NSB

0.1.0

Generated by Doxygen 1.14.0

1 Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

2 Hierarchical Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Namespace Documentation

5.1 nsb_client Namespace Reference

Classes

- class CommunicationInterface

  Base class for communication interfaces.
- class NSBAppClient

  NSB Application Client interface.
- class NSBClient

  NSB client base class.
- class NSBSimClient

  NSB Simulator Client interface.
- class SocketInterface

  Socket interface for client-server communication.

Functions

- test_ping ()
- test_lifecycle ()

Variables

- int SERVER_CONNECTION_TIMEOUT = 10

  Maximum time a client will wait to connect to the daemon.
- int DAEMON_RESPONSE_TIMEOUT = 600

  Maximum time a client will wait to get a response from the daemon.
- int RECEIVE_BUFFER_SIZE = 4096

  Buffer size when receiving data.
- int SEND_BUFFER_SIZE = 4096

  Buffer size when sending data.

5.1.1 Function Documentation

test_lifecycle()

```
nsb_client.test_lifecycle ()
```

test_ping()

```
nsb_client.test_ping ()
```

5.1.2 Variable Documentation

DAEMON_RESPONSE_TIMEOUT

```
int nsb_client.DAEMON_RESPONSE_TIMEOUT = 600
```

Maximum time a client will wait to get a response from the daemon.

RECEIVE_BUFFER_SIZE

```
int nsb_client.RECEIVE_BUFFER_SIZE = 4096
```

Buffer size when receiving data.

SEND_BUFFER_SIZE

```
int nsb_client.SEND_BUFFER_SIZE = 4096
```

Buffer size when sending data.

SERVER_CONNECTION_TIMEOUT

```
int nsb_client.SERVER_CONNECTION_TIMEOUT = 10
```

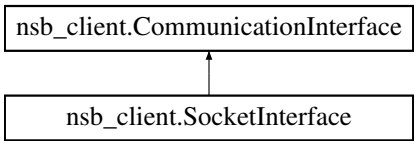
Maximum time a client will wait to connect to the daemon.

6 Class Documentation

6.1 nsb_client.CommunicationInterface Class Reference

Base class for communication interfaces.

Inheritance diagram for nsb_client.CommunicationInterface:



6.1.1 Detailed Description

Base class for communication interfaces.

As NSB is expected to support different communication paradigms and protocols – including sockets, RabbitMQ, and Websockets – we plan to provide various different communication interfaces with the same basic functions. The SocketInterface class should be used as an example to develop other interfaces.

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

- nsb_client.py

6.2 MessageEntry Struct Reference

Message storage struct.

```
#include <nsb_daemon.h>
```

Public Member Functions

- MessageEntry ()

  Blank constructor.
- MessageEntry (std::string src, std::string dest, std::string data)

  Populated constructor.

Public Attributes

- std::string source

  The source identifier.
- std::string destination

  The destination identifier.
- std::string payload

  The payload as a bytestring, but in const char∗ form.

6.2.1 Detailed Description

Message storage struct.

This struct contains source and destination information and the payload and is intended to be used to store messages in the daemon's transmission and reception buffers.

6.2.2 Constructor & Destructor Documentation

MessageEntry() [1/2]

```
MessageEntry::MessageEntry ()  [inline]
```

Blank constructor.

MessageEntry() [2/2]

```
MessageEntry::MessageEntry (
            std::string src,
            std::string dest,
            std::string data)  [inline]
```

Populated constructor.

6.2.3 Member Data Documentation

destination

```
std::string MessageEntry::destination
```

The destination identifier.

payload

```
std::string MessageEntry::payload
```

The payload as a bytestring, but in const char∗ form.

source

```
std::string MessageEntry::source
```

The source identifier.

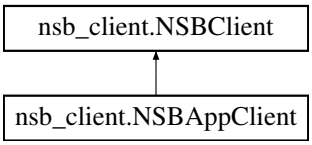
The documentation for this struct was generated from the following file:

- nsb_daemon.h

6.3 nsb_client.NSBAppClient Class Reference

NSB Application Client interface.

Inheritance diagram for nsb_client.NSBAppClient:



Public Member Functions

- __init__ (self, str identifier, str server_address, int server_port)
  Constructs the NSB Application Client interface.
- send (self, str dest_id, bytes payload)
  Sends a payload to the specified destination via NSB.
- receive (self, str|None dest_id=None)
  Receives a payload that has been addressed to the client via NSB.

Public Member Functions inherited from nsb_client.NSBClient

- __init__ (self, str server_address, int server_port)
  Base constructor for NSB Clients.
- ping (self, int timeout=DAEMON_RESPONSE_TIMEOUT)
  Pings the server.
- exit (self)
  Instructs server and self to exit and shutdown.

Public Attributes

- logger = logging.getLogger(f"{self._id} (app)")

Public Attributes inherited from nsb_client.NSBClient

- comms = SocketInterface(server_address, server_port)

Protected Attributes

- _id = identifier

6.3.1 Detailed Description

NSB Application Client interface.

This client provides the high-level NSB interface to send and receive messages via NSB by communicating to the daemon.

6.3.2 Constructor & Destructor Documentation

__init__()

```
nsb_client.NSBAppClient.__init__ (
              self,
             str identifier,
             str server_address,
             int server_port)
```

Constructs the NSB Application Client interface.

This method uses the base NSBClient's constructor, which initializes a network interface to connect and communicate with the NSB daemon. It also an identifier that should correspond to the identifier used in the NSB system.

Parameters

identifier	The identifier for this NSB application client, which should correspond to the identifier in NSB and simulator.
server_address	The address of the NSB daemon.
server_port	The port of the NSB daemon.

6.3.3 Member Function Documentation

receive()

```
nsb_client.NSBAppClient.receive (
              self,
             str|None dest_id = None)
```

Receives a payload that has been addressed to the client via NSB.

If the destination is specified, it will receive a payload for that destination. This method creates an NSB RECEIVE message with the appropriate information and payload and sends it to the daemon. It will then get a response that either contains a MESSAGE code and carries the retrieved payload or contains a NO_MESSAGE code. If a message is found, the entire NSB message is returned to provide access to the metadata.

Parameters

dest_id	The identifier of the destination NSB client. The default None value will automatically assume the destination is self.

Returns

nsb_pb2.nsbm|None The NSB message containing the received payload and metadata if a message is found, otherwise None.

send()

```
nsb_client.NSBAppClient.send (
              self,
             str dest_id,
             bytes payload)
```

Sends a payload to the specified destination via NSB.

This method creates an NSB SEND message with the appropriate information and payload and sends it to the daemon. It does not expect a response from the daemon.

Parameters

dest_id	The identifier of the destination NSB client.
payload	The payload to send to the destination.

6.3.4 Member Data Documentation

_id

```
nsb_client.NSBAppClient._id = identifier  [protected]
```

logger

```
nsb_client.NSBAppClient.logger = logging.getLogger(f"{self._id} (app)")
```

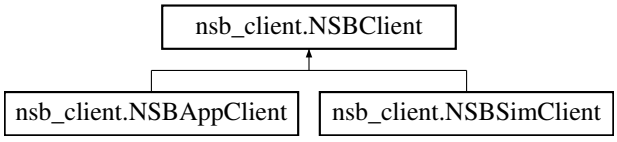
The documentation for this class was generated from the following file:

- nsb_client.py

6.4 nsb_client.NSBClient Class Reference

NSB client base class.

Inheritance diagram for nsb_client.NSBClient:



Public Member Functions

- __init__ (self, str server_address, int server_port)

  Base constructor for NSB Clients.
- ping (self, int timeout=DAEMON_RESPONSE_TIMEOUT)

  Pings the server.
- exit (self)

  Instructs server and self to exit and shutdown.

Public Attributes

- comms = SocketInterface(server_address, server_port)

6.4.1 Detailed Description

NSB client base class.

This class serves as the base for the implemented clients (AppClient and SimClient) will be built on. It provides basic methods and shared operation methods.

6.4.2 Constructor & Destructor Documentation

__init__()

```
nsb_client.NSBClient.__init__ (
              self,
             str server_address,
             int server_port)
```

Base constructor for NSB Clients.

Sets the communications module to the desired network interface (currently SocketInterface) to connect to the daemon server.

Parameters

server_address	The address of the NSB daemon.
server_port	The port of the NSB daemon.

See also

SocketInterface

6.4.3 Member Function Documentation

exit()

```
nsb_client.NSBClient.exit (
              self)
```

Instructs server and self to exit and shutdown.

This method sends an EXIT message to the server before deleting itself.

ping()

```
nsb_client.NSBClient.ping (
               self,
              int timeout = DAEMON_RESPONSE_TIMEOUT)
```

Pings the server.

This method sends a PING message to the server and awaits a response. It returns whether or not the server is reachable.

Parameters

timeout	Maximum time to wait for a response from the server. Default is set to DAEMON_RESPONSE_TIMEOUT.

Returns

bool True if the server is reachable and responds correctly, False otherwise.

6.4.4 Member Data Documentation

comms

```
nsb_client.NSBClient.comms = SocketInterface(server_address, server_port)
```

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

- nsb_client.py

6.5 NSBDaemon Class Reference

```
#include <nsb_daemon.h>
```

Public Member Functions

- NSBDaemon (int s_port)

  Construct a new NSBDaemon::NSBDaemon object.
- ~NSBDaemon ()

  Destroy the NSBDaemon::NSBDaemon object.
- void start ()

  Start the NSB Daemon.
- void stop ()

  Stops the NSB Daemon.
- bool is_running () const

  Checks if the server is running.

Private Member Functions

- void start_server (int port)

  Start the socket-connected server within the NSB Daemon.
- void handle_message (int fd, std::vector< char > message)

  A multiplexer to parse messages and redirect them to handlers.
- void handle_ping (nsb::nsbm *incoming_msg, nsb::nsbm *outgoing_msg, bool *response_required)

  Handles PING messages.
- void handle_send (nsb::nsbm *incoming_msg, nsb::nsbm *outgoing_msg, bool *response_required)

  Handles SEND messages from the NSB Application Client.
- void handle_fetch (nsb::nsbm *incoming_msg, nsb::nsbm *outgoing_msg, bool *response_required)

  Handles FETCH messages from the NSB Simulator Client.
- void handle_post (nsb::nsbm *incoming_msg, nsb::nsbm *outgoing_msg, bool *response_required)

  Handles POST messages from the NSB Simulator Client.
- void handle_receive (nsb::nsbm *incoming_msg, nsb::nsbm *outgoing_msg, bool *response_required)

  Handles RECEIVE messages from the NSB Application Client.

Private Attributes

- std::atomic< bool > running

  A flag set to indicate daemon server status.
- int server_port

  The server port accessible to client connections.
- std::list< MessageEntry > tx_buffer

  Transmission buffer to store sent payloads waiting to be fetched.
- std::list< MessageEntry > rx_buffer

  Reception buffer to store posted payloads waiting to be received.

6.5.1 Constructor & Destructor Documentation

NSBDaemon()

```
NSBDaemon::NSBDaemon (
            int s_port)
```

Construct a new NSBDaemon::NSBDaemon object.

This method initializes attributes and verifies the Protobuf version.

Parameters

s_port	The port that NSB clients will connect to.

~NSBDaemon()

```
NSBDaemon::~NSBDaemon ()
```

Destroy the NSBDaemon::NSBDaemon object.

This method will check to see if the server is still running and stop it if necessary. It will then shut down the Protobuf library.

6.5.2 Member Function Documentation

handle_fetch()

```
void NSBDaemon::handle_fetch (
            nsb::nsbm * incoming_msg,
            nsb::nsbm * outgoing_msg,
            bool * response_required)  [private]
```

Handles FETCH messages from the NSB Simulator Client.

This method first creates a blank MessageEntry. If a source has been specified, it will search the transmission buffer for a message with that source, either setting the blank MessageEntry to the found entry if the query was resolved or leaving it blank if not found. If a source has not been specified, the top MessageEntry of the buffer will be popped off and used; otherwise, if the buffer is empty, the MessageEntry will be left blank.

If a message was found, a NSB FETCH message indicating MESSAGE will be sent with the metadata and payload. Otherwise, a NSB FETCH message indicating NO_MESSAGE will be sent back to the client.

Parameters

incoming_msg	The incoming message that is being handled.
outgoing_msg	A template message that can be used if a response is required.
response_required	Whether or not a response is required and the outgoing message will be sent back to the client.

See also

MessageEntry

handle_message()

```
void NSBDaemon::handle_message (
            int fd,
            std::vector< char > message)  [private]
```

A multiplexer to parse messages and redirect them to handlers.

This method is invoked by the server (in the start_server() method) to handle an incoming message. It parses the message using Protobuf, and then redirects the incoming message and a template outgoing message (in case a response is necessary) to one of the operation-specific handlers.

If the operation is not understood, the server will respond with a negative PING message.

Parameters

fd	The file descriptor of the client connection.
message	The incoming message to parse and handle.

See also

start_server()
handle_ping()
handle_send()
handle_fetch()
handle_post()
handle_receive()

handle_ping()

```
void NSBDaemon::handle_ping (
            nsb::nsbm * incoming_msg,
            nsb::nsbm * outgoing_msg,
            bool * response_required)  [private]
```

Handles PING messages.

Since the PING has been received, it can be assumed to be successful. As such this method populates the outgoing message as an NSB PING message indicating success.

Parameters

incoming_msg	The incoming message that is being handled.
outgoing_msg	A template message that can be used if a response is required.
response_required	Whether or not a response is required and the outgoing message will be sent back to the client.

handle_post()

```
void NSBDaemon::handle_post (
            nsb::nsbm * incoming_msg,
            nsb::nsbm * outgoing_msg,
            bool * response_required)  [private]
```

Handles POST messages from the NSB Simulator Client.

This method handles POST messages by parsing the incoming message and storing the source, destination, and payload as a MessageEntry. The new MessageEntry will be pushed back in the reception buffer where it will be ready to be received by the NSB Application Client.

Parameters

incoming_msg	The incoming message that is being handled.
outgoing_msg	A template message that can be used if a response is required.
response_required	Whether or not a response is required and the outgoing message will be sent back to the client.

See also

MessageEntry

handle_receive()

handle_receive()

```
void NSBDaemon::handle_receive (
            nsb::nsbm * incoming_msg,
            nsb::nsbm * outgoing_msg,
            bool * response_required)  [private]
```

Handles RECEIVE messages from the NSB Application Client.

This method first creates a blank MessageEntry. If a destination has been specified, it will search the reception buffer for a message with that destination, either setting the blank MessageEntry to the found entry if the query was resolved or leaving it blank if not found. If a destination has not been specified, the top MessageEntry of the buffer will be popped off and used; otherwise, if the buffer is empty, the MessageEntry will be left blank.

If a message was found, a NSB RECEIVE message indicating MESSAGE will be sent with the metadata and payload. Otherwise, a NSB RECEIVE message indicating NO_MESSAGE will be sent back to the client.

Parameters

incoming_msg	The incoming message that is being handled.
outgoing_msg	A template message that can be used if a response is required.
response_required	Whether or not a response is required and the outgoing message will be sent back to the client.

See also

MessageEntry

handle_send()

```
void NSBDaemon::handle_send (
            nsb::nsbm * incoming_msg,
            nsb::nsbm * outgoing_msg,
            bool * response_required)  [private]
```

Handles SEND messages from the NSB Application Client.

This method handles SEND messages by parsing the incoming message and storing the source, destination, and payload as a MessageEntry. The new MessageEntry will be pushed back in the transmission buffer where it will be ready to be fetched by the NSB Simulator Client.

Parameters

incoming_msg	The incoming message that is being handled.
outgoing_msg	A template message that can be used if a response is required.
response_required	Whether or not a response is required and the outgoing message will be sent back to the client.

See also

MessageEntry

handle_fetch()

is_running()

```
bool NSBDaemon::is_running () const
```

Checks if the server is running.

Returns

true if the server is running, false otherwise.

false if the server is not running.

start()

```
void NSBDaemon::start ()
```

Start the NSB Daemon.

This method will launch the server at the server port using the start_server method.

See also

start_server(int port)

start_server()

```
void NSBDaemon::start_server (
            int port)  [private]
```

Start the socket-connected server within the NSB Daemon.

This is the main servicing method that runs for the lifetime of the NSB Daemon. It opens a multiple connection-enabled server and maintains persistent connections as communication channels with each NSB client that connects to it. New connections are managed through an updating vector of file descriptors where each represents a different connection. When messages come in from existing connections, they will be passed onto the handle_↩ message method.

This method is invoked by the start() method.

Parameters

port	The port that will be accessible for clients to connect.

See also

start()

handle_message()

stop()

```
void NSBDaemon::stop ()
```

Stops the NSB Daemon.

Checks if server is running and stops it, resulting in the daemon shutting down.

6.5.3 Member Data Documentation

running

```
std::atomic<bool> NSBDaemon::running  [private]
```

A flag set to indicate daemon server status.

rx_buffer

```
std::list<MessageEntry> NSBDaemon::rx_buffer  [private]
```

Reception buffer to store posted payloads waiting to be received.

See also

MessageEntry
handle_post()
handle_receive()

server_port

```
int NSBDaemon::server_port  [private]
```

The server port accessible to client connections.

tx_buffer

```
std::list<MessageEntry> NSBDaemon::tx_buffer  [private]
```

Transmission buffer to store sent payloads waiting to be fetched.

See also

MessageEntry
handle_send()
handle_fetch()

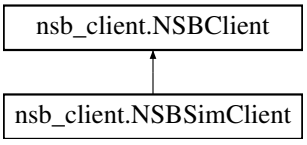
The documentation for this class was generated from the following files:

- nsb_daemon.h
- nsb_daemon.cc

6.6 nsb_client.NSBSimClient Class Reference

NSB Simulator Client interface.

Inheritance diagram for nsb_client.NSBSimClient:



Public Member Functions

- __init__ (self, str server_address, int server_port)

  Constructs the NSB Simulator Client interface.
- fetch (self, str|None src_id=None)

  Fetches a payload that needs to be sent over the simulated network.
- post (self, str src_id, str dest_id, bytes payload, bool success=True)

  Posts a payload to the specified destination via NSB.

Public Member Functions inherited from nsb_client.NSBClient

- __init__ (self, str server_address, int server_port)

  Base constructor for NSB Clients.
- ping (self, int timeout=DAEMON_RESPONSE_TIMEOUT)

  Pings the server.
- exit (self)

  Instructs server and self to exit and shutdown.

Public Attributes

- logger = logging.getLogger("(SimClient)")

Public Attributes inherited from nsb_client.NSBClient

- comms = SocketInterface(server_address, server_port)

6.6.1 Detailed Description

NSB Simulator Client interface.

This client provides the high-level NSB interface to fetch and post messages via NSB by communicating to the daemon.

6.6.2 Constructor & Destructor Documentation

__init__()

```
nsb_client.NSBSimClient.__init__ (
              self,
             str server_address,
             int server_port)
```

Constructs the NSB Simulator Client interface.

This method uses the base NSBClient's constructor, which initializes a network interface to connect and communicate with the NSB daemon.

Parameters

server_address	The address of the NSB daemon.
server_port	The port of the NSB daemon.

6.6.3 Member Function Documentation

fetch()

```
nsb_client.NSBSimClient.fetch (
              self,
             str|None src_id = None)
```

Fetches a payload that needs to be sent over the simulated network.

If the source is specified, it will try and fetch a payload for that source. This method creates an NSB FETCH message with the appropriate information and payload and sends it to the daemon. It will then get a response that either contains a MESSAGE code and carries the fetched payload or contains a NO_MESSAGE code. If a message is found, the entire NSB message is returned to provide access to the metadata.

Parameters

src_id	The identifier of the targe source. The default None value will result in fetching the most recent message, regardless of source.

Returns

nsb_pb2.nsbm|None The NSB message containing the fetched payload and metadata if a message is found, otherwise None.

post()

```
nsb_client.NSBSimClient.post (
              self,
             str src_id,
             str dest_id,
             bytes payload,
             bool success = True)
```

Posts a payload to the specified destination via NSB.

This is intended to be used when a payload is finished being processed (either successfully delivered or dropped) and the simulator client needs to hand it off back to NSB. This method creates an NSB SEND message with the appropriate information and payload and sends it to the daemon.

Parameters

src_id	The identifier of the source NSB client.
dest_id	The identifier of the destination NSB client.
payload	The payload to post to the destination.
success	Whether the post was successful or not. If False, it will set the OpCode to NO_MESSAGE.

6.6.4 Member Data Documentation

logger

```
nsb_client.NSBSimClient.logger = logging.getLogger("(SimClient)")
```

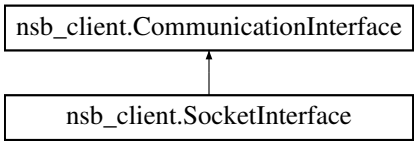
The documentation for this class was generated from the following file:

- nsb_client.py

6.7 nsb_client.SocketInterface Class Reference

Socket interface for client-server communication.

Inheritance diagram for nsb_client.SocketInterface:



Public Member Functions

- __init__ (self, str server_address, int server_port)
  Constructor for the NSBClient class.
- __del__ (self)
  Closes connection to the server.

Public Attributes

- server_addr = server_address
- server_port = server_port
- logger = logging.getLogger("NSBClient")
- conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

Protected Member Functions

- _configure (self)
  Configures the socket connection with appropriate options.
- _connect (self, int timeout=SERVER_CONNECTION_TIMEOUT)
  Connects to the daemon with the stored server address and port.
- _close (self)
  Healthily closes the socket connection.
- _send_msg (self, bytes message)
  Sends a message to the server.
- _recv_msg (self, int|None timeout=None)
  Sends a message to the server.

6.7.1 Detailed Description

Socket interface for client-server communication.

This class implements aocket interface network to facilitate network communication between NSB clients and the server. This can be used as a template to develop other interfaces for client communication, which must define the same methods with the same arguments as done in this class.

6.7.2 Constructor & Destructor Documentation

__init__()

```
nsb_client.SocketInterface.__init__ (
              self,
             str server_address,
             int server_port)
```

Constructor for the NSBClient class.

Sets the address and port of the server at the NSB daemon before connecting to the server.

Parameters

server_address	The address of the NSB daemon.
server_port	The port of the NSB daemon.

See also

nsb_client.SocketInterface._connect(timeout)

__del__()

```
nsb_client.SocketInterface.__del__ (
              self)
```

Closes connection to the server.

See also

_close()

6.7.3 Member Function Documentation

_close()

```
nsb_client.SocketInterface._close (
              self)  [protected]
```

Healthily closes the socket connection.

Attempts to shutdown the socket, then closes.

_configure()

```
nsb_client.SocketInterface._configure (
              self)  [protected]
```

Configures the socket connection with appropriate options.

This method is called by the _connect() method. It configures the sockets to work with the multi-connection server at the daemon with lower latency.

See also

_connect()

_connect()

```
nsb_client.SocketInterface._connect (
              self,
             int timeout = SERVER_CONNECTION_TIMEOUT)  [protected]
```

Connects to the daemon with the stored server address and port.

This method uses the _configure() method to configure the socket and then attempts to connect to the daemon.

Parameters

timeout	Maximum time in seconds to wait to connect to the daemon.

Exceptions

TimeoutError	Raised if the connection to the server times out after the specified timeout period.

_recv_msg()

```
nsb_client.SocketInterface._recv_msg (
              self,
             int|None timeout = None)  [protected]
```

Sends a message to the server.

This method uses selectors to wait for the socket to be ready before sending SEND_BUFFER SIZE bytes at a time, making it non-blocking compliant.

Parameters

timeout	Maximum time in seconds to wait for a response from the server. If None, it will wait indefinitely.

_send_msg()

```
nsb_client.SocketInterface._send_msg (
              self,
             bytes message)  [protected]
```

Sends a message to the server.

This method uses selectors to wait for the socket to be ready before sending SEND_BUFFER SIZE bytes at a time, making it non-blocking compliant.

Parameters

message	The message to send to the server.

Exceptions

RuntimeError	Raised if the socket connection is broken or if the socket is not ready to send.

6.7.4 Member Data Documentation

conn

```
nsb_client.SocketInterface.conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

logger

```
nsb_client.SocketInterface.logger = logging.getLogger("NSBClient")
```

server_addr

```
nsb_client.SocketInterface.server_addr = server_address
```

server_port

```
nsb_client.SocketInterface.server_port = server_port
```

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

- nsb_client.py

7 File Documentation

7.1 nsb_client.py File Reference

Classes

- class nsb_client.CommunicationInterface
  Base class for communication interfaces.
- class nsb_client.SocketInterface
  Socket interface for client-server communication.
- class nsb_client.NSBClient
  NSB client base class.
- class nsb_client.NSBAppClient
  NSB Application Client interface.
- class nsb_client.NSBSimClient
  NSB Simulator Client interface.

Namespaces

- namespace nsb_client

Functions

- nsb_client.test_ping ()
- nsb_client.test_lifecycle ()

Variables

- int nsb_client.SERVER_CONNECTION_TIMEOUT = 10

  Maximum time a client will wait to connect to the daemon.
- int nsb_client.DAEMON_RESPONSE_TIMEOUT = 600

  Maximum time a client will wait to get a response from the daemon.
- int nsb_client.RECEIVE_BUFFER_SIZE = 4096

  Buffer size when receiving data.
- int nsb_client.SEND_BUFFER_SIZE = 4096

  Buffer size when sending data.

7.2 nsb_daemon.cc File Reference

```
#include "nsb_daemon.h"
```

Functions

- int main ()

  Main process to run the NSB Daemon.

7.2.1 Function Documentation

main()

```
int main ()
```

Main process to run the NSB Daemon.

Returns

int

7.3 nsb_daemon.h File Reference

```
#include <string>
#include <list>
#include <vector>
#include <map>
#include <array>
#include <atomic>
#include <thread>
#include <iostream>
#include <cstdio>
#include <format>
#include <signal.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <unistd.h>
#include <fcntl.h>
#include <sqlite3.h>
#include "nsb.pb.h"
```

Classes

- struct MessageEntry

  Message storage struct.
- class NSBDaemon

Variables

- int MAX_BUFFER_SIZE = 4096

  The maximum buffer size for sending and receiving messages.

7.3.1 Variable Documentation

MAX_BUFFER_SIZE

```
int MAX_BUFFER_SIZE = 4096
```

The maximum buffer size for sending and receiving messages.

7.4 nsb_daemon.h

Go to the documentation of this file.

```
00001 // nsb_daemon.h
00002
00003 #ifndef NSB_DAEMON_H
00004 #define NSB_DAEMON_H
00005
00006 #include <string>
00007 #include <list>
00008 #include <vector>
00009 #include <map>
00010 #include <array>
00011 // Thread libraries.
00012 #include <atomic>
00013 #include <thread>
00014 // I/O libraries.
00015 #include <iostream>
00016 #include <cstdio>
00017 #include <format>
00018 #include <signal.h>
00019 // Networking libraries.
00020 #include <arpa/inet.h>
00021 #include <netinet/in.h>
00022 #include <sys/socket.h>
00023 #include <unistd.h>
00024 #include <fcntl.h>
00025 // Data.
00026 #include <sqlite3.h>
00027
00028 #include "nsb.pb.h"
00029
00030
00032 int MAX_BUFFER_SIZE = 4096;
00033
00042 struct MessageEntry {
00044     std::string source;
00046     std::string destination;
00048     std::string payload;
00049     // Constructors.
00051     MessageEntry() : source(""), destination(""), payload("") {}
00053     MessageEntry(std::string src, std::string dest, std::string data)
00054         : source(std::move(src)), destination(std::move(dest)), payload(std::move(data)) {}
00055 };
00056
00057 class NSBDaemon {
00058 public:
00066     NSBDaemon(int s_port);
00073     ~NSBDaemon();
00082     void start();
00089     void stop();
00096     bool is_running() const;
00097
00098 private:
00100     std::atomic<bool> running;
00102     int server_port;
00110     std::list<MessageEntry> tx_buffer;
00118     std::list<MessageEntry> rx_buffer;
00137     void start_server(int port);
00159     void handle_message(int fd, std::vector<char> message);
00160
00161     /* Operation-specific handlers. */
00162
00176     void handle_ping(nsb::nsbm* incoming_msg, nsb::nsbm* outgoing_msg, bool* response_required);
00194     void handle_send(nsb::nsbm* incoming_msg, nsb::nsbm* outgoing_msg, bool* response_required);
00217     void handle_fetch(nsb::nsbm* incoming_msg, nsb::nsbm* outgoing_msg, bool* response_required);
00235     void handle_post(nsb::nsbm* incoming_msg, nsb::nsbm* outgoing_msg, bool* response_required);
00258     void handle_receive(nsb::nsbm* incoming_msg, nsb::nsbm* outgoing_msg, bool* response_required);
00259 };
00260
00261 #endif // NSB_DAEMON_H
```

Index