X_SOCKET contains
> **- stats structure**
> **- error status**

X_STATS contains
> **- cumulative counters**
> **- average performance data**
> **- incremental performance data**

X_SOCKET * x_socket(int socktype, X_SOCKET *skt)

> Return a new socket of type *socktype* which is SOCK_STREAM or SOCK_DGRAM. This call
> will always return a socket structure, however if the socket creation fails the socket structure will
> record an error status which can be obtained through x_sockstatus(). If *skt* is NULL, space for the
> socket structure is allocated using malloc, otherwise the supplied structure is initialized and
> returned.

> **Errors**
>> X_BADTYPE   invalid socktype
>> X_IMPL   underlying socket descriptor failure

int   x_close(X_SOCKET *skt)

> Close the socket *skt*. Returns 0 on success and -1 on error. This call returns -1 immediately
> without changing the error status of *skt* if *skt* is marked with an error on entry. This call does not
> free the socket structure.

> **Errors**
>> X_BADSOCK   bad X_SOCKET
>> X_IMPL   failure closing underlying socket descriptor

int   x_connect(X_SOCKET *skt, const struct sockaddr *addr, int len)

> For a SOCK_DGRAM socket *skt* this call specifies the peer address, defined by *addr* and *len*, to
> be used in sending and receiving datagrams. For a SOCK_STREAM socket *skt* this call attempts
> a connection operation with the peer defined by *addr* and *len*. Returns 0 on success and -1 on
> error. This call returns -1 immediately without changing the error status of *skt* if *skt* is marked
> with an error on entry.

> **Errors**
>> X_BADSOCK   bad X_SOCKET
>> X_TIMEOUT   connection attempt timedout
>> X_REFUSED   connection attempt refused
>> X_NOCONN   socket is not able to perform a connect

int   x_bind(X_SOCKET *skt, const struct sockaddr *addr, int len)

> Associate the socket *skt* with the specified *addr*. Returns 0 on success and -1 on error. This call
> returns -1 immediately without changing the error status of *skt* if *skt* is marked with an error on

entry.

Errors

X_BADSOCK	bad X_SOCKET
X_NOBIND	socket is not able to perform a bind because it is bound
X_BADADDR	address is unavailable or in use

int   **x_listen(X_SOCKET *skt, int backlog)**

Configures socket *skt* to begin listening for incoming connection requests using *backlog* as the
size of the queue for pending requests. Returns 0 on success and -1 on error. This call returns -1
immediately without changing the error status of *skt* if *skt* is marked with an error on entry.

Errors

X_BADSOCK	bad X_SOCKET
X_NOLISTEN	socket is not able to perform a listen

X_SOCKET *x_accept(X_SOCKET *skt, struct sockaddr *addr, int *len)

For a socket *skt* in the listening state, this call removes the first pending connection request from
the queue and returns a socket for use with the connection. If, on entry, *skt* is marked with an
error, this call immediately returns NULL without changing the error status.

Errors

X_BADSOCK	bad X_SOCKET
X_NOACCEPT	socket is not able to perform an accept because it is not listening
X_NOTSTREAM	socket is not of type SOCK_STREAM
X_WOULDBLOCK	socket is non-blocking and there are no waiting connections
X_IMPL	underlying socket error

int   **x_setsockopt(X_SOCKET *skt, int optname, void *optval, int optlen)**

Uses the data described by *optval* and *optlen* to set *optname* on socket *skt*. Returns 0 on success
and -1 on error. If, on entry, *skt* is marked with an error, this call immediately returns -1 without
changing the error status. The possible option names are

SOCK_STREAM

SOCK_DGRAM

QTTL	int	The duration (in msec) that unacknowledged messages will remain available for retransmission.

Both

SNDSIZE	int	The packet size used for the underlying transport. This defaults to the MTU.
NOPMTUD	int	Disable PMTUD.

Errors

X_BADSOCK	bad X_SOCKET

	X_BADOPT	unknown or illegal option/length

int   **x_getsockopt(X_SOCKET *skt, int optname, void *optval, int *optlen)**

Returns, in *optval* and *optlen*, the data describing *optname* on socket *skt*. On entry *optlen* is the size of space pointed to by *optval*. On exit, *optlen* is changed to reflect the actual size of the returned data. This call returns 0 on success and -1 on error. If, on entry, *skt* is marked with an error, this call immediately returns -1 without changing the error status. In addition to the option names valid for the x_setsockopt() call the following additional options are available.

SOCK_STREAM

SOCK_DGRAM
        MTU   int   current MTU
        MSGSIZE   int   The maximum size of an individual message.

X_STATS * **x_sockstats(X_SOCKET *skt)**

Compute and return current performance data for socket *skt*. If, on entry, *skt* is marked with an error, this call immediately returns NULL.

This structure contains average performance statistics for the socket and incremental statistics since the last call to x_sockstats(). Performance statistics include the following.

    - send rate in Mbps
    - recv rate in Mbps
    - retransmit traffic as a fraction of total traffic
    - round trip time

For a SOCK_DGRAM socket, configured for partial reliability, the performance statistics also include the number of messages that expire after exceeding the QTTL limit.

int   **x_sockerror(X_SOCKET *skt)**

Returns the error condition of socket *skt*. If *skt* is X_LAST_ERROR the last library wide error is returned.

char *   **x_errortext(int err)**

Returns a text description of the error code *err*.

size_t   **x_sendfile(X_SOCKET *skt, int fd, off_t offset, size_t size)**

Mmap the file described by *fd* and send *size* bytes of data starting at *offset*. Returns the actual number of bytes sent or -1 on error.

size_t   **x_send(X_SOCKET *skt, const void * buf, size_t len)**

Send *len* bytes from *buf* on socket *skt*. Returns the actual number of bytes sent or -1 on error.

If *skt* is of type SOCK_DGRAM then the data is treated as a single message and *len* must be less then the maximum message length. Messages greater than SNDSIZE are fragmented. On a fully reliable socket messages will be delivered intact but may not arrive in the original order. On an unreliable socket messages are either delivered intact or discarded. If the socket is configured for partial reliability, message fragments will be retransmitted until no longer available in which case the message will be discarded.

size_t   **x_recvfile(X_SOCKET *skt, int fd, off_t offset, size_t size)**

Reads *size* bytes from socket *skt* writing them into the file described by *fd* at *offset*. Returns the number of bytes read on success and -1 on error.

size_t   **x_recv(X_SOCKET *skt, void *buf, size_t len)**

Read up to *len* bytes from socket *skt* into *buf*. If *skt* is of type SOCK_DGRAM then an entire message is read into *buf* with bytes in excess of *len* being discarded. Returns the number of bytes read on success and -1 on error.

void   **x_selectmark(X_SOCKET *skt, int mark)**

Called prior to using x_select(), this function marks *skt* with the conditions to be tested by the select call. *Mark* is a bitwise OR of X_READABLE, X_WRITABLE and X_EXCEPTION.

int   **x_selecttest(X_SOCKET *skt)**

Returns the result of the x_select() call for *skt*. The return value will be a bitwise OR of X_READABLE, X_WRITABLE and X_EXCEPTION depending on *skt* was marked and the result of the x_select() call.

int   **x_select(int len, X_SOCKET **skts, struct timeval *timeout)**

Returns the number of sockets in the array *skts* of length *len* that satisfy the marked conditions. *Timeout* is the maximum time to wait before the call returns. If *timeout* is null the call blocks indefinitely. A *timeout* value of zero can be used to effect a poll operation. *Timeout* is not changed by the call. The call returns -1 on error.