MiscSocket

Inherits From:	Object
Declared In:	MiscSocket.h

Class Description

This is an abstract class which encapsulates some basic operations on sockets, and provides a framework for subclasses which manage a particular domain or type of socket.

A socket is a potential end point for network or process communication, and has two important properties with which it is created: its domain and its type. The domain specifies where the communication is to be done. Most often, this is either the UNIX or the Internet domain. Sockets in the UNIX domain are used for inter-process communication. Sockets in the Internet domain are used for communication between hosts on the Internet.

The type of a socket is the type of transport service that the socket will provide. There are three common types of transport service: raw, datagram, and stream. Raw service provides no special handling of the data read from or written to a socket. Datagram service packages data written to a socket into an independent message, which is then read as an individual message at the other end.[2] In the current implementation, datagram service is provided via the User Datagram Protocol. UDP does not provide reliable transport, so messages may be lost, duplicated, delivered in a different order from which they were sent, or modified in transport. Reliability, if desired, must be provided by the application; the programmer must handle lost, duplicated, or out of order messages[3]. The third type of service, stream service, provides a stream of data to and from the socket, similar to the UNIX file paradigm as a stream of bytes. Stream service is provided via the Transmission Control Protocol, TCP, which, unlike UDP, provides reliable data transmission. The remote end of a connection will read the exact sequence of bytes written to the local end of the connection (in the absence of network or host failures). However, unlike UDP, there are no message boundaries, but rather a continuous stream of data.

Instances of the class are reusable, in that they may be sent multiple initialization messages. This may allow an application to avoid some memory allocation and deallocation in some cases. Note however that the initialization method **initDomain:type:** frees the instance on error.

Methods implemented in this class generally return self on success and nil upon failure. When nil is returned, the value of the *errno* global variable can be examined for the cause; values that this variable may take on are described in the header file <sys/errno.h> and the UNIX manual page *intro*(2). If the method is successful, the previous value of *errno* is preserved.

[2] There are system- and network-imposed limits on the size of datagrams. Applications limiting datagram size to IP_MSS - sizeof(struct udpiphdr), or 548, bytes should have no problem with this.

[3] Each UDP message has a header, which contains a checksum. If the checksum check, when a message arrives at the UDP software layer of a host, fails, the message is dropped. The application

will never see a UDP message which has a bad checksum (i.e., has been modified in transport).

Instance Variables

int **sock**	Socket descriptor
int **domain**	Socket's domain
int **type**	Socket type

Method Types

Initializing instances	- close
	- init
	- initDomain:type:
Copying instances	- copyFromZone:

Freeing instances	- free
Accessing properties	- domain - isClosed - socket - type

Instance Methods

close

- close

Places the receiver in the closed state. This method is a synonym for **init**, and can be used when *close* better describes what is desired than *init*. Returns self.

copyFromZone:

- **copyFromZone:**(NXZone *)zone

Returns a copy of the receiver. If the socket is not closed, the copy gets a duplicate socket descriptor as well. Thus, the copy can be closed or freed without interfering with the original object. Note, however, that the socket descriptors of both objects refer to the same structures in the kernel, the same connection end point; only one of the objects will read a particular chunk of data from the connection±the data is not duplicated for each descriptor. Care should be taken that the two objects do not interfere with one another. If the socket descriptor cannot be duplicated, the copy is freed and nil is returned. The global *errno* may contain the following value: EMFILE (the process is out of descriptors).

domain

- (int)**domain**

Returns the domain identifier of the socket if it is not closed, or -1 if it is. The domain identifier is one of the constants PF_INET, PF_UNIX, etc. defined in the header file <sys/socket.h>.

free
- free

Closes the socket if it is open and frees the instance. Returns nil.

init
- init

Places the instance in the closed (initialized) state, and returns self. This method may be applied multiple times to an instance, so that instances may be reused. This method is the designated initializer of MiscSocket instances. Subclasses should implement their own version of this method (beginning with [super init]) to initialize its instance variables. To maintain the multiple-initialization property, the method should check the value of instance variables and free allocated objects and memory as part of the initialization, and extended **init-** methods (**initDomain:type:**, for example) in the class should call [self init] first thing in the method.

initDomain:type:

- initDomain:(int)aDomain **type:**(int)aType

Initializes the instance to the closed state, then "opens" it by creating a new socket for the instance, in the domain and of the type specified. The socket is "open", but is not usable for communication activity. The parameter *aDomain* is one of the domain (protocol family) constants from the header file <sys/socket.h>, typically PF_UNIX or PF_INET. *aType* is one of the type constants defined in this class's header file: MiscSOCK_RAW, MiscSOCK_DGRAM, or MiscSOCK_STREAM. MiscSOCK_RAW specifies no special transport handling for the connection; only the superuser can create raw sockets. MiscSOCK_DGRAM specifies datagram-oriented transport, and MiscSOCK_STREAM specifies stream-oriented transport. In the current implementation, datagram and stream service is provided via UDP and TCP, respectively. Returns self on success. If the socket cannot be created, the instance is freed and nil is returned. The global *errno* may contain the following values: EPROTONOSUPPORT (domain/type combination not supported), ESOCKTNOSUPPORT (invalid type value), EMFILE (the process is out of descriptors), ENFILE (the system descriptor table is full), EACCES (permission is denied), ENOBUFS (system lacks resources).

isClosed

  - (BOOL)**isClosed**

Returns YES if the socket is in the closed state, NO otherwise.

socket

  - (int)**socket**

Returns the socket descriptor for the socket, or -1 if the receiver is closed.

type

  - (int)**type**

Returns the type of the socket if it is not closed, or -1 if it is. The type is one of the constants MiscSOCK_RAW, MiscSOCK_DGRAM, or MiscSOCK_STREAM defined in the header file for this class.