## NAME

**sctp** - Internet Stream Control Transmission Protocol

## SYNOPSIS

**options SCTP**
**options SCTP_SUPPORT**

**#include <sys/types.h>**
**#include <sys/socket.h>**
**#include <netinet/sctp.h>**

*int*
**socket**(*AF_INET*, *SOCK_STREAM*, *IPPROTO_SCTP*);

*int*
**socket**(*AF_INET*, *SOCK_SEQPACKET*, *IPPROTO_SCTP*);

## DESCRIPTION

The SCTP protocol provides reliable, flow-controlled, two-way transmission of data. It is a message oriented protocol and can support the SOCK_STREAM and SOCK_SEQPACKET abstractions. SCTP uses the standard Internet address format and, in addition, provides a per-host collection of "port addresses". Thus, each address is composed of an Internet address specifying the host and network, with a specific SCTP port on the host identifying the peer entity.

There are two models of programming in SCTP. The first uses the SOCK_STREAM abstraction. In this abstraction sockets utilizing the SCTP protocol are either "active" or "passive". Active sockets initiate connections to passive sockets. By default, SCTP sockets are created active; to create a passive socket, the listen(2) system call must be used after binding the socket with the bind(2) or sctp_bindx(3) system calls. Only passive sockets may use the accept(2) call to accept incoming connections. Only active sockets may use the connect(2) call to initiate connections.

The other abstraction SOCK_SEQPACKET provides a "connectionless" mode of operation in that the user may send to an address (using any of the valid send calls that carry a socket address) and an association will be setup implicitly by the underlying SCTP transport stack. This abstraction is the only one capable of sending data on the third leg of the four-way handshake. A user must still call listen(2) to allow the socket to accept connections. Calling listen(2) however does not restrict the user from still initiating implicit connections to other peers.

The SCTP protocol directly supports multi-homing. So when binding a socket with the "wildcard" address INADDR_ANY, the SCTP stack will inform the peer about all of the local addresses that are

deemed in scope of the peer. The peer will then possibly have multiple paths to reach the local host.

The SCTP transport protocol is also multi-streamed. Multi-streaming refers to the ability to send sub-ordered flows of messages. A user performs this by specifying a specific stream in one of the extended send calls such as the sctp_send(3) function call. Sending messages on different streams will allow parallel delivery of data i.e., a message loss in stream 1 will not block the delivery of messages sent in stream 2.

The SCTP transport protocol also provides a unordered service as well. The unordered service allows a message to be sent and delivered with no regard to the ordering of any other message.

The SCTP kernel implementation may either be compiled into the kernel, or loaded dynamically as a module. To support dynamic loading of the stack, the kernel must be compiled with **options SCTP_SUPPORT**.

## Extensions

The FreeBSD implementation of SCTP also supports the following extensions:

sctp partial reliability
: This extension allows one to have message be skipped and not delivered based on some user specified parameters.

sctp dynamic addressing
: This extension allows addresses to be added and deleted dynamically from an existing association.

sctp authentication
: This extension allows the user to authenticate specific peer chunks (including data) to validate that the peer who sent the message is in fact the peer who setup the association. A shared key option is also provided for so that two stacks can pre-share keys.

packet drop
: Some routers support a special satellite protocol that will report losses due to corruption. This allows retransmissions without subsequent loss in bandwidth utilization.

stream reset
: This extension allows a user on either side to reset the stream sequence numbers used by any or all streams.

## Socket Options

SCTP supports a number of socket options which can be set with setsockopt(2) and tested with getsockopt(2) or sctp_opt_info(3):

SCTP_NODELAY

Under most circumstances, SCTP sends data when it is presented; when outstanding data has not yet been acknowledged, it gathers small amounts of output to be sent in a single packet once an acknowledgement is received. For some clients, such as window systems that send a stream of mouse events which receive no replies, this packetization may cause significant delays. The boolean option SCTP_NODELAY defeats this algorithm.

SCTP_RTOINFO

This option returns specific information about an associations "Retransmission Time Out". It can also be used to change the default values.

SCTP_ASSOCINFO

This option returns specific information about the requested association.

SCTP_INITMSG

This option allows you to get or set the default sending parameters when an association is implicitly setup. It allows you to change such things as the maximum number of streams allowed inbound and the number of streams requested of the peer.

SCTP_AUTOCLOSE

For the one-to-many model (SOCK_SEQPACKET) associations are setup implicitly. This option allows the user to specify a default number of idle seconds to allow the association be maintained. After the idle timer (where no user message have been sent or have been received from the peer) the association will be gracefully closed. The default for this value is 0, or unlimited (i.e., no automatic close).

SCTP_SET_PEER_PRIMARY_ADDR

The dynamic address extension allows a peer to also request a particular address of its be made into the primary address. This option allows the caller to make such a request to a peer. Note that if the peer does not also support the dynamic address extension, this call will fail. Note the caller must provide a valid local address that the peer has been told about during association setup or dynamically.

SCTP_PRIMARY_ADDR

This option allows the setting of the primary address that the caller wishes to send to. The caller provides the address of a peer that is to be made primary.

SCTP_ADAPTATION_LAYER

The dynamic address extension also allows a user to pass a 32 bit opaque value upon association setup. This option allows a user to set or get this value.

SCTP_DISABLE_FRAGMENTS

By default SCTP will fragment user messages into multiple pieces that will fit on the network and then later, upon reception, reassemble the pieces into a single user message. If this option is enabled instead, any send that exceeds the path maximum transfer unit (P-MTU) will fail and the message will NOT be sent.

SCTP_PEER_ADDR_PARAMS

This option will allow a user to set or get specific peer address parameters.

SCTP_DEFAULT_SEND_PARAM

When a user does not use one of the extended send calls (e.g., sctp_sendmsg(3)) a set of default values apply to each send. These values include things like the stream number to send to as well as the per-protocol id. This option lets a caller both get and set these values. If the user changes these default values, then these new values will be used as the default whenever no information is provided by the sender (i.e., the non-extended API is used).

SCTP_EVENTS

SCTP has non-data events that it can communicate to its application. By default these are all disabled since they arrive in the data path with a special flag MSG_NOTIFICATION set upon the received message. This option lets a caller both get what events are current being received as well as set different events that they may be interested in receiving.

SCTP_I_WANT_MAPPED_V4_ADDR

SCTP supports both IPV4 and IPV6. An association may span both IPV4 and IPV6 addresses since SCTP is multi-homed. By default, when opening an IPV6 socket, when data arrives on the socket from a peer's V4 address the V4 address will be presented with an address family of AF_INET. If this is undesirable, then this option can be enabled which will then convert all V4 addresses into mapped V6 representations.

SCTP_MAXSEG

By default SCTP chooses its message fragmentation point based upon the smallest P-MTU of the peer. This option lets the caller set it to a smaller value. Note that while the user can change this value, if the P-MTU is smaller than the value set by the user, then the P-MTU value will override any user setting.

SCTP_DELAYED_SACK

This option lets the user both set and get the delayed ack time (in milliseconds) and the ack frequency that SCTP is using. The default delayed ack time is 200 milliseconds and the default ack frequency is 2.

SCTP_PARTIAL_DELIVERY_POINT

SCTP at times may need to start delivery of a very large message before the entire message has arrived. By default SCTP waits until the incoming message is larger than one fourth of the receive buffer. This option allows the stacks value to be overridden with a smaller value.

SCTP_FRAGMENT_INTERLEAVE

SCTP at times will start partial delivery (as mentioned above). In the normal case successive reads will continue to return the rest of the message, blocking if needed, until all of that message is read. However this means other messages may have arrived and be ready for delivery and be blocked behind the message being partially delivered. If this option is enabled, when a partial delivery message has no more data to be received, then a subsequent read may return a different message that is ready for delivery. By default this option is off since the user must be using the extended API's to be able to tell the difference between messages (via the stream and stream sequence number).

SCTP_AUTH_CHUNK

By default only the dynamic addressing chunks are authenticated. This option lets a user request an additional chunk be authenticated as well. Note that successive calls to this option will work and continue to add more chunks that require authentication. Note that this option only effects future associations and not existing ones.

SCTP_AUTH_KEY

This option allows a user to specify a shared key that can be later used to authenticate a peer.

SCTP_HMAC_IDENT

This option will let you get or set the list of HMAC algorithms used to authenticate peers. Note that the HMAC values are in priority order where the first HMAC identifier is the most preferred and the last is the least preferred.

SCTP_AUTH_ACTIVE_KEY

This option allows you to make a key active for the generation of authentication information. Note that the peer must have the same key or else the data will be discarded.

SCTP_AUTH_DELETE_KEY

This option allows you to delete an old key.

SCTP_USE_EXT_RECVINFO

The sockets api document allows an extended send/receive information structure to be used. The extended structure includes additional fields related to the next message to be received (after the current receive completes) if such information is known. By default the system will not pass this

information. This option allows the user to request this information.

SCTP_AUTO_ASCONF
By default when bound to all address and the system administrator has enables automatic dynamic addresses, the SCTP stack will automatically generate address changes into add and delete requests to any peers by setting this option to true. This option allows an endpoint to disable that behavior.

SCTP_MAXBURST
By default SCTP implements micro-burst control so that as the congestion window opens up no large burst of packets can be generated. The default burst limit is four. This option lets the user change this value.

SCTP_CONTEXT
Many sctp extended calls have a context field. The context field is a 32 bit opaque value that will be returned in send failures. This option lets the caller set the default context value to use when none is provided by the user.

SCTP_EXPLICIT_EOR
By default, a single send is a complete message. SCTP generates an implied record boundary. If this option is enabled, then all sends are part of the same message until the user indicates an end of record with the special flag SCTP_EOR passed in the sctp_sndrcvinfo flags field. This effectively makes all sends part of the same message until the user specifies differently. This means that a caller must NOT change the stream number until after the SCTP_EOR is passed to SCTP else an error will be returned.

SCTP_STATUS
This option is a read-only option that returns various status information about the specified association.

SCTP_GET_PEER_ADDR_INFO
This read-only option returns information about a peer address.

SCTP_PEER_AUTH_CHUNKS
This read-only option returns a list of the chunks the peer requires to be authenticated.

SCTP_LOCAL_AUTH_CHUNKS
This read-only option returns a list of the locally required chunks that must be authenticated.

SCTP_RESET_STREAMS

This socket option is used to cause a stream sequence number or all stream sequence numbers to be reset. Note that the peer SCTP endpoint must also support the stream reset extension as well.

## MIB Variables

The SCTP protocol implements a number of variables in the *net.inet.sctp* branch of the sysctl(3) MIB.

### Congestion Control

*default_cc_module*

Default congestion control module. Default value is 0. The minimum is 0, and the maximum is 3. A value of 0 enables the default congestion control algorithm. A value of 1 enables the High Speed congestion control algorithm. A value of 2 enables the HTCP congestion control algorithm. A value of 3 enables the data center congestion control (DCCC) algorithm.

*initial_cwnd*

Defines the initial congestion window size in MTUs.

*cwnd_maxburst*

Use congestion control instead of 'blind' logic to limit maximum burst when sending. Default value is 1. May be set to 0 or 1.

*ecn_enable*

Enable Explicit Congestion Notification (ECN). Default value is 1. May be set to 0 or 1.

*rttvar_steady_step*

Number of identical bandwidth measurements DCCC takes to try step down the congestion window. Default value is 20. The minimum is 0, and the maximum is 65535.

*rttvar_eqret*

Whether DCCC reduces the congestion window size when round-trip time and bandwidth remain unchanged. Default value is 0. May be set to 0 or 1.

*rttvar_bw*

Shift amount DCCC uses for bandwidth smoothing on round-trip-time calculation. Default value is 4. The minimum is 0, and the maximum is 32.

*rttvar_rtt*

Shift amount DCCC uses for round-trip-time smoothing on round-trip-time calculation. Default value is 5. The minimum is 0, and the maximum is 32.

*use_dcccecn*
Enable ECN when using DCCC. Default value is 1. May be set to 0 or 1.

**Misc**

*getcred*
Get the ucred of a SCTP connection.

*assoclist*
List of active SCTP associations.

*stats* SCTP statistics (struct sctp_stat).

*diag_info_code*
Diagnostic information error cause code.

*blackhole*
Enable SCTP blackholing. See blackhole(4) for more details.

*sendall_limit*
Maximum message size (in bytes) that can be transmitted with SCTP_SENDALL flags set.

*buffer_splitting*
Enable send/receive buffer splitting.

*vtag_time_wait*
Vtag wait time in seconds, 0 to disable.

*nat_friendly_init*
Enable sending of the NAT-friendly SCTP option on INITs.

*enable_sack_immediately*
Enable sending of the SACK-IMMEDIATELY bit.

*udp_tunneling_port*
Set the SCTP/UDP tunneling port.

*mobility_fasthandoff*
Enable SCTP fast handoff.

*mobility_base*
Enable SCTP base mobility

*default_frag_interleave*
Default fragment interleave level.

*default_ss_module*
Default stream scheduling module.

*log_level*
Ltrace/KTR trace logging level.

*max_retran_chunk*
Number of retransmissions of a DATA chunk before an association is aborted.

*min_residual*
Minimum residual data chunk in second part of split.

*strict_data_order*
Enforce strict data ordering, abort if control inside data.

*abort_at_limit*
Abort when one-to-one hits qlimit.

*hb_max_burst*
Confirmation heartbeat max burst.

*do_sctp_drain*
Flush chunks in receive queues with TSN higher than the cumulative TSN if the system is low on mbufs.

*max_chained_mbufs*
Default max number of small mbufs on a chain.

*abc_l_var*
SCTP ABC max increase per SACK (L).

*nat_friendly*
SCTP NAT friendly operation.

*cmt_use_dac*
CMT DAC on/off flag.

*cmt_on_off*
CMT settings.

*outgoing_streams*
Default number of outgoing streams.

*incoming_streams*
Default number of incoming streams.

*add_more_on_output*
When space-wise is it worthwhile to try to add more to a socket send buffer.

*path_pf_threshold*
Default potentially failed threshold.

*path_rtx_max*
Default maximum of retransmissions per path.

*assoc_rtx_max*
Default maximum number of retransmissions per association.

*init_rtx_max*
Default maximum number of retransmissions for INIT chunks.

*valid_cookie_life*
Default cookie lifetime in seconds.

*init_rto_max*
Default maximum retransmission timeout during association setup in ms.

*rto_initial*
Default initial retransmission timeout in ms.

*rto_min*
Default minimum retransmission timeout in ms.

*rto_max*

Default maximum retransmission timeout in ms.

*secret_lifetime*
Default secret lifetime in seconds.

*shutdown_guard_time*
Shutdown guard timer in seconds (0 means 5 times RTO.Max).

*pmtu_raise_time*
Default PMTU raise timer in seconds.

*heartbeat_interval*
Default heartbeat interval in ms.

*asoc_resource*
Max number of cached resources in an association.

*sys_resource*
Max number of cached resources in the system.

*sack_freq*
Default SACK frequency.

*delayed_sack_time*
Default delayed SACK timer in ms.

*chunkscale*
Tunable for scaling of number of chunks and messages.

*min_split_point*
Minimum size when splitting a chunk.

*pcbhashsize*
Tunable for PCB hash table sizes.

*tcbhashsize*
Tunable for TCB hash table sizes.

*maxchunks*
Default max chunks on queue per association.

*fr_maxburst*
: Default max burst for SCTP endpoints when fast retransmitting.

*maxburst*
: Default max burst for SCTP endpoints.

*peer_chkoh*
: Amount to debit peers rwnd per chunk sent.

*strict_sacks*
: Enable SCTP Strict SACK checking.

*pktdrop_enable*
: Enable SCTP PKTDROP.

*nrsack_enable*
: Enable SCTP NR-SACK.

*reconfig_enable*
: Enable SCTP RE-CONFIG.

*asconf_enable*
: Enable SCTP ASCONF.

*auth_enable*
: Enable SCTP AUTH.

*pr_enable*
: Enable PR-SCTP.

*auto_asconf*
: Enable SCTP Auto-ASCONF.

*recvspace*
: Maximum incoming SCTP buffer size.

*sendspace*
: Maximum outgoing SCTP buffer size.

**SEE ALSO**

accept(2), bind(2), connect(2), listen(2), sctp_bindx(3), sctp_connectx(3), sctp_opt_info(3), sctp_recvmsg(3), sctp_sendmsg(3), blackhole(4)

## BUGS

The **sctp** kernel module cannot be unloaded.