, and setSockOpt().

Referenced by n2nc::Server::listen().

Here is the call graph for this function:



Here is the caller graph for this function:

6.19.3.7 bool n2nc::net::Socket::isConnected () [virtual]

Checks if the socket is connected

Definition at line 161 of file socket.cpp.

6.19.3.8 int n2nc::net::Socket::getAddressFamily ()

Returns:

the address family in OS defined format

Definition at line 31 of file socket.cpp.

6.19.3.9 int n2nc::net::Socket::getType ()

Returns:

the type in OS defined format

Definition at line 105 of file socket.cpp.

6.19.3.10 int n2nc::net::Socket::getFD () const

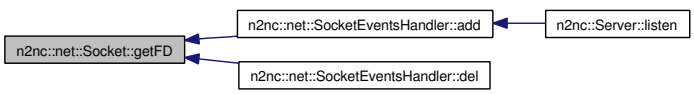
return the OS defined format file description (HANDLE in win32)

Definition at line 36 of file socket.cpp.

References m_fd.

Referenced by n2nc::net::SocketEventsHandler::add(), and n2nc::net::SocketEventsHandler::del().

Here is the caller graph for this function:



6.19.3.11 int n2nc::net::Socket::setSockOpt (int *optname*, bool *value*, int *level* = SOL_SOCKET)

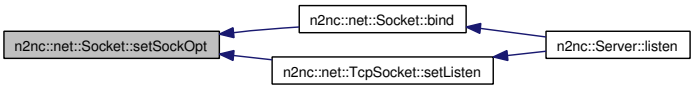
sets boolean socket option for now, boolean options only

Definition at line 40 of file socket.cpp.

References m_fd.

Referenced by bind(), and n2nc::net::TcpSocket::setListen().

Here is the caller graph for this function:



6.19.3.12 bool n2nc::net::Socket::getSockOpt (int *optname*, int *level* = SOL_SOCKET)

gets a boolean socket option

Definition at line 45 of file socket.cpp.

References m_fd.

6.19.3.13 SocketAddress * n2nc::net::Socket::getLocalSocketAddress () const

gets the sockaddr (new) associated to local endpoint

Definition at line 110 of file socket.cpp.

References m_fd.

6.19.3.14 SocketAddress * n2nc::net::Socket::getRemoteSocketAddress () const

gets the sockaddr (new) associated to remote endpoint

Definition at line 135 of file socket.cpp.

References m_fd.

6.19.4 Member Data Documentation

6.19.4.1 int n2nc::net::Socket::m_fd [protected]

save some context switch but led memory

Definition at line 62 of file socket.h.

Referenced by n2nc::net::TcpSocket::accept(), bind(), close(), getFD(), getLocalSocketAddress(), getRemoteSocketAddress(), getSockOpt(), open(), recv(), recvCheck(), send(), n2nc::net::TcpSocket::setListen(), setSockOpt(), and Socket().

The documentation for this class was generated from the following files:

- socket.h
- socket.cpp

6.20 n2nc::net::SocketAddress Class Reference

socket address base class (similar to sockaddr_storage)

`#include <socketaddress.h>`

Public Types

- typedef sockaddr_storage **Storage**

Public Member Functions

- SocketAddress (struct sockaddr *sa)
- **SocketAddress** (Address *addr, uint16_t port)
- int **getAddressFamily** ()
- struct sockaddr * **getRawSockAddr** ()
- int **getRawSockAddrLen** ()
- std::string **toString** ()
- Address * **getAddress** ()

6.20.1 Detailed Description

socket address base class (similar to sockaddr_storage)

Author:

fabsoft <fabsoft@gmail.com>

Examples:

socket_ex.cpp.

Definition at line 16 of file socketaddress.h.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 n2nc::net::SocketAddress::SocketAddress (struct sockaddr * *sa*)

Creates a new SocketAddress represented by sockaddr sa

Definition at line 12 of file socketaddress.cpp.

The documentation for this class was generated from the following files:

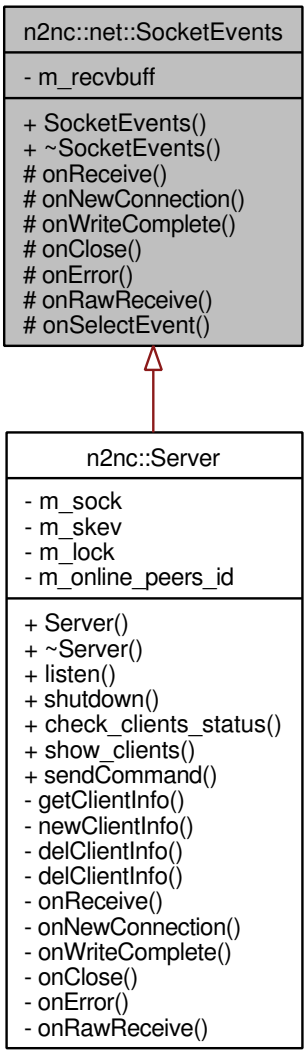
- socketaddress.h
- socketaddress.cpp

6.21 n2nc::net::SocketEvents Class Reference

Socket events to avoid explicits callbacks.

`#include <socketevents.h>`

Inheritance diagram for n2nc::net::SocketEvents:



Protected Member Functions

- virtual int **onReceive** (Socket &sock, int dtlen)=0
- virtual int **onNewConnection** (Socket &sock)=0
- virtual int **onWriteComplete** (Socket &sock)=0
- virtual int **onClose** (Socket &sock)=0
- virtual int **onError** (Socket &sock)=0
- virtual int **onRawReceive** (Socket &sock)=0
- virtual int onSelectEvent (SocketEventsHandler::socket_extra_t *sock_extra, SocketEventsHandler::check_for_t cause)

Friends

- class **SocketEventsHandler**

6.21.1 Detailed Description

Socket events to avoid explicits callbacks.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 17 of file socketevents.h.

6.21.2 Member Function Documentation

6.21.2.1 int n2nc::net::SocketEvents::onSelectEvent (SocketEventsHandler::socket_extra_t * *sock_extra*, SocketEventsHandler::check_for_t *cause*) [protected, virtual]

when returns 0

NOTE defaulting to get the socket disabled in select polling to avoid possible uncontrolled loops.

Definition at line 15 of file socketevents.cpp.

References n2nc::net::SocketEventsHandler::EXCEPT, n2nc::net::SocketEventsHandler::NOTHING, n2nc::net::SocketEventsHandler::READ, and n2nc::net::SocketEventsHandler::WRITE.

The documentation for this class was generated from the following files:

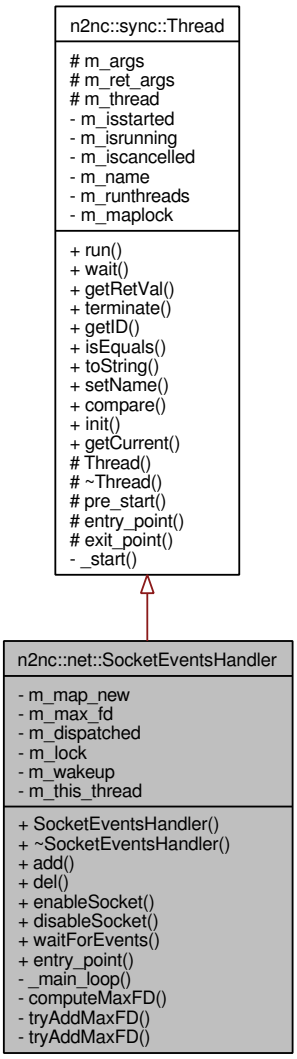
- socketevents.h
- socketevents.cpp

6.22 n2nc::net::SocketEventsHandler Class Reference

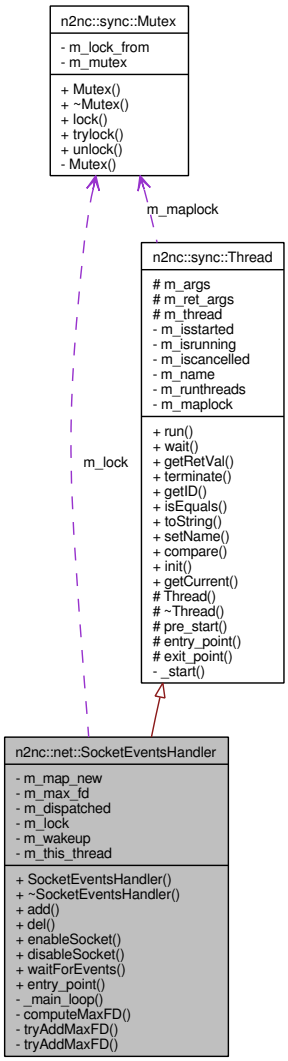
Provides asynchronus socket managment. this class provides a method to handle all events generated by Socket. Basically it starts a new thread which poll on any Socket by using select(), it generates a callback event defined in SocketEvents interface. This way avoid the plenty use of select() and move programmer to a event-based socket management.

`#include <socketeventshandler.h>`

Inheritance diagram for n2nc::net::SocketEventsHandler:



Collaboration diagram for n2nc::net::SocketEventsHandler:



Public Types

- enum check_for_t { READ = 1, WRITE = 2, EXCEPT = 4, NOTHING = 8 }

Public Member Functions

- SocketEventsHandler ()
- int add (Socket &sock, check_for_t checkfor, SocketEvents &se)
- int del (Socket &sock)
- bool **enableSocket** (const Socket &sock)
- bool **disableSocket** (const Socket &sock)
- int waitForEvents ()
- virtual void * entry_point ()

Classes

- struct **socket_extra_t**

6.22.1 Detailed Description

Provides asynchronus socket managment. this class provides a method to handle all events generated by Socket. Basically it starts a new thread which poll on any Socket by using select(), it generates a callback event defined in SocketEvents interface. This way avoid the plenty use of select() and move programmer to a event-based socket management.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 26 of file socketeventshandler.h.

6.22.2 Member Enumeration Documentation

6.22.2.1 enum n2nc::net::SocketEventsHandler::check_for_t

Enumerator:

READ Checks for readable socket

WRITE Checks for writeable socket

EXCEPT Checks for exception on socket

NOTHING Dont checks the socket. It enable/disable the state.

Definition at line 32 of file socketeventshandler.h.

6.22.3 Constructor & Destructor Documentation

6.22.3.1 n2nc::net::SocketEventsHandler::SocketEventsHandler ()

The class should have one instance only.(singleton pattern)

Definition at line 8 of file socketeventshandler.cpp.

6.22.4 Member Function Documentation

6.22.4.1 int n2nc::net::SocketEventsHandler::add (Socket & *sock*, check_for_t *checkfor*, SocketEvents & *se*)

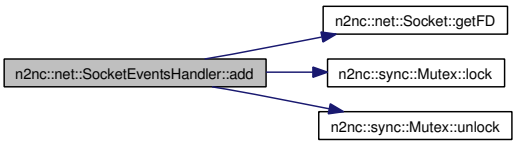
Adds a Socket to wait for events. If *se* is NULL or omitted, then this Socket *sock* will not use SocketEvents paradigm.

Definition at line 22 of file socketeventshandler.cpp.

References n2nc::net::Socket::getFD(), n2nc::sync::Mutex::lock(), and n2nc::sync::Mutex::unlock().

Referenced by n2nc::Server::listen().

Here is the call graph for this function:



Here is the caller graph for this function:

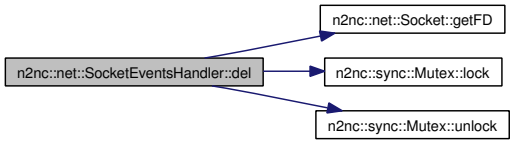
6.22.4.2 int n2nc::net::SocketEventsHandler::del (Socket & *sock*)

Removes sock from handler

Definition at line 43 of file socketeventshandler.cpp.

References n2nc::net::Socket::getFD(), n2nc::sync::Mutex::lock(), and n2nc::sync::Mutex::unlock().

Here is the call graph for this function:



6.22.4.3 int n2nc::net::SocketEventsHandler::waitForEvents ()

Main loop to catch up all events related to Socket.

Definition at line 65 of file socketeventshandler.cpp.

References n2nc::sync::Thread::run().

Referenced by n2nc::Server::listen().

Here is the call graph for this function:

Here is the caller graph for this function:

6.22.4.4 void * n2nc::net::SocketEventsHandler::entry_point () [virtual]

must implements thread code

Implements n2nc::sync::Thread.

Definition at line 59 of file socketeventshandler.cpp.

The documentation for this class was generated from the following files:

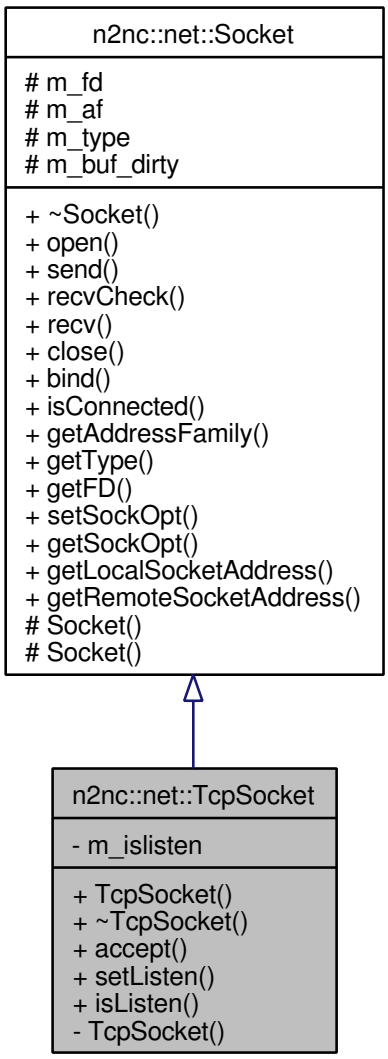
- socketeventshandler.h
- socketeventshandler.cpp

6.23 n2nc::net::TcpSocket Class Reference

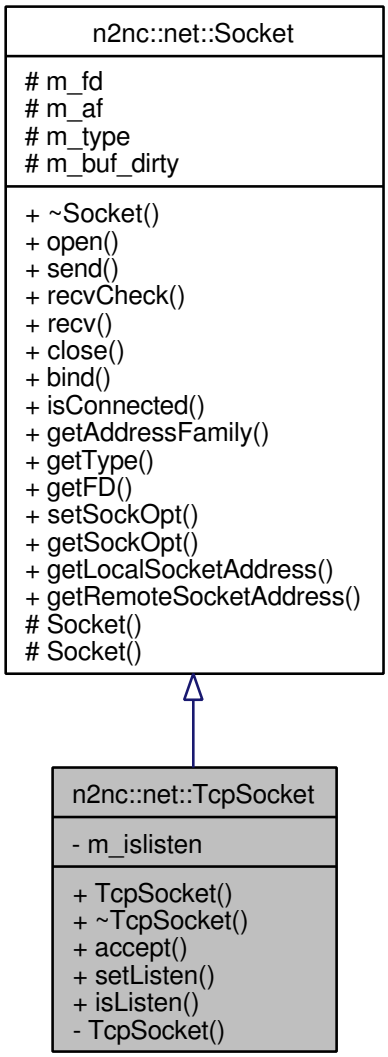
Tcp Socket implementation.

`#include <tcpsocket.h>`

Inheritance diagram for n2nc::net::TcpSocket:



Collaboration diagram for n2nc::net::TcpSocket:



Public Member Functions

- **TcpSocket** (int af)
- Socket * accept ()
- int setListen (int backlog=10)
- bool **isListen** ()

6.23.1 Detailed Description

Tcp Socket implementation.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

socket_ex.cpp.

Definition at line 14 of file tcpsocket.h.

6.23.2 Member Function Documentation

6.23.2.1 Socket * n2nc::net::TcpSocket::accept ()

Accept a new incoming connection,

Returns:

a new Socket representing the new connected socket

cleanest way. i should fit sockaddr directly by getRawSockAddr() but i prefer a local copy and use constructor

this address can be gathered from getpeername()

FIXME all in this context: check for EAGAIN

Definition at line 22 of file tcpsocket.cpp.

References n2nc::net::Socket::m_af, n2nc::net::Socket::m_fd, m_islisten, and n2nc::net::Socket::m_type.

6.23.2.2 int n2nc::net::TcpSocket::setListen (int *backlog* = `10`)

Associate a socketaddress to this socket Sets this socket as listenig socket. Once used this socket become connection-oriented socket

Definition at line 10 of file tcpsocket.cpp.

References n2nc::net::Socket::m_fd, and n2nc::net::Socket::setSockOpt().

Referenced by n2nc::Server::listen().

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

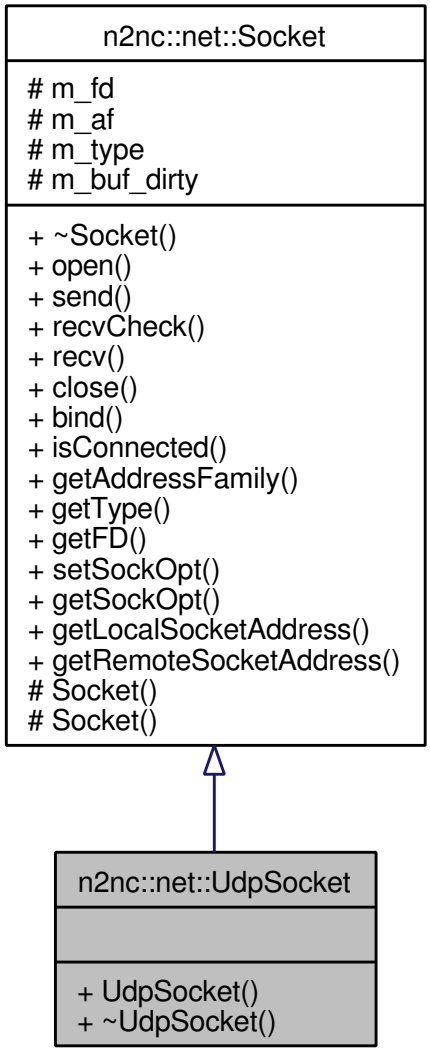
- tcpsocket.h
- tcpsocket.cpp

6.24 n2nc::net::UdpSocket Class Reference

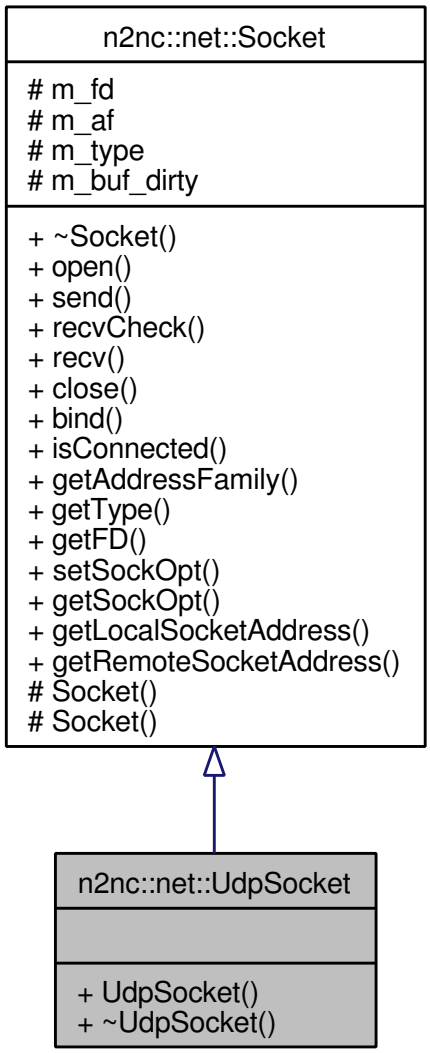
Udp Socket implementation.

`#include <udpsocket.h>`

Inheritance diagram for n2nc::net::UdpSocket:



Collaboration diagram for n2nc::net::UdpSocket:



Public Member Functions

- **UdpSocket** (int af)

6.24.1 Detailed Description

Udp Socket implementation.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 13 of file udpsocket.h.

The documentation for this class was generated from the following files:

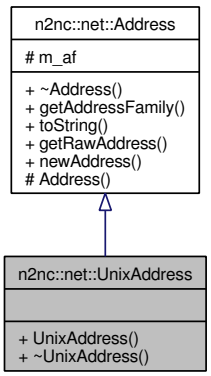
- udpsocket.h
- udpsocket.cpp

6.25 n2nc::net::UnixAddress Class Reference

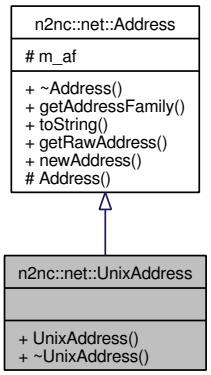
Unix Address family(AF_UNIX) abstaction.

`#include <unixaddress.h>`

Inheritance diagram for n2nc::net::UnixAddress:



Collaboration diagram for n2nc::net::UnixAddress:



6.25.1 Detailed Description

Unix Address family(AF_UNIX) abstaction.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 14 of file unixaddress.h.

The documentation for this class was generated from the following files:

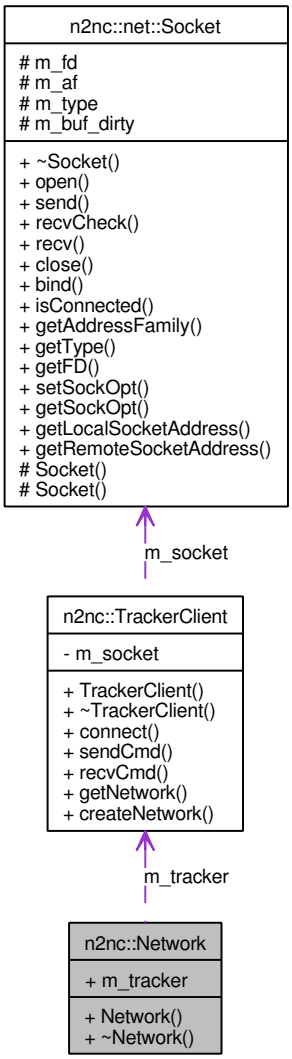
- unixaddress.h
- unixaddress.cpp

6.26 n2nc::Network Class Reference

This class represents a network.

`#include <network.h>`

Collaboration diagram for n2nc::Network:



Public Member Functions

- **Network** (TrackerClient &tracker)

Public Attributes

- TrackerClient & **m_tracker**

Friends

- class **TrackerClient**

6.26.1 Detailed Description

This class represents a network.

Basically a network is a VPN's end-point, plus all info about SocketAddress and AAA (Authorization, Authentication and Accounting)

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 18 of file network.h.

The documentation for this class was generated from the following files:

- network.h
- network.cpp

6.27 n2nc::NetworkChannel Class Reference

This class represents the comunication channel between 2 networks.

`#include <networkchannel.h>`

Public Member Functions

- bool **wantConnect** ()
- bool **wantDisconnect** ()

6.27.1 Detailed Description

This class represents the comunication channel between 2 networks.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 11 of file networkchannel.h.

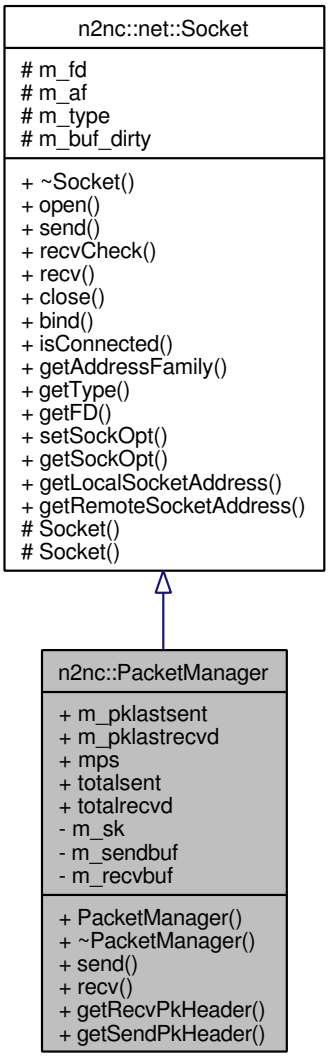
The documentation for this class was generated from the following files:

- networkchannel.h
- networkchannel.cpp

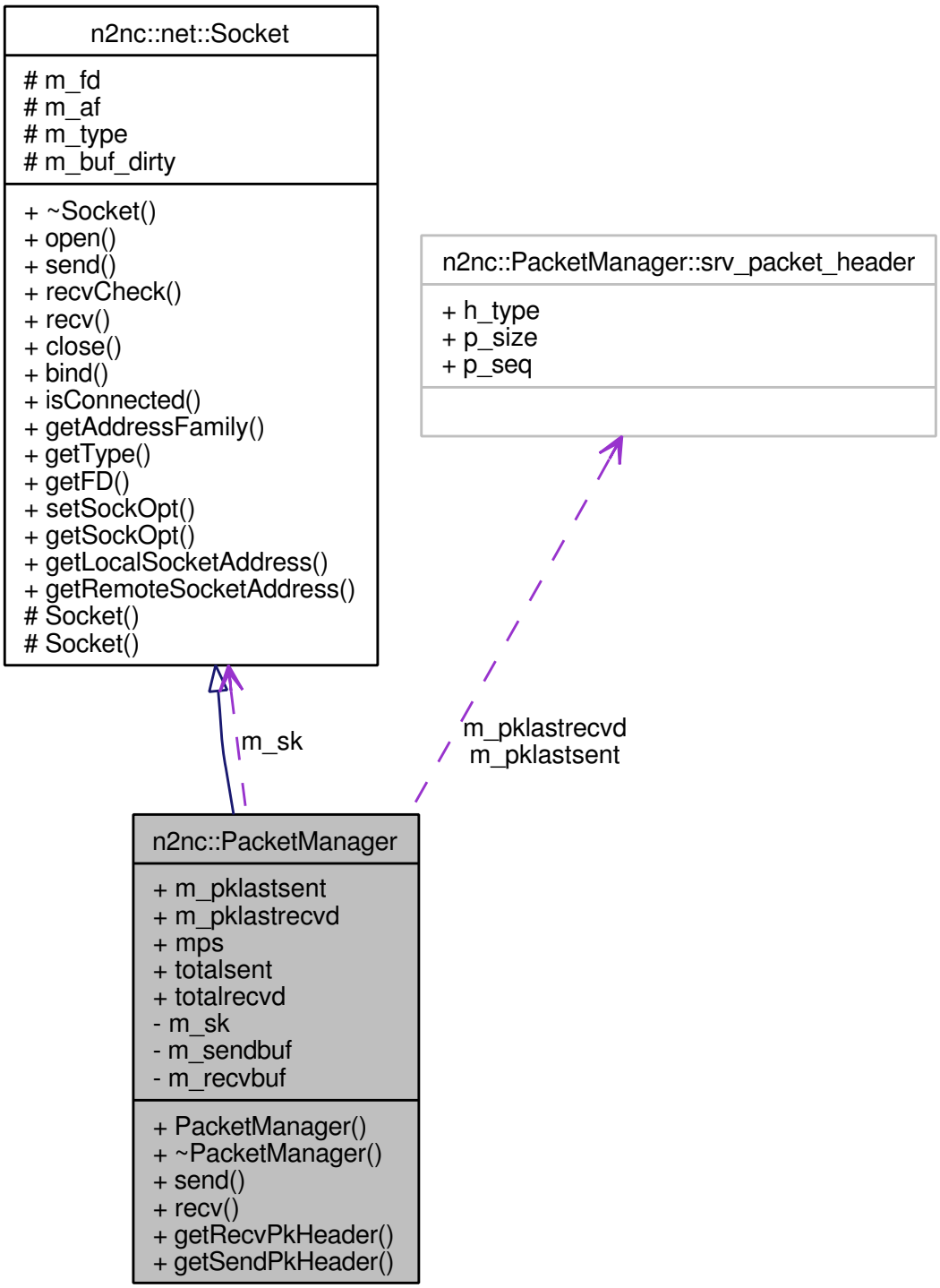
6.28 n2nc::PacketManager Class Reference

`#include <packetmanager.h>`

Inheritance diagram for n2nc::PacketManager:



Collaboration diagram for n2nc::PacketManager:



Public Types

- enum **srv_packet_type** { **SRV_PK_GARBAGE** = 0, **SRV_PK_GARBAGE_ACK**, **SRV_PK_-DATA** }

- typedef struct srv_packet_header **srv_packet_header_t**

Public Member Functions

- **PacketManager** (net::Socket *sk)
- virtual int send (void *buf, size_t len)
- virtual int recv (void *buf, size_t len)
- srv_packet_header_t * **getRecvPkHeader** ()
- srv_packet_header_t * **getSendPkHeader** ()

Public Attributes

- srv_packet_header **m_pklastsent**
- srv_packet_header **m_pklastrecvd**
- size_t mps
- uint64_t **totalsent**
- uint64_t **totalrecvd**

Classes

- struct **srv_packet_header**

6.28.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 13 of file packetmanager.h.

6.28.2 Member Function Documentation

6.28.2.1 int n2nc::PacketManager::send (void * *buf*, size_t *len*) [virtual]

Send data buf to remote peer

Reimplemented from n2nc::net::Socket.

Definition at line 27 of file packetmanager.cpp.

References n2nc::net::Socket::send().

Here is the call graph for this function:

6.28.2.2 int n2nc::PacketManager::recv (void * *buf*, size_t *len*) [virtual]

Receive len leght data from remote peer and store it to buf

Reimplemented from n2nc::net::Socket.

Definition at line 49 of file packetmanager.cpp.

References n2nc::net::Socket::recv().

Here is the call graph for this function:

6.28.3 Member Data Documentation

6.28.3.1 size_t n2nc::PacketManager::mps

maximum packet size

Definition at line 38 of file packetmanager.h.

The documentation for this class was generated from the following files:

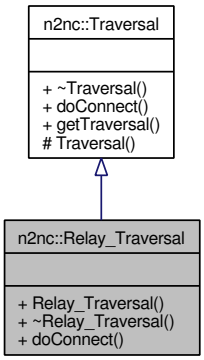
- packetmanager.h
- packetmanager.cpp

6.29 n2nc::Relay_Traversal Class Reference

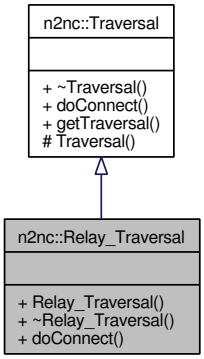
This class provides the implementation of tracker's relay/bouncer.

`#include <relay_traversal.h>`

Inheritance diagram for n2nc::Relay_Traversal:



Collaboration diagram for n2nc::Relay_Traversal:



Public Member Functions

- virtual net::Socket & doConnect (Network &net)

6.29.1 Detailed Description

This class provides the implementation of tracker's relay/bouncer.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 13 of file relay_traversal.h.

6.29.2 Member Function Documentation

6.29.2.1 net::Socket & n2nc::Relay_Traversal::doConnect (Network & *net*) [virtual]

Performs a connection to Network *net* FIXME if error ?

Implements n2nc::Traversal.

Definition at line 12 of file relay_traversal.cpp.

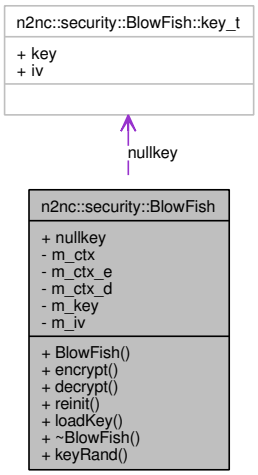
The documentation for this class was generated from the following files:

- relay_traversal.h
- relay_traversal.cpp

6.30 n2nc::security::BlowFish Class Reference

`#include <blowfish.h>`

Collaboration diagram for n2nc::security::BlowFish:



Public Member Functions

- **BlowFish** (key_t *key)
- int **encrypt** (void *inbuf, void *outbuf, size_t inlen)
- int **decrypt** (void *inbuf, void *outbuf, size_t inlen)
- int **reinit** ()
- int **loadKey** (key_t *key)

Static Public Member Functions

- static int **keyRand** (key_t *key)

Public Attributes

- key_t **nullkey**

Classes

- struct **key_t**

6.30.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 12 of file blowfish.h.

The documentation for this class was generated from the following files:

- blowfish.h
- blowfish.cpp

6.31 n2nc::security::Rsa Class Reference

`#include <rsa.h>`

Public Types

- typedef uint32_t **ID**

Public Member Functions

- int **loadPubFromFile** (std::string filename)
- int **loadPriFromFile** (std::string filename)
- int **encrypt** (void *inbuf, void *outbuf, size_t inlen)
- int **decrypt** (void *inbuf, void *outbuf, size_t inlen)
- int **genkey** (std::string filename)
- int **getID** ()

6.31.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 11 of file rsa.h.

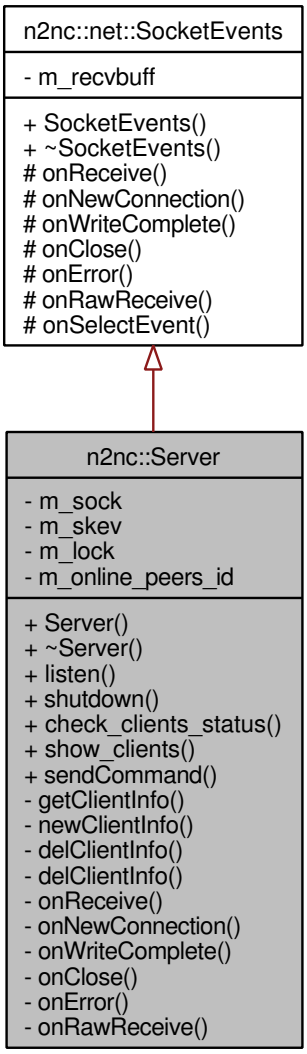
The documentation for this class was generated from the following files:

- rsa.h
- rsa.cpp

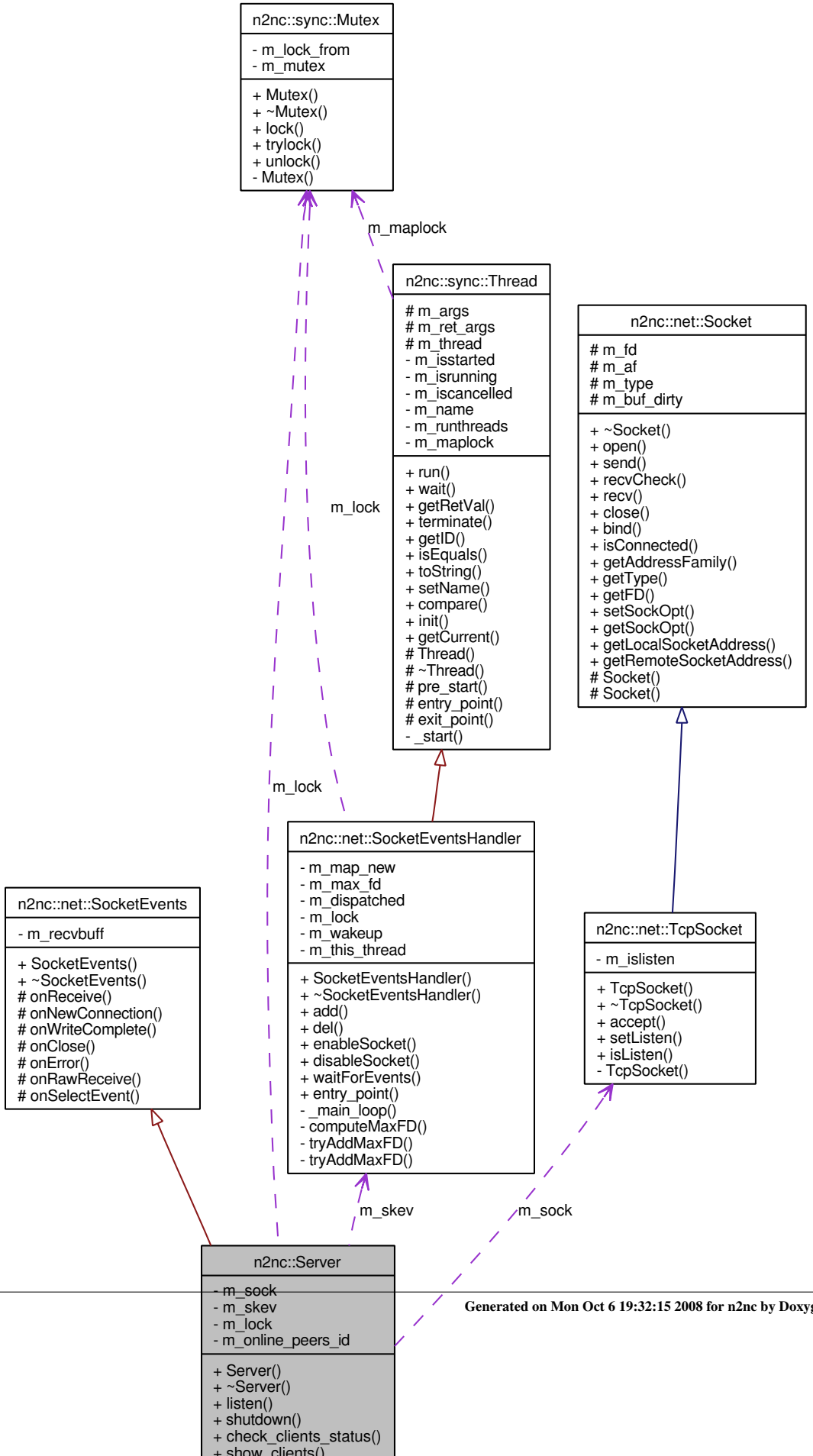
6.32 n2nc::Server Class Reference

`#include <server.h>`

Inheritance diagram for n2nc::Server:



Collaboration diagram for n2nc::Server:



Public Types

- enum **server_command** {

  CMD_REGISTER, **CMD_REGISTER_ACK**, **CMD_PING**, **CMD_LEAVE**,

  CMD_GET_CLIENT_INFO, **CMD_GET_CLIENT_INFO_REPLY**, **CMD_CONNECT**, **CMD_CONNECT_ACK**,

  CMD_CONNECT_OK, **CMD_CONNECT_OK_ACK**, **CMD_SMSG**, **CMD_SMSG_ACK**,

  CMD_DISCONNECT }
- enum **server_messages** { **MSG_OK** = 0, **MSG_ALREADY_REGISTERED**, **MSG_NOT_FOUND** }

Public Member Functions

- int listen (int port)
- int shutdown (std::string message)
- int check_clients_status ()
- int **show_clients** ()
- int **sendCommand** (enum server_command cmd, net::Socket &sock)

Classes

- struct **packet**

6.32.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 20 of file server.h.

6.32.2 Member Function Documentation

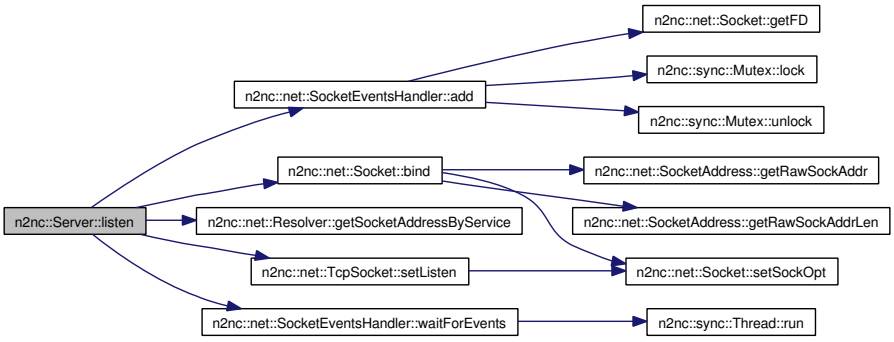
6.32.2.1 int n2nc::Server::listen (int *port*)

brings the server to listen at given port

Definition at line 61 of file server.cpp.

References n2nc::net::SocketEventsHandler::add(), n2nc::net::Socket::bind(), n2nc::net::Resolver::getSocketAddressByService(), n2nc::net::SocketEventsHandler::READ, n2nc::net::TcpSocket::setListen(), and n2nc::net::SocketEventsHandler::waitForEvents().

Here is the call graph for this function:



6.32.2.2 int n2nc::Server::shutdown (std::string *message*)

causes the server shutdown just after sending the given message to clients

Definition at line 75 of file server.cpp.

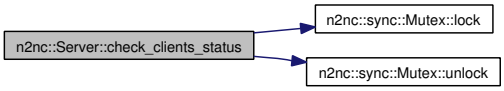
6.32.2.3 int n2nc::Server::check_clients_status ()

checks if any client goes to timeout, if so that client will be removed from the list of online peers

Definition at line 265 of file server.cpp.

References n2nc::sync::Mutex::lock(), and n2nc::sync::Mutex::unlock().

Here is the call graph for this function:



The documentation for this class was generated from the following files:

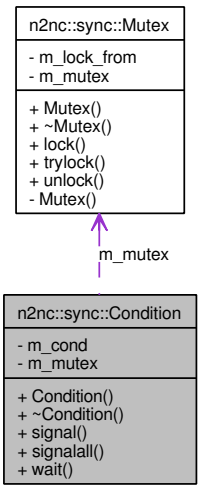
- server.h
- server.cpp

6.33 n2nc::sync::Condition Class Reference

Conditional variable implementation.

`#include <condition.h>`

Collaboration diagram for n2nc::sync::Condition:



Public Member Functions

- **Condition** (Mutex &mutex)
- bool **signal** ()
- bool **signalall** ()
- bool wait ()

6.33.1 Detailed Description

Conditional variable implementation.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 15 of file condition.h.

6.33.2 Member Function Documentation

6.33.2.1 bool n2nc::sync::Condition::wait ()

Check if the thread has acquired the lock

error

Definition at line 10 of file condition.cpp.

References n2nc::sync::Mutex::m_mutex.

The documentation for this class was generated from the following files:

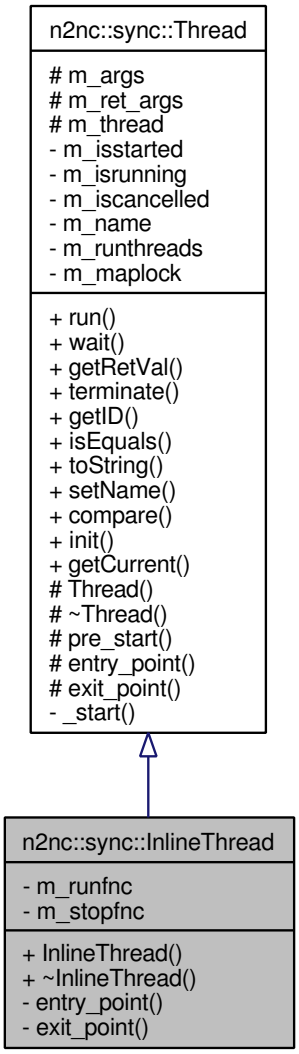
- condition.h
- condition.cpp

6.34 n2nc::sync::InlineThread Class Reference

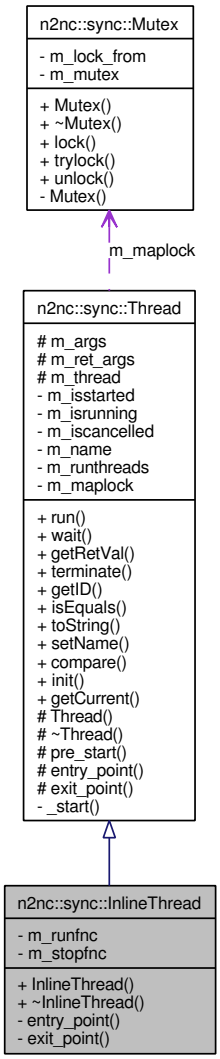
C-Style threading model.

`#include <thread.h>`

Inheritance diagram for n2nc::sync::InlineThread:



Collaboration diagram for n2nc::sync::InlineThread:



Public Types

- typedef void ∗(∗ m_start_routine)(void ∗)
- typedef int(∗ **m_stop_routine**)(void ∗)

Public Member Functions

- InlineThread (m_start_routine fnc, m_stop_routine fncstop=NULL)

6.34.1 Detailed Description

C-Style threading model.

This thread class utility take a function pointer instead implementing virtual method.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

threads_ex.cpp.

Definition at line 107 of file thread.h.

6.34.2 Member Typedef Documentation

6.34.2.1 typedef void*(* n2nc::sync::InlineThread::m_start_routine)(void *)

Start routine fingerprint

6.34.3 Constructor & Destructor Documentation

6.34.3.1 n2nc::sync::InlineThread::InlineThread (m_start_routine *fnc*, m_stop_routine *fncstop* = NULL) [inline]

Sets entry point for the thread

Definition at line 113 of file thread.h.

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

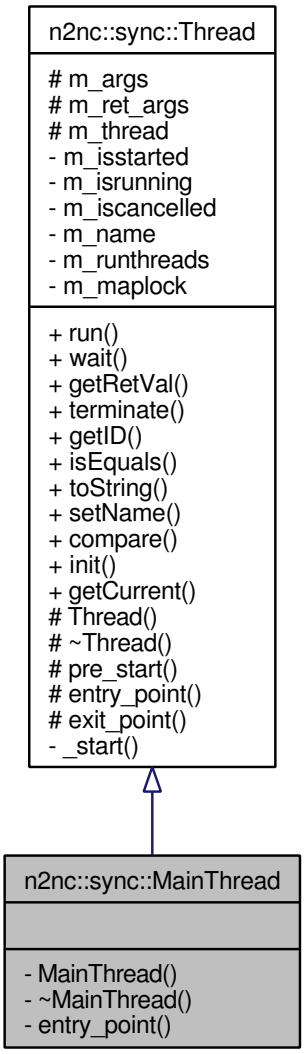
- thread.h

6.35 n2nc::sync::MainThread Class Reference

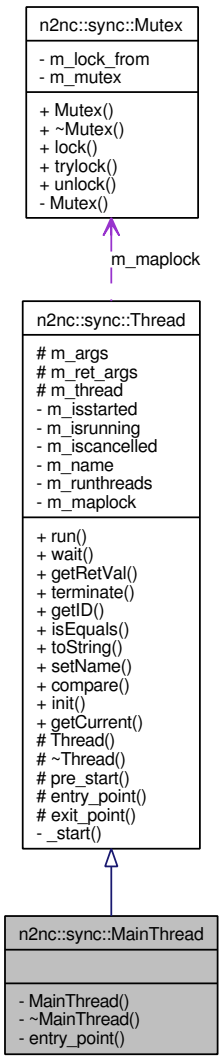
Workaround to emulate the main thread. Do not use this.

`#include <thread.h>`

Inheritance diagram for n2nc::sync::MainThread:



Collaboration diagram for n2nc::sync::MainThread:



Friends

- class **Thread**

6.35.1 Detailed Description

Workaround to emulate the main thread. Do not use this.

Do not call this directly. Use Thread::init() instead. once Thread::init() is called by main thread, it initializes a Thread object associated to main thread. By the way it is possible to call eg. Thread::getCurrent()->setDescription("what im doing"); and LOGMET() << "hello" ; output: Thread: what im doing Hello.

Definition at line 83 of file thread.h.

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

- thread.h

6.36 n2nc::sync::Mutex Class Reference

Mutex implementation.

`#include <mutex.h>`

Public Member Functions

- bool **lock** ()
- bool **trylock** ()
- bool **unlock** ()

Friends

- class **Condition**

6.36.1 Detailed Description

Mutex implementation.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 15 of file mutex.h.

The documentation for this class was generated from the following files:

- mutex.h
- mutex.cpp

6.37 n2nc::sync::Semaphore Class Reference

This class implements the semaphore paradigm.

`#include <semaphore.h>`

Public Member Functions

- **Semaphore** (uint init_value=0)
- int **getValue** ()
- bool **tryWait** ()
- bool **wait** ()
- bool **post** ()

6.37.1 Detailed Description

This class implements the semaphore paradigm.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

threads_ex.cpp.

Definition at line 14 of file semaphore.h.

The documentation for this class was generated from the following files:

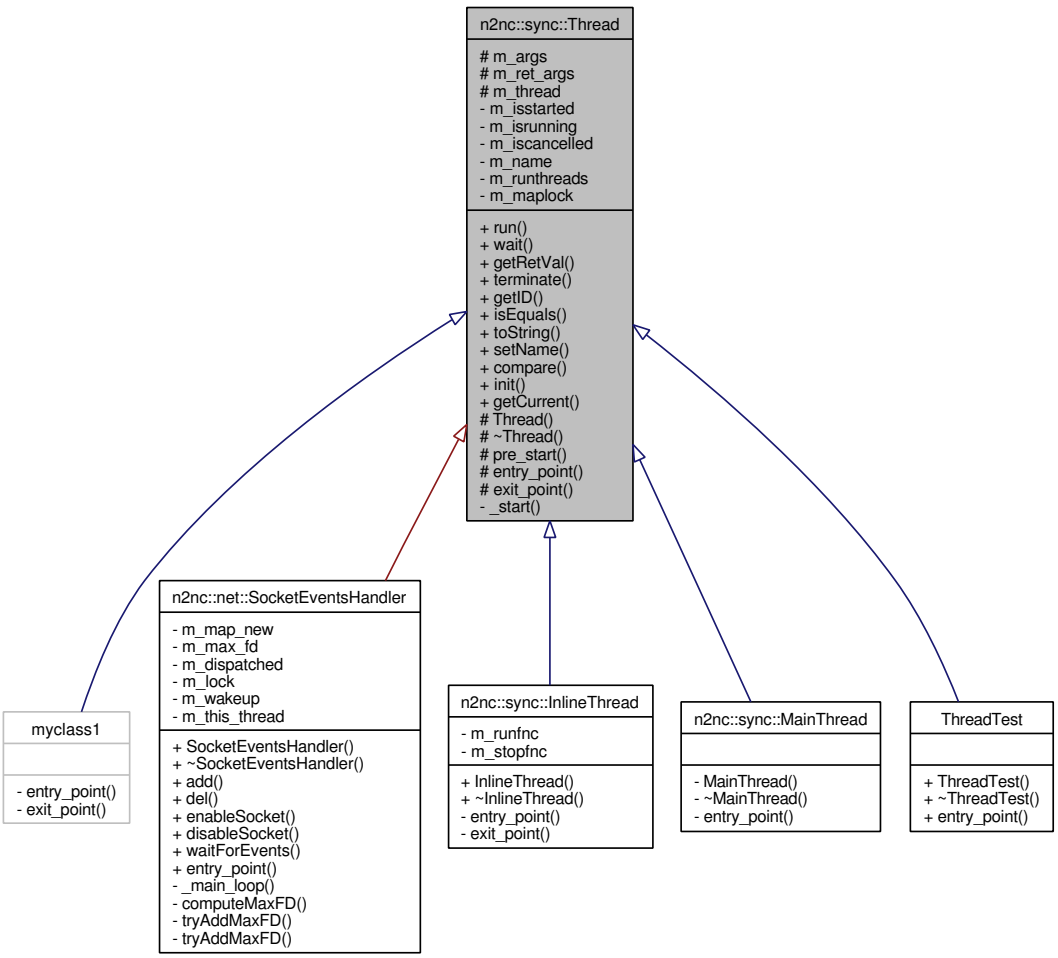
- semaphore.h
- semaphore.cpp

6.38 n2nc::sync::Thread Class Reference

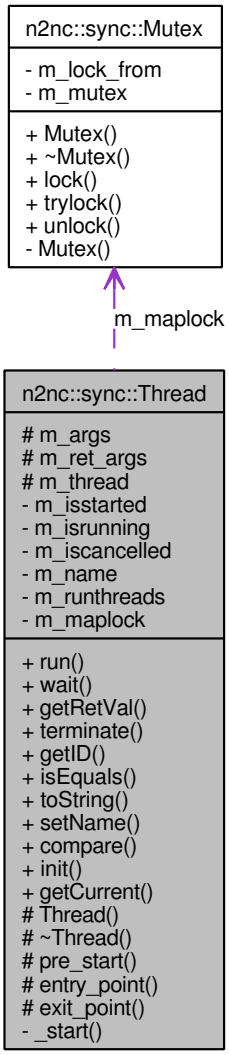
Base Threads interface.

`#include <thread.h>`

Inheritance diagram for n2nc::sync::Thread:



Collaboration diagram for n2nc::sync::Thread:



Public Member Functions

- int run (void *args)
- int wait (void **retval)
- void * getRetVal ()
- int terminate ()
- int getID ()
- bool isEquals (Thread *t)
- std::string toString ()
- void **setName** (std::string name)

Static Public Member Functions

- static bool **compare** (Thread &t1, Thread &t2)

- static bool **init** ()
- static Thread & **getCurrent** ()

Protected Member Functions

- virtual int pre_start ()
- virtual void * entry_point ()=0
- virtual int exit_point ()

Protected Attributes

- void * **m_args**
- void * **m_ret_args**
- pthread_t **m_thread**

6.38.1 Detailed Description

Base Threads interface.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

socket_ex.cpp, and threads_ex.cpp.

Definition at line 15 of file thread.h.

6.38.2 Member Function Documentation

6.38.2.1 int n2nc::sync::Thread::run (void * *args*)

Runs the thread by calling entry_point() routine

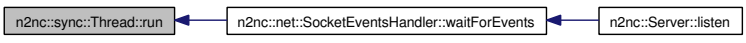
Examples:

socket_ex.cpp, and threads_ex.cpp.

Definition at line 21 of file thread.cpp.

Referenced by n2nc::net::SocketEventsHandler::waitForEvents().

Here is the caller graph for this function:



6.38.2.2 int n2nc::sync::Thread::wait (void ** *retval*)

Waits (join) the thread and returns the exit-value

Definition at line 41 of file thread.cpp.

6.38.2.3 void * n2nc::sync::Thread::getRetVal ()

FIXME getRetVal

Examples:

threads_ex.cpp.

Definition at line 123 of file thread.cpp.

6.38.2.4 int n2nc::sync::Thread::terminate ()

Cancels the thread execution

Examples:

socket_ex.cpp.

Definition at line 35 of file thread.cpp.

6.38.2.5 int n2nc::sync::Thread::getID ()

the thread ID

6.38.2.6 bool n2nc::sync::Thread::isEquals (Thread * *t*)

Compares thread t. Same of Thread::compare(this, t2)

6.38.2.7 std::string n2nc::sync::Thread::toString ()

Returns:

the thread name/description

Definition at line 119 of file thread.cpp.

6.38.2.8 int n2nc::sync::Thread::pre_start () [protected, virtual]

setups thread function

Definition at line 16 of file thread.cpp.

6.38.2.9 virtual void* n2nc::sync::Thread::entry_point () [protected, pure virtual]

must implements thread code

Implemented in n2nc::net::SocketEventsHandler, and ThreadTest.

6.38.2.10 int n2nc::sync::Thread::exit_point () [protected, virtual]

must sets the thread in non-running state, here it should implements clean-up routine

Definition at line 28 of file thread.cpp.

The documentation for this class was generated from the following files:

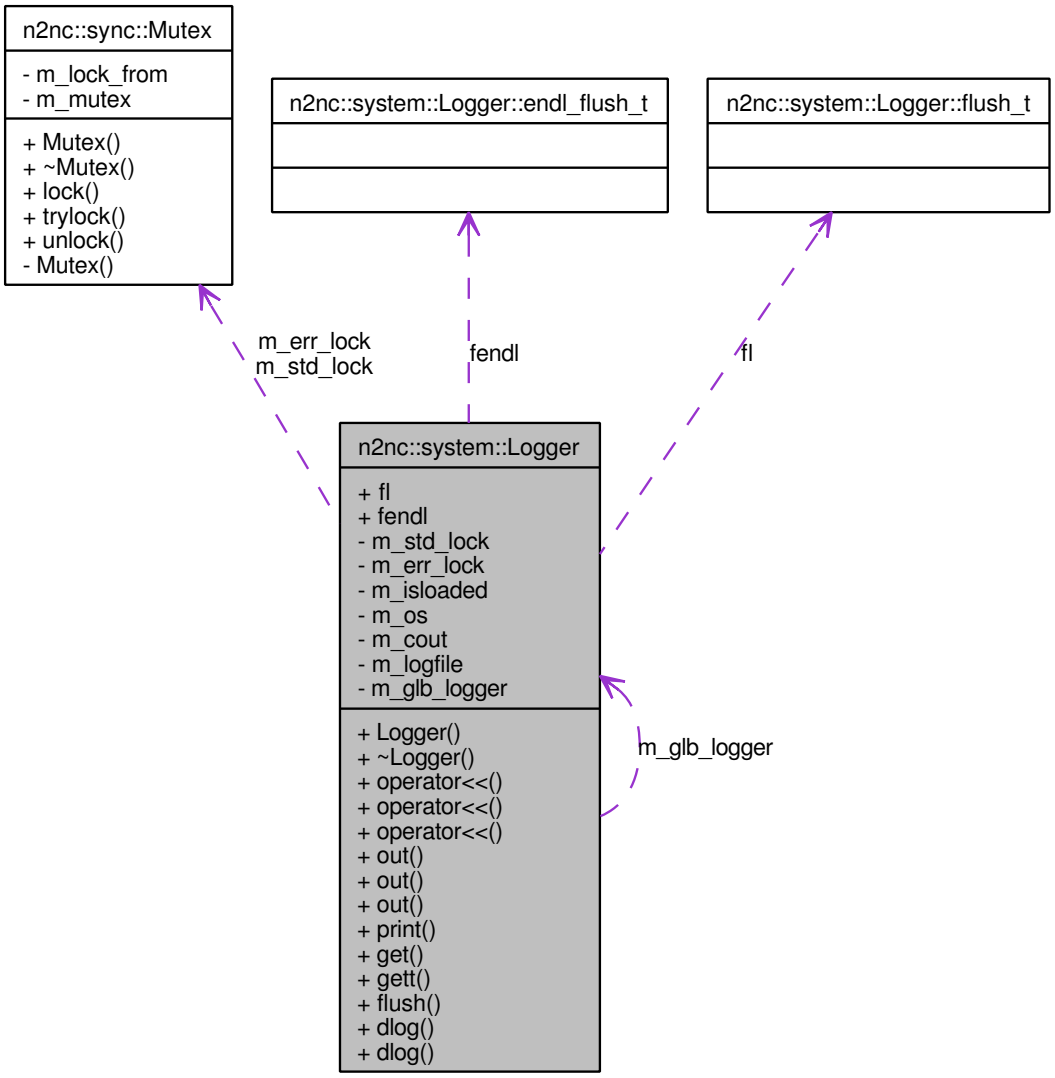
- thread.h
- thread.cpp

6.39 n2nc::system::Logger Class Reference

Provides a thead-safe and reentrant logging facility.

`#include <logger.h>`

Collaboration diagram for n2nc::system::Logger:



Public Member Functions

- Logger & **operator**<< (char *str)
- Logger & **operator**<< (std::string str)
- Logger & **operator**<< (std::string &str)

Static Public Member Functions

- static int **out** (std::string &str)

- static int **out** (std::string str)
- static int **out** (char *str)
- static void **print** (char *frm,...)
- static std::ostringstream & get ()
- static std::ostringstream & gett ()
- static std::ostringstream & **flush** ()
- static std::ostream & dlog ()
- static std::ostream & dlog (std::ostream &os)

Static Public Attributes

- static flush_t **fl**
- static endl_flush_t **fendl**

Friends

- void **operator**<< (std::ostream &, Logger::endl_flush_t)
- void **operator**<< (std::ostream &, Logger::flush_t)

Classes

- class endl_flush_t
- class flush_t

6.39.1 Detailed Description

Provides a thead-safe and reentrant logging facility.

more about logger

Author:

fabsoft <fabsoft@gmail.com> FIXME thread-safety & reentrant compliance needed !

Definition at line 28 of file logger.h.

6.39.2 Member Function Documentation

6.39.2.1 std::ostringstream & n2nc::system::Logger::get () [static]

Returns:

the ostringstream logger

Definition at line 23 of file logger.cpp.

6.39.2.2 std::ostringstream & n2nc::system::Logger::gett () [static]

Returns:

the ostringstream logger with additional thread info

Definition at line 27 of file logger.cpp.

6.39.2.3 std::ostream & n2nc::system::Logger::dlog () [static]

Debug log Debug output

Definition at line 96 of file logger.cpp.

6.39.2.4 std::ostream & n2nc::system::Logger::dlog (std::ostream & *os*) [static]

Debug output to stream os

Definition at line 104 of file logger.cpp.

The documentation for this class was generated from the following files:

- logger.h
- logger.cpp

6.40 n2nc::system::Logger::endl_flush_t Class Reference

`#include <logger.h>`

6.40.1 Detailed Description

Insert a newline and Flush the stream.

Definition at line 57 of file logger.h.

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

- logger.h

6.41 n2nc::system::Logger::flush_t Class Reference

`#include <logger.h>`

6.41.1 Detailed Description

Flush the stream.

Definition at line 59 of file logger.h.

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

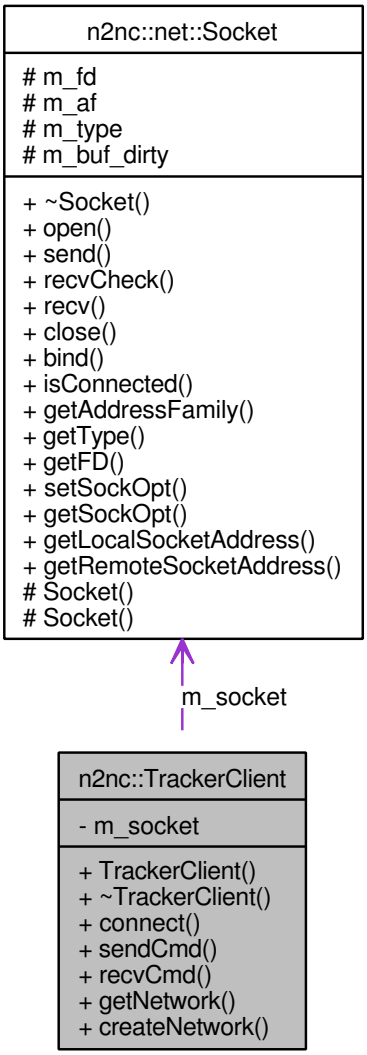
- logger.h

6.42 n2nc::TrackerClient Class Reference

This class proviedes the code to connect and exchange commands to and from a TrackerServer.

`#include <trackerclient.h>`

Collaboration diagram for n2nc::TrackerClient:



Public Types

- enum **command_t** { **JOIN_NET**, **PART_NET** }

Public Member Functions

- bool connect (const net::SocketAddress &address)
- bool sendCmd (std::string cmd)
- std::string recvCmd ()

- n2nc::Network & getNetwork (std::string name)
- bool **createNetwork** (std::string netname)

6.42.1 Detailed Description

This class proviedes the code to connect and exchange commands to and from a TrackerServer.

See also:

TrackerServer

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 17 of file trackerclient.h.

6.42.2 Member Function Documentation

6.42.2.1 bool n2nc::TrackerClient::connect (const net::SocketAddress & *address*)

Create a connection to the server tracker

6.42.2.2 bool n2nc::TrackerClient::sendCmd (std::string *cmd*)

Sends a command to the server tracker

6.42.2.3 std::string n2nc::TrackerClient::recvCmd ()

Receives a command from the server tracker

6.42.2.4 n2nc::Network& n2nc::TrackerClient::getNetwork (std::string *name*)

Get Network by *name*

The documentation for this class was generated from the following files:

- trackerclient.h
- trackerclient.cpp

6.43 n2nc::TrackerServer Class Reference

This class provides the server tracker implementation.

`#include <trackerserver.h>`

Public Member Functions

- int start (int max_con)

6.43.1 Detailed Description

This class provides the server tracker implementation.

See also:

TrackerClient

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 15 of file trackerserver.h.

6.43.2 Member Function Documentation

6.43.2.1 int n2nc::TrackerServer::start (int *max_con*)

Starts the main listening socket

Parameters:

max_con Indicates the maximum number of connected allowed clients.

The documentation for this class was generated from the following files:

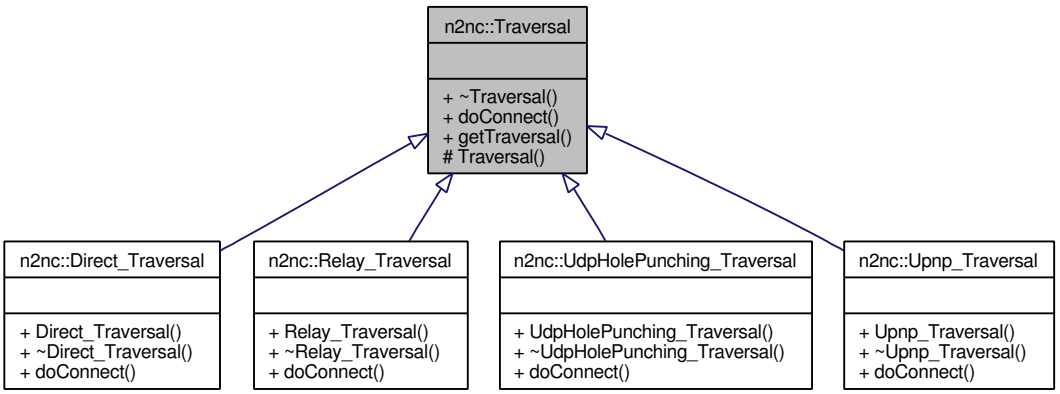
- trackerserver.h
- trackerserver.cpp

6.44 n2nc::Traversal Class Reference

This class provides the base interface to transport NAT traversal procedures.

`#include <traversal.h>`

Inheritance diagram for n2nc::Traversal:



Public Types

- enum method_t { UDP_HOLE_PUNCHING = TRVERSAL_UDP_HOLE_PUNCHING_CONST, RELAY_CONST = TRVERSAL_RELAY_CONST, UPNP_CONST = TRVERSAL_UPNP_CONST, DIRECT = TRVERSAL_DIRECT }

Public Member Functions

- virtual net::Socket & doConnect (Network &net)=0
- Traversal & getTraversal (method_t method)

6.44.1 Detailed Description

This class provides the base interface to transport NAT traversal procedures.

More on traversal

Author:

fabsoft <fabsoft@gmail.com>

continued description

Definition at line 23 of file traversal.h.

6.44.2 Member Enumeration Documentation

6.44.2.1 enum n2nc::Traversal::method_t

Enumeration for choice the traversal method

Enumerator:

UDP_HOLE_PUNCHING Udp Hole Punching method

RELAY_CONST Relay method

UPNP_CONST Upnp method

DIRECT Direct connection(dummy) method

Definition at line 28 of file traversal.h.

6.44.3 Member Function Documentation

6.44.3.1 virtual net::Socket& n2nc::Traversal::doConnect (Network & *net*) [pure virtual]

Performs a connection to Network *net* FIXME if error ?

Implemented in n2nc::Direct_Traversal, n2nc::Relay_Traversal, n2nc::UdpHolePunching_Traversal, and n2nc::Upnp_Traversal.

6.44.3.2 Traversal& n2nc::Traversal::getTraversal (method_t *method*)

Default factory method to get a new Traversal Method

The documentation for this class was generated from the following files:

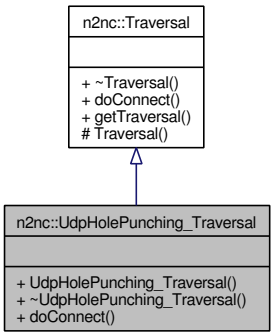
- traversal.h
- traversal.cpp

6.45 n2nc::UdpHolePunching_Traversal Class Reference

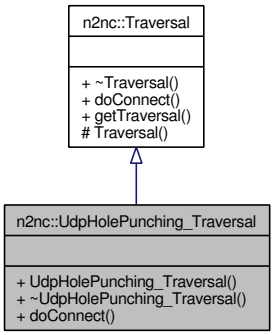
This class provides Udp Hole Punching NAT traversal method.

`#include <udpholepunching_traversal.h>`

Inheritance diagram for n2nc::UdpHolePunching_Traversal:



Collaboration diagram for n2nc::UdpHolePunching_Traversal:



Public Member Functions

- virtual net::Socket & doConnect (Network &net)

6.45.1 Detailed Description

This class provides Udp Hole Punching NAT traversal method.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 13 of file udpholepunching_traversal.h.

6.45.2 Member Function Documentation

6.45.2.1 net::Socket & n2nc::UdpHolePunching_Traversal::doConnect (Network & *net*) [virtual]

Performs a connection to Network *net* FIXME if error ?

Implements n2nc::Traversal.

Definition at line 12 of file udpholepunching_traversal.cpp.

The documentation for this class was generated from the following files:

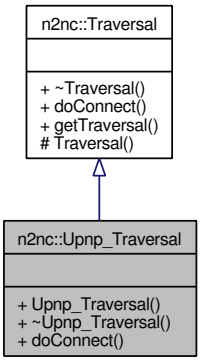
- udpholepunching_traversal.h
- udpholepunching_traversal.cpp

6.46 n2nc::Upnp_Traversal Class Reference

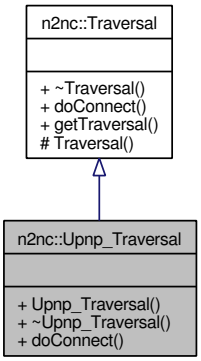
This class provides the UPNP implementation.

`#include <upnp_traversal.h>`

Inheritance diagram for n2nc::Upnp_Traversal:



Collaboration diagram for n2nc::Upnp_Traversal:



Public Member Functions

- virtual net::Socket & doConnect (Network &arg1)

6.46.1 Detailed Description

This class provides the UPNP implementation.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 13 of file upnp_traversal.h.

6.46.2 Member Function Documentation

6.46.2.1 net::Socket & n2nc::Upnp_Traversal::doConnect (Network & *net*) [virtual]

Performs a connection to Network *net* FIXME if error ?

Implements n2nc::Traversal.

Definition at line 16 of file upnp_traversal.cpp.

The documentation for this class was generated from the following files:

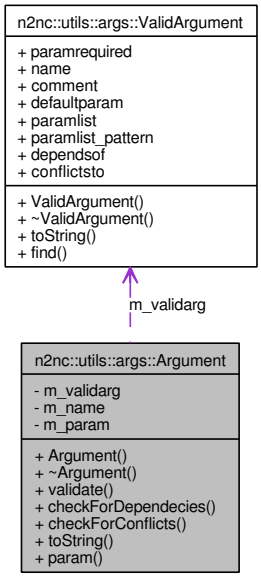
- upnp_traversal.h
- upnp_traversal.cpp

6.47 n2nc::utils::args::Argument Class Reference

This class represent a already istanciated argument.

`#include <argument.h>`

Collaboration diagram for n2nc::utils::args::Argument:



Public Member Functions

- Argument (string name, string param)
- bool validate (std::vector< ValidArgument > &validargs) throw ()
- bool checkForDependecies (vector< Argument > &args)
- bool checkForConflicts (vector< Argument > &args)
- string toString ()
- string param ()

Friends

- class **ArgumentsHelper**

6.47.1 Detailed Description

This class represent a already istanciated argument.

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 17 of file argument.h.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 n2nc::utils::args::Argument::Argument (string *name*, string *param*)

Creates a new argument, but leave it unchecked

Definition at line 7 of file argument.cpp.

6.47.3 Member Function Documentation

6.47.3.1 bool n2nc::utils::args::Argument::validate (std::vector< ValidArgument > & *validargs*) throw ()

check if the argument has valid name and parameter

Definition at line 13 of file argument.cpp.

6.47.3.2 bool n2nc::utils::args::Argument::checkForDependecies (vector< Argument > & *args*)

Tests if the dependencies arguments are satisfied.

Definition at line 68 of file argument.cpp.

References n2nc::utils::args::ValidArgument::dependsof.

6.47.3.3 bool n2nc::utils::args::Argument::checkForConflicts (vector< Argument > & *args*)

Tests if there some conflictual argument.

Returns:

true if no conflicts are checked.

Definition at line 96 of file argument.cpp.

References n2nc::utils::args::ValidArgument::conflictsto.

6.47.3.4 string n2nc::utils::args::Argument::toString ()

Tests if the given paramenter is valid. Tests if the given name is valid.

Definition at line 117 of file argument.cpp.

6.47.3.5 string n2nc::utils::args::Argument::param ()

Returns:

The parameter

Definition at line 159 of file argument.cpp.

The documentation for this class was generated from the following files:

- argument.h
- argument.cpp

6.48 n2nc::utils::args::ArgumentsHelper Class Reference

This class provides a helper/utility to handle command line argumets.

`#include <argumentshelper.h>`

Public Member Functions

- ArgumentsHelper (char **args, int argc)
- ValidArgument * **addValid** (string name)
- bool validate ()
- int **size** ()
- Argument * **getArgumentByName** (string name) throw ()
- Argument * **operator[ ]** (string name) throw ()
- string **toString** ()
- string **dumpAllValid** ()
- string **getRawArg** (int pos)
- string **getFreeArg** (int pos)

6.48.1 Detailed Description

This class provides a helper/utility to handle command line argumets.

This utility handle dependencies and conflicts arguments

Author:

fabsoft <fabsoft@gmail.com>

Examples:

argument_ex.cpp.

Definition at line 21 of file argumentshelper.h.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 n2nc::utils::args::ArgumentsHelper::ArgumentsHelper (char ** *argv*, int *argc*)

Definition at line 7 of file argumentshelper.cpp.

6.48.3 Member Function Documentation

6.48.3.1 bool n2nc::utils::args::ArgumentsHelper::validate ()

validates all.

Note:

default parameters will be setted up HERE.

Examples:

argument_ex.cpp.

Definition at line 61 of file argumentshelper.cpp.

The documentation for this class was generated from the following files:

- argumentshelper.h
- argumentshelper.cpp

6.49 n2nc::utils::args::ValidArgument Class Reference

This class represents a expected argument.

`#include <validargument.h>`

Public Member Functions

- **ValidArgument** (string name)
- string **toString** ()

Static Public Member Functions

- static ValidArgument * **find** (string name, vector< ValidArgument > &valids)

Public Attributes

- bool paramrequired
- std::string name
- std::string comment
- std::string defaultparam
- std::vector< std::string > paramlist
- std::vector< std::string > paramlist_pattern
- std::vector< std::string > dependsof
- std::vector< std::string > conflictsto

6.49.1 Detailed Description

This class represents a expected argument.

Author:

fabsoft <fabsoft@gmail.com>

Examples:

argument_ex.cpp.

Definition at line 16 of file validargument.h.

6.49.2 Member Data Documentation

6.49.2.1 bool n2nc::utils::args::ValidArgument::paramrequired

Indicates if a parameter is mandatory.

Examples:

argument_ex.cpp.

Definition at line 28 of file validargument.h.

6.49.2.2 std::string n2nc::utils::args::ValidArgument::name

The name of argument.

Definition at line 30 of file validargument.h.

6.49.2.3 std::string n2nc::utils::args::ValidArgument::comment

The comment to be displayed by ArgumentsHelper::dumpAllValid()

Examples:

argument_ex.cpp.

Definition at line 32 of file validargument.h.

6.49.2.4 std::string n2nc::utils::args::ValidArgument::defaultparam

If the param is not required and a a parameter is not supplied the assuming *defaultparam*

Examples:

argument_ex.cpp.

Definition at line 34 of file validargument.h.

6.49.2.5 std::vector<std::string> n2nc::utils::args::ValidArgument::paramlist

A list of expected parameters for this argument.

Examples:

argument_ex.cpp.

Definition at line 36 of file validargument.h.

6.49.2.6 std::vector<std::string> n2nc::utils::args::ValidArgument::paramlist_pattern

A list of expected patternized parameters for this argument.

Definition at line 38 of file validargument.h.

6.49.2.7 std::vector<std::string> n2nc::utils::args::ValidArgument::dependsof

A list of dependant arguments(name) for this argument.

Examples:

argument_ex.cpp.

Definition at line 40 of file validargument.h.

Referenced by n2nc::utils::args::Argument::checkForDependecies().

6.49.2.8 std::vector<std::string> n2nc::utils::args::ValidArgument::conflictsto

A list of conlictual arguments(name) for this argument.

Definition at line 42 of file validargument.h.

Referenced by n2nc::utils::args::Argument::checkForConflicts().

The documentation for this class was generated from the following files:

- validargument.h
- validargument.cpp

6.50 n2nc::utils::SharedOptions Class Reference

`#include <sharedoptions.h>`

Public Member Functions

- std::string **addPair** (std::string key, std::string value)
- std::string **getValue** (std::string key)

6.50.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Definition at line 12 of file sharedoptions.h.

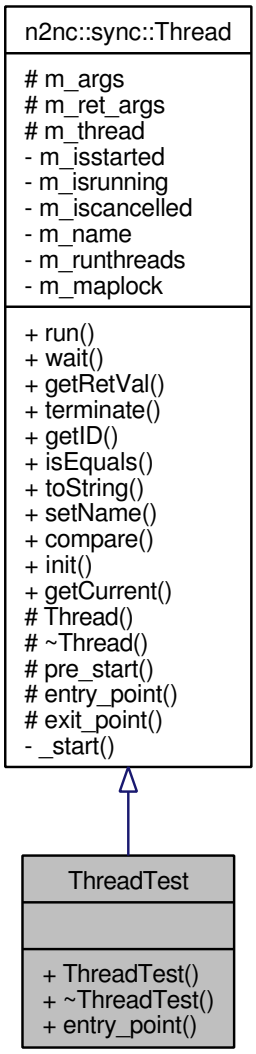
The documentation for this class was generated from the following files:

- sharedoptions.h
- sharedoptions.cpp

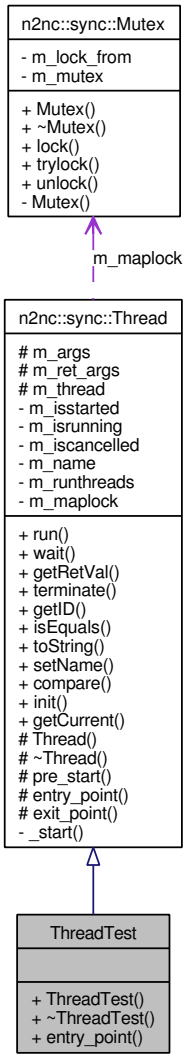
6.51 ThreadTest Class Reference

`#include <threadtest.h>`

Inheritance diagram for ThreadTest:



Collaboration diagram for ThreadTest:



Public Member Functions

- void * entry_point ()

6.51.1 Detailed Description

Author:

fabsoft <fabsoft@gmail.com>

Examples:

socket_ex.cpp.

Definition at line 13 of file threadtest.h.

6.51.2 Member Function Documentation

6.51.2.1 void * ThreadTest::entry_point () [virtual]

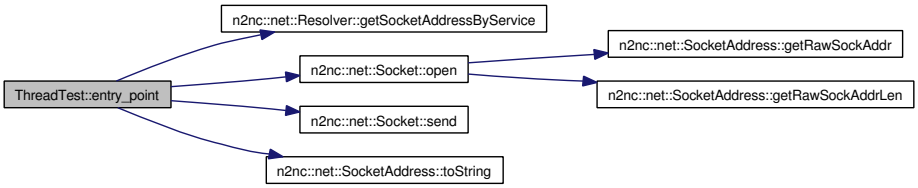
must implements thread code

Implements n2nc::sync::Thread.

Definition at line 13 of file threadtest.cpp.

References n2nc::net::Resolver::getSocketAddressByService(), n2nc::net::Socket::open(), n2nc::net::Socket::send(), and n2nc::net::SocketAddress::toString().

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- threadtest.h
- threadtest.cpp

Chapter 7

Example Documentation

7.1 argument_ex.cpp

An example of using this helper. The output is

```
 COOD
```

```
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include "nixsys.h"
#include "argumentshelper.h"

using namespace n2nc::utils::args ;
using namespace std;

int main(int argc, char *argv[])
{
  ArgumentsHelper args(argv, argc) ;
  ValidArgument* t_val ;
  t_val = args.addValid("-arg1");
  t_val->comment = "argname_comment" ;
  t_val->dependsof.push_back("-arg2") ;
  t_val->paramlist.push_back("2") ;
  t_val->paramrequired = true ;

  t_val = args.addValid("-arg2");

  t_val = args.addValid("-arg3");
  t_val->paramlist.push_back("3");
  t_val->paramlist.push_back("paramfor_arg3");
  t_val->defaultparam = "defaultparam" ;

  if (args["-arg2"])
    cerr << args["-arg2"]->param() << endl ;

  if (args["-arg3"])
    cerr << args["-arg3"]->param() << endl ;

  cerr << args.dumpAllValid() ;
  cerr << args.toString() << endl ;
  cerr << "free arg " << args.getFreeArg(0) << endl ;
  cerr << "valid?: " << args.validate() << endl ;
```

```
  return EXIT_SUCCESS;
}
```

7.2 configuration_ex.cpp

This is an example of using Configuration utility. more about this example

```
#include "configuration.h"

using namespace n2nc ;

int main(){
  Configuration conf ;
  conf.addContext( ConfigurationCtx("ctxname") ) ;
  conf["ctxname"].addOption(ConfigurationOpt("optname","optval")) ;
  return EXIT_SUCCESS ;
}
```

7.3 filter_ex.cpp

Example on how to use the Filter interface.

Filter Implementation.

Filter Header.

```
#include "filter.h"

using namespace n2nc ;
using namespace std ;

/*
  \brief Simple implemented Filter

  \see filter_dummy.cpp Implementation code.
*/
int main(){
Filter *myfilter,*myfilter1,*myfilter2; ;
  myfilter = Filter::load_filter("../filters/.libs/libdummy.0.0.0");
  myfilter1 = Filter::load_filter("../filters/.libs/libdummy");
  cerr << myfilter->toString() << endl ;
  myfilter->egress(NULL,NULL,0,NULL);
  Filter::free_filter(myfilter);
  Filter::free_filter(myfilter1);

  return EXIT_SUCCESS ;
}
```

7.4 socket_ex.cpp

A simple Socket example.

```
#include "socket.h"
#include "tcpsocket.h"
#include "resolver.h"
#include "thread.h"
#include "threadtest.h"


using namespace n2nc ;

int main(){
// std::string mys("127.0.0.1");
// std::string mys("::1");
// std::string myport("6099");

// net::Address *testaddr = new net::IP4Address(mys);
// std::cerr << testaddr->toString();

// std::string mys3("195.210.91.83");
// n2nc::net::Address *ma = n2nc::net::Address::newAddress(mys2);
// n2nc::net::Address *ma = sa->getAddress();
  net::Socket *sock = new net::TcpSocket(AF_INET6);
//   net::Socket sockcpy = *sock ;
  net::SocketAddress *sa = net::Resolver::getSocketAddressByService("::1","6099",SOCK_STREAM,AF_INET6,fals

  std::cerr << "test: " << sa->getAddress()->toString() << std::endl ;

  sock->bind(*sa);
  ((net::TcpSocket*)(sock))->setListen();
  std::cerr << "listening to: " << sa->toString() << std::endl ;

  sleep(2);

  n2nc::sync::Thread *tr = new ThreadTest();
  tr->run(NULL);

  sleep(2);

  net::Socket *consock = ((net::TcpSocket*)sock)->accept();

// system("read");
// sa->toString();
// delete sa ;
// sleep(5);
  int left = 3;
  char mybuf[100];
  while(left--){
    consock->recv(mybuf,100);
    std::cerr << mybuf << std::endl;
    sleep(1);
  }
  tr->terminate();
  sleep(2);
  delete consock ;
  delete sock ;
  delete sa ;
  return EXIT_SUCCESS ;
}
```

7.5 threads_ex.cpp

Threads example.

```
#include "nixsys.h"
#include "thread.h"
#include "semaphore.h"

using namespace std ;
using namespace n2nc ;

n2nc::sync::Semaphore g_sem(0);

void* fun1(void *arg1){
  cerr << "fun1 callled with args: " << (char*)arg1 ;
  cerr << " Thread name: " << n2nc::sync::Thread::getCurrent().toString() << endl ;
  sleep(3);
  /* NOTE the possible race condition */
  g_sem.post();

  return (void*)"fun1" ;
}

int fun1_exit(void* arg1){
  cerr << "fun1 exited with args: " << (char*)arg1 ;
  cerr << " Thread name: " << n2nc::sync::Thread::getCurrent().toString() << endl ;
}

class myclass1 : public n2nc::sync::Thread {
    virtual void* entry_point(){
      cerr << "myclass1 callled with args: " << (char*)this->m_args ;
      cerr << " Thread name: " << n2nc::sync::Thread::getCurrent().toString() << endl ;
      sleep(6);
      return (void*)"myclass1" ;
    }
    virtual int exit_point(){
      /* NOTE HERE we avoid race condition because the thread
      has already terminated its routine and saved returns values. */
      g_sem.post();
    }

};

int main(int argc, char *argv[]){
  void *myretval1,*myretval2 ;
  n2nc::sync::Thread::init();
  n2nc::sync::Thread *t1 = new n2nc::sync::InlineThread(&fun1);
  n2nc::sync::Thread *t2 = new myclass1() ;
  n2nc::sync::Thread *t3 = new n2nc::sync::InlineThread(&fun1,&fun1_exit);
  t1->setName("fun1 thread name");
  t2->setName("myclass1 thread name");
  t3->setName("fun1 thread name, with exit point callback");
  t1->run((void*)"args to function fun1");
  t2->run((void*)"args to class myclass1");
  t3->run((void*)"args to function fun1 with callback");
  /* wait the 2 threads termination through a global semaphore */
  g_sem.wait();
  g_sem.wait();
  g_sem.wait();
/*  t1->wait(&myretval1);
  t2->wait(&myretval2);*/
  cerr << (char*)t1->getRetVal() << endl ;
  cerr << (char*)t2->getRetVal() << endl ;
  cerr << (char*)t3->getRetVal() << endl ;
  return EXIT_SUCCESS;
}
```

Index