Stream Control Transmission Protocol

Stream Control Transmission Protocol (SCTP) is a reliable datagram-oriented IP transport protocol specified by RFC 2960. It provides the layer between an SCTP user application and an unreliable end-to-end datagram service such as IP. The basic service offered by SCTP is the reliable transfer of user datagrams between peer SCTP users. It performs this service within the context of an association between two SCTP hosts. SCTP is connection-oriented, but SCTP association is a broader concept than the Transmission Control Protocol (TCP) connection, for example.

SCTP provides the means for each SCTP endpoint to provide its peer with a list of transport addresses, such as address and UDP port combinations. This list is provided during association startup and shows the transport addresses through which the endpoint can be reached and from which messages originate. The SCTP association includes transfer over all the possible source and destination combinations that might be generated from the two endpoint lists (also known as multihoming).

SCTP is not explicitly configured on routers, but it underlies several Cisco applications. The commands described in this document are useful for troubleshooting when SCTP issues are suspected as the cause of problems.

- Finding Feature Information, on page 1
- Prerequisites for SCTP, on page 2
- Information About SCTP, on page 2
- How to Configure SCTP, on page 4
- Configuration Examples for SCTP, on page 20
- Additional References, on page 21
- Feature Information for SCTP, on page 22

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for SCTP

- Cisco IOS Release 12.2(2)MB, Cisco IOS Release 12.2(4)T, or a later release.

Information About SCTP

Benefits of SCTP

SCTP provides the following services and features:

- Acknowledged reliable nonduplicated transfer of user data
- Application-level segmentation to conform to the maximum transmission unit (MTU) size
- Sequenced delivery of user datagrams within multiple streams
- Optional multiplexing of user datagrams into SCTP datagrams
- Enhanced reliability through support of multihoming at either end or both ends of the association
- Congestion avoidance and resistance to flooding and masquerade attacks

SCTP Release 2

SCTP Release 2 introduced updated output for the following SCTP commands:

- **show ip sctp association parameters**
- **show ip sctp association statistics**

SCTP Show Clear CLI Enhancements

The SCTP Show/Clear CLI Enhancements feature provides access to additional SCTP information that can help with troubleshooting potential problems. These enhancements also make the updated SCTP **show** and **clear** commands consistent with the CLI of other transport protocols.

The following commands were replaced by this feature:

clear ip sctp statistics , **show ip sctp association list**, **show ip sctp association parameters**, **show ip sctp association statistics**, **show ip sctp errors**, **show ip sctp instances**, **show ip sctp statistics**

The following commands were introduced by this feature:

clear sctp statistics , **show sctp association**, **show sctp association list**, **show sctp association parameters**, **show sctp association statistics**, **show sctp errors**, **show sctp instance**,**show sctp instances**,**show sctp instances**, **show sctp statistics**

SCTP Stream Reset

SCTP Release 4 introduced the SCTP stream reset feature. The SCTP stream reset feature enables SCTP to reset stream transport sequence numbers and all stream sequence numbers. Applications that use SCTP may reset a stream and restart the numbering sequence of the stream at zero. A notification to the upper layer that the stream numbering sequence has been reset is sent by the application. Resetting an SCTP stream enables applications to reuse streams for different purposes while preserving the stream sequence number for the application so that message flows can be tracked. Without the SCTP stream reset feature, reusing streams results in message numbers larger than expected. SCTP stream reset enables SCTP to:

- Dynamically reset a peer's outbound streams
- Dynamically reset a local host's outbound stream
- Dynamically reset specific numbered streams

The resetting of a stream occurs upon request from an upper layer application. This capability is enabled by default in Cisco IOS Release 12.4(15)T and later releases.

SCTP Authentication

SCTP Release 4 introduced the SCTP Authentication feature. The SCTP Authentication feature enables SCTP to:

- Set up a dynamic shared association key with no shared secret
- Allow a shared secret to be combined with an association key
- Use the shared association secret to authenticate chunks
- Negotiate which chunk types must be authenticated

The SCTP Authentication feature enables applications to use these optional extensions. Use the **ip sctp authenticate** command to define chunks that the client requires be authenticated.

SCTP Authentication procedures use either Message Digest 5 (MD5) or Secure Hash Algorithm 1 (SHA-1), which can be memory and CPU intensive. Enabling SCTP Authentication on DATA chunks could impact CPU utilization when a large number of authenticated chunks are sent.

You can define the types of SCTP chunks the client requires be authenticated. Optionally, you can use the **ip sctp asconf** command to configure SCTP to send an Address Configuration Change (ASCONF) chunk automatically in response to an IP address change in an SCTP stream, or to check that the endpoint supports authentication before sending the ASCONF chunk.

The table below lists the SCTP chunk types and numbers that can be authenticated by entering the **ip sctp authenticate** command in global configuration mode.

Table 1: SCTP Authentication Chunk Types

SCTP Chunk Type	SCTP Chunk Number	Description
abort association	0x06	ABORT chunk.
asconf	0xc1	ASCONF. Address configuration change chunk.
asconf-ack	0x80	ASCONF-ACK. ASCONF acknowledgement.

SCTP Chunk Type	SCTP Chunk Number	Description
cookie-ack	0x0b	COOKIE acknowledgment chunk.
cookie-echo	0x0a	COOKIE-ECHO chunk.
data	0x00	DATA chunk.
fwd-tsn	0xc0	FWD-CUM-TSN chunk. Forwarded cumulative transmission sequence number chunk.
heartbeat	0x04	HEARTBEAT request chunk.
heartbeat-ack	0x05	HEARTBEAT acknowledgement chunk.
packet-drop	0x81	PACKET-DROP chunk.
sack	0x03	Selective acknowledgment chunk.
shutdown	0x07	SHUTDOWN chunk.
shutdown-ack	0x08	SHUTDOWN acknowledgment chunk.
stream-reset	0x82	STREAM-RESET chunk.

SCTP Add-IP

SCTP Release 4 introduced the SCTP Add-IP feature. The SCTP Add-IP feature enables the ability to add or delete an IP address for an endpoint of an existing SCTP association and to communicate this change to the remote end. An ADD-IP chunk is sent to the remote end adding or removing the redundant server addresses of the association. The ADD-IP chunk also deletes all addresses of a failed host from an association.

In Cisco IOS software, adding or deleting an IP address from an SCTP association can be triggered programmatically by the application, or automatically in response to an IP address change on the router.

The SCTP Add-IP feature also enables an application to programmatically set the primary address for an SCTP association.

The SCTP Add-IP feature introduces two new SCTP chunk types, the Address Configuration (ASCONF) chunk, and the Address Acknowledgement (ASCONF-ACK) chunk. The ASCONF chunk is used by the sender in an SCTP stream to communicate to the remote endpoint that the SCTP stream contains a configuration change request that must be acknowledged. The receiving endpoint of an ASCONF chunk uses the ASCONF-ACK chunk to acknowledge receipt of the ASCONF chunk.

The **ip asconf auto**command is used to configure an application to automatically send an ASCONF chunk in response to an IP Address change on the endpoint and to configure SCTP to either accept the ASCONF chunk automatically, or to require the chunk be authenticated.

How to Configure SCTP

SCTP is not explicitly configured on routers, but it underlies several Cisco applications. This section contains the following tasks:

Configuring SCTP Authentication Parameters

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip sctp authenticate** {*chunk-type* | *chunk-number*}
4. **ip sctp asconf** {**authenticate check** | **auto**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip sctp authenticate** {*chunk-type*	*chunk-number*} **Example:** `Router(config)# ip sctp authenticate sack`	Defines Stream Control Transmission Protocol (SCTP) data chunks that the client requires be authenticated.
Step 4	**ip sctp asconf** {**authenticate check**	**auto**} **Example:** `Router(config)# ip sctp asconf auto`	(Optional) Configures SCTP to send ASCONF chunks automatically in response to an IP address change on a router, or to check that the endpoint supports authentication before sending the ASCONF chunk.

Displaying Information About SCTP Associations and Parameters

To display information about SCTP associations and instances, use the following commands.

Note SCTP commands that display statistical information show only the information that is available since the last time a **clear sctp statistics** command was executed. The **clear sctp statistics** command clears all SCTP statistics, both those compiled for individual associations and those compiled overall.

SUMMARY STEPS

1. **show sctp association list**
2. **show sctp association parameters**
3. **show sctp association statistics**

4. **show sctp errors**
5. **show sctp instance**
6. **show sctp instances**
7. **show sctp statistics**

DETAILED STEPS

Step 1 show sctp association list

The **show sctp association list** command provides the current SCTP association and instance identifiers, the current state of SCTP associations, and the local and remote port numbers and addresses that are used in the associations. The following example shows three current associations that are in the established state. Each association belongs to the same instance, as noted by their instance identifiers.

Example:

```
Router# show sctp association list

*** SCTP Association List ****
AssocID:0,  Instance ID:0
Current state:ESTABLISHED
Local port:8989, Addrs:10.1.0.2 10.2.0.2
Remote port:8989, Addrs:10.6.0.4 10.5.0.4
AssocID:1,  Instance ID:0
Current state:ESTABLISHED
Local port:8989, Addrs:10.1.0.2 10.2.0.2
Remote port:8990, Addrs:10.6.0.4 10.5.0.4
AssocID:2,  Instance ID:0
Current state:ESTABLISHED
Local port:8989, Addrs:10.1.0.2 10.2.0.2
Remote port:8991, Addrs:10.6.0.4 10.5.0.4
```

Step 2 show sctp association parameters

The **show sctp association parameters** command provides information to determine the stability of SCTP associations, dynamically calculated statistics about destinations, and values to assess network congestion. This command also displays parameter values for the specified association.

The association configuration section displays information similar to that in the **show sctp association list** command, including association identifiers, state, and local and remote port and address information. The current primary destination is also displayed.

The following sample output shows the IP SCTP association parameters for association 0:

Example:

```
Router# show sctp association parameters 0

** SCTP Association Parameters **
AssocID: 0  Context: 0  InstanceID: 1
Assoc state: ESTABLISHED  Uptime: 19:05:57.425
Local port: 8181
Local addresses: 10.1.0.3 10.2.0.3
Remote port: 8181
Primary dest addr: 10.5.0.4
Effective primary dest addr: 10.5.0.4
Destination addresses:
10.5.0.4:   State:  ACTIVE
```

```
  Heartbeats:  Enabled   Timeout: 30000 ms
  RTO/RTT/SRTT: 1000/16/38 ms   TOS: 0  MTU: 1500
  cwnd: 5364  ssthresh: 3000  outstand: 768
  Num retrans: 0  Max retrans: 5  Num times failed: 0
10.6.0.4:   State:  ACTIVE
  Heartbeats:  Enabled   Timeout: 30000 ms
  RTO/RTT/SRTT: 1000/4/7 ms   TOS: 0  MTU: 1500
  cwnd: 3960  ssthresh: 3000  outstand: 0
  Num retrans: 0  Max retrans: 5  Num times failed: 0
Local vertag: 9A245CD4  Remote vertag: 2A08D122
Num inbound streams: 10  outbound streams: 10
Max assoc retrans: 5  Max init retrans: 8
CumSack timeout: 200 ms  Bundle timeout: 100 ms
Min RTO: 1000 ms  Max RTO: 60000 ms
LocalRwnd: 18000  Low: 13455   RemoteRwnd: 15252  Low: 13161
Congest levels: 0  current level: 0  high mark: 325
```

Step 3 **show sctp association statistics**

This command shows only the information that has become available since the last time a**clear sctp statistics** command was executed for an SCTP association.

The following sample output shows the statistics accumulated for SCTP association 0:

Example:

```
Router# show sctp association statistics 0

** SCTP Association Statistics **
AssocID/InstanceID: 0/1
Current State: ESTABLISHED
Control Chunks
  Sent: 623874  Rcvd: 660227
Data Chunks Sent
  Total: 14235644  Retransmitted: 60487
  Ordered: 6369678  Unordered: 6371263
  Avg bundled: 18  Total Bytes: 640603980
Data Chunks Rcvd
  Total: 14496585  Discarded: 1755575
  Ordered: 6369741  Unordered: 6371269
  Avg bundled: 18  Total Bytes: 652346325
  Out of Seq TSN: 3069353
ULP Dgrams
  Sent: 12740941  Ready: 12740961  Rcvd: 12740941
```

Step 4 **show sctp errors**

The **show sctp errors** command displays all errors across all associations that have been logged since the last time that the SCTP statistics were cleared with the **clear sctp statistics** command. If no errors have been logged, this is indicated in the output. The following sample output shows a session that has SCTP errors:

Example:

```
Router# show sctp errors

** SCTP Error Statistics **
Invalid verification tag:       5
Communication Lost:             64
Destination Address Failed:     3
Unknown INIT params rcvd:       16
Invalid cookie signature:       5
Expired cookie:                 1
```

```
Peer restarted:                1
No Listening instance:         2
```

Step 5 **show sctp instance**

The **show sctp instance** command displays information for the currently configured instance with the ID specified in the command syntax. The instance number, local port, and address information are displayed. The instance state is either *available* or *deletion pending*. An instance enters the deletion pending state when a request is made to delete it but there are currently established associations for that instance. The instance cannot be deleted immediately and instead enters the pending state. No new associations are allowed in this instance, and when the last association is terminated or fails, the instance is deleted.

The default inbound and outbound stream numbers (shown in the example output in the next section) are used for establishing incoming associations, the maximum number of associations allowed for this instance is shown, and a snapshot of each existing association is shown, if any exists.

The following sample output displays information for SCTP instance 0. In this example, instance 0 is using local port 1000 and has three current associations.

Example:

```
Router# show sctp instance 0

Instance ID:0 Local port:1000 State:available
Local addrs:10.1.0.2 10.2.0.2
Default streams inbound:1  outbound:1
  Current associations: (max allowed:200)
  AssocID:0  State:ESTABLISHED  Remote port:8989
    Dest addrs:10.6.0.4 10.5.0.4
  AssocID:1  State:ESTABLISHED  Remote port:8990
    Dest addrs:10.6.0.4 10.5.0.4
  AssocID:2  State:ESTABLISHED  Remote port:8991
    Dest addrs:10.6.0.4 10.5.0.4
```

Step 6 **show sctp instances**

The **show sctp instances** command displays information for each of the currently configured instances. The instance number, local port, and address information are displayed. The instance state is either *available* or *deletion pending* . An instance enters the deletion pending state when a request is made to delete it but there are currently established associations for that instance. The instance cannot be deleted immediately and instead enters the pending state. No new associations are allowed in this instance, and when the last association is terminated or fails, the instance is deleted.

The default inbound and outbound stream numbers are used for establishing incoming associations, the maximum number of associations allowed for this instance is shown, and a snapshot of each existing association is shown, if any exists.

Note When you enter the **show sctp instances** command, you must type the complete word **instances** in the command syntax. If you try to enter an abbreviated form of this word, there will be a partial match that identifies the **show sctp instance** *instance-id* command.

The following sample output shows available IP SCTP instances. In this example, two current instances are active and available. The first is using local port 8989, and the second is using 9191. Instance identifier 0 has three current associations, and instance identifier 1 has no current associations.

Example:

```
Router# show sctp instances

*** SCTP Instances ****
Instance ID:0  Local port:8989
```

```
Instance state:available
Local addrs:10.1.0.2 10.2.0.2
Default streams inbound:1  outbound:1
  Current associations: (max allowed:6)
  AssocID:0  State:ESTABLISHED  Remote port:8989
    Dest addrs:10.6.0.4 10.5.0.4
  AssocID:1  State:ESTABLISHED  Remote port:8990
    Dest addrs:10.6.0.4 10.5.0.4
  AssocID:2  State:ESTABLISHED  Remote port:8991
    Dest addrs:10.6.0.4 10.5.0.4
Instance ID:1  Local port:9191
Instance state:available
Local addrs:10.1.0.2 10.2.0.2
Default streams inbound:1  outbound:1
No current associations established for this instance.
Max allowed:6
```

Step 7 **show sctp statistics**

The **show sctp statistics** command displays the overall SCTP statistics accumulated since the last **clear sctp statistics** command. It includes numbers for all currently established associations, and for any that have been terminated. The statistics indicated are similar to those shown for individual associations. The following sample output shows SCTP statistics:

Example:

```
Router# show sctp statistics

*** SCTP Overall Statistics ****
Total Chunks Sent:           2097
Total Chunks Rcvd:           2766
Data Chunks Rcvd In Seq:     538
Data Chunks Rcvd Out of Seq: 0
Total Data Chunks Sent:      538
Total Data Chunks Rcvd:      538
Total Data Bytes Sent:       53800
Total Data Bytes Rcvd:       53800
Total Data Chunks Discarded: 0
Total Data Chunks Retrans:   0
Total SCTP Dgrams Sent:      1561
Total SCTP Dgrams Rcvd:      2228
Total ULP Dgrams Sent:       538
Total ULP Dgrams Ready:      538
Total ULP Dgrams Rcvd:       538
```

Troubleshooting SCTP Associations and Parameters

This section describes the debug commands available for troubleshooting SCTP associations and parameters.

In a live system, the debug commands for performance, state, signal, and warnings are the most useful. These debug commands show any association or destination address failures and can be used to monitor the stability of any established associations.

Debug commands other than those for performance, state, signal, and warnings can generate a great deal of output and therefore can cause associations to fail. These commands should be used only in test environments or when there are very low amounts of traffic.

Caution Many SCTP debug commands should be used with extreme caution or not at all in live systems, depending on the amount of traffic, because the extra messages they generate may cause associations to fail. This caution is repeated in descriptions of the commands that may cause disruption to live systems.

Note SCTP debug commands display information for all current SCTP associations and cannot be limited to particular associations.

SUMMARY STEPS

1. **debug ip sctp api**
2. **debug ip sctp congestion**
3. **debug ip sctp init**
4. **debug ip sctp multihome**
5. **debug ip sctp performance**
6. **debug ip sctp rcvchunks**
7. **debug ip sctp rto**
8. **debug ip sctp segments**
9. **debug ip sctp segmentv**
10. **debug ip sctp signal**
11. **debug ip sctp state**
12. **debug ip sctp sndchunks**
13. **debug ip sctp timer**
14. **debug ip sctp warnings**

DETAILED STEPS

Step 1 debug ip sctp api

The **debug ip sctp api** command shows all SCTP calls to the application programming interface (API) that are being executed and the parameters associated with these calls.

Caution The **debug ip sctp api** command should not be used in a live system that has any significant amount of traffic running because it can generate a lot of traffic, which can cause associations to fail.

The following is sample output for this command:

Example:

```
Router# debug ip sctp api

*Mar  1 00:31:14.211: SCTP: sctp_send: Assoc ID: 1
*Mar  1 00:31:14.211: SCTP:            stream num: 10
*Mar  1 00:31:14.211: SCTP:            bptr: 62EE332C, dptr: 4F7B598
*Mar  1 00:31:14.211: SCTP:            datalen: 100
*Mar  1 00:31:14.211: SCTP:            context: 1
*Mar  1 00:31:14.211: SCTP:            lifetime: 0
*Mar  1 00:31:14.211: SCTP:            unorder flag: FALSE
```

```
*Mar  1 00:31:14.211: SCTP:            bundle flag: TRUE
*Mar  1 00:31:14.211: SCTP: sctp_send successful return
*Mar  1 00:31:14.211: SCTP: sctp_receive: Assoc ID: 1
*Mar  1 00:31:14.215: SCTP:               max data len: 100
.
.
.
```

Step 2 **debug ip sctp congestion**

The**debug ip sctp congestion** command displays various events related to calculating the current congestion parameters, including congestion window (cwnd) values per destination address and local and remote receiver window (rwnd) parameters. Information is displayed when bundling and sending data chunks, indicating the current cwnd and rwnd values and remote rwnd values, thus showing when data can or cannot be sent or bundled. When chunks are acknowledged by the remote peer, the number of bytes outstanding and remote rwnd values are updated.

Information is also displayed when new chunks are received, thus decreasing the local rwnd space, and when chunks are freed because the upper-layer protocol (ULP) is receiving datagrams from SCTP and thus freeing local rwnd space. The following is sample output for this command:

Example:

```
Router# debug ip sctp congestion

SCTP: Assoc 0: Slow start 10.6.0.4, cwnd 3000
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7800
SCTP: Assoc 0: Free chunks, local rwnd 9000
SCTP: Assoc 0: Data chunks rcvd, local rwnd 8200
SCTP: Assoc 0: Add Sack, local a_rwnd 8200
SCTP: Assoc 0: Free chunks, local rwnd 9000
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7800
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7000
SCTP: Assoc 0: Add Sack, local a_rwnd 7000
SCTP: Assoc 0: Free chunks, local rwnd 9000
SCTP: Assoc 0: Bundle for 10.5.0.4, rem rwnd 14000, cwnd 19500, outstand 0
SCTP: Assoc 0: Bundled 12 chunks, remote rwnd 12800, outstand 1200
SCTP: Assoc 0: Bundling data, next chunk dataLen (100) > remaining mtu size
SCTP: Assoc 0: Bundle for 10.5.0.4, rem rwnd 12800, cwnd 19500, outstand 1200
SCTP: Assoc 0: Bundled 12 chunks, remote rwnd 11600, outstand 2400
SCTP: Assoc 0: Bundling data, next chunk dataLen (100) > remaining mtu size
SCTP: Assoc 0: Bundle for 10.5.0.4, rem rwnd 11600, cwnd 19500, outstand 2400
SCTP: Assoc 0: Bundled 12 chunks, remote rwnd 10400, outstand 3600
SCTP: Assoc 0: Bundling data, next chunk dataLen (100) > remaining mtu size
SCTP: Assoc 0: Bundle for 10.5.0.4, rem rwnd 10400, cwnd 19500, outstand 3600
SCTP: Assoc 0: Bundled 4 chunks, remote rwnd 10000, outstand 4000
SCTP: Assoc 0: No additional chunks waiting.
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7800
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7000
SCTP: Assoc 0: Add Sack, local a_rwnd 7000
SCTP: Assoc 0: Chunk A22F3B45 ack'd, dest 10.5.0.4, outstanding 3900
SCTP: Assoc 0: Chunk A22F3B46 ack'd, dest 10.5.0.4, outstanding 3800
SCTP: Assoc 0: Chunk A22F3B47 ack'd, dest 10.5.0.4, outstanding 3700
SCTP: Assoc 0: Chunk A22F3B48 ack'd, dest 10.5.0.4, outstanding 3600
SCTP: Assoc 0: Chunk A22F3B49 ack'd, dest 10.5.0.4, outstanding 3500
SCTP: Assoc 0: Chunk A22F3B4A ack'd, dest 10.5.0.4, outstanding 3400
SCTP: Assoc 0: Chunk A22F3B4B ack'd, dest 10.5.0.4, outstanding 3300
SCTP: Assoc 0: Chunk A22F3B4C ack'd, dest 10.5.0.4, outstanding 3200
SCTP: Assoc 0: Chunk A22F3B4D ack'd, dest 10.5.0.4, outstanding 3100
SCTP: Assoc 0: Chunk A22F3B4E ack'd, dest 10.5.0.4, outstanding 3000
SCTP: Assoc 0: Chunk A22F3B4F ack'd, dest 10.5.0.4, outstanding 2900
SCTP: Assoc 0: Chunk A22F3B50 ack'd, dest 10.5.0.4, outstanding 2800
SCTP: Assoc 0: Chunk A22F3B51 ack'd, dest 10.5.0.4, outstanding 2700
SCTP: Assoc 0: Chunk A22F3B52 ack'd, dest 10.5.0.4, outstanding 2600
```

```
SCTP: Assoc 0: Chunk A22F3B53 ack'd, dest 10.5.0.4, outstanding 2500
SCTP: Assoc 0: Chunk A22F3B54 ack'd, dest 10.5.0.4, outstanding 2400
SCTP: Assoc 0: Chunk A22F3B55 ack'd, dest 10.5.0.4, outstanding 2300
SCTP: Assoc 0: Chunk A22F3B56 ack'd, dest 10.5.0.4, outstanding 2200
```

Step 3 **debug ip sctp init**

The **debug ip sctp init** command shows datagrams and other information related to the initializing of new associations. All initialization chunks are shown, including the INIT, INIT_ACK, COOKIE_ECHO, and COOKIE_ACK chunks. This debug command can be used to see the chunks associated with any initialization sequence, but does not display data chunks sent once the association is established. Therefore, it is safe to use in a live system that has traffic flowing when you have trouble with associations that fail and have to be reestablished.

Example:

```
Router# debug ip sctp init

*Mar  1 00:53:07.279: SCTP Test: Attempting to open assoc to remote port 8787...assoc ID is 0
*Mar  1 00:53:07.279: SCTP: Process Assoc Request
*Mar  1 00:53:07.279: SCTP: Assoc 0: dest addr list:
*Mar  1 00:53:07.279: SCTP:                 addr 10.5.0.4
*Mar  1 00:53:07.279: SCTP:                 addr 10.6.0.4
*Mar  1 00:53:07.279:
...
*Mar  1 00:53:13.279: SCTP: Assoc 0: Send Init
*Mar  1 00:53:13.279: SCTP:        INIT_CHUNK, len 42
*Mar  1 00:53:13.279: SCTP:        Initiate Tag: B4A10C4D, Initial TSN: B4A10C4D, rwnd 9000
*Mar  1 00:53:13.279: SCTP:        Streams Inbound: 13, Outbound: 13
*Mar  1 00:53:13.279: SCTP:        IP Addr: 10.1.0.2
*Mar  1 00:53:13.279: SCTP:        IP Addr: 10.2.0.2
*Mar  1 00:53:13.279: SCTP:        Supported addr types: 5
*Mar  1 00:53:13.307: SCTP: Process Init
*Mar  1 00:53:13.307: SCTP:        INIT_CHUNK, len 42
*Mar  1 00:53:13.307: SCTP:        Initiate Tag: 3C2D8327, Initial TSN: 3C2D8327, rwnd 18000
*Mar  1 00:53:13.307: SCTP:        Streams Inbound: 13, Outbound: 13
*Mar  1 00:53:13.307: SCTP:        IP Addr: 10.5.0.4
*Mar  1 00:53:13.307: SCTP:        IP Addr: 10.6.0.4
*Mar  1 00:53:13.307: SCTP:        Supported addr types: 5
*Mar  1 00:53:13.307: SCTP: Assoc 0: Send InitAck
*Mar  1 00:53:13.307: SCTP:        INIT_ACK_CHUNK, len 124
*Mar  1 00:53:13.307: SCTP:        Initiate Tag: B4A10C4D, Initial TSN: B4A10C4D, rwnd 9000
*Mar  1 00:53:13.307: SCTP:        Streams Inbound: 13, Outbound: 13
*Mar  1 00:53:13.307: SCTP:        Responder cookie len 88
*Mar  1 00:53:13.307: SCTP:        IP Addr: 10.1.0.2
*Mar  1 00:53:13.307: SCTP:        IP Addr: 10.2.0.2
*Mar  1 00:53:13.311: SCTP: Assoc 0: Process Cookie
*Mar  1 00:53:13.311: SCTP:        COOKIE_ECHO_CHUNK, len 88
*Mar  1 00:53:13.311: SCTP: Assoc 0: dest addr list:
*Mar  1 00:53:13.311: SCTP:                 addr 10.5.0.4
*Mar  1 00:53:13.311: SCTP:                 addr 10.6.0.4
*Mar  1 00:53:13.311:
*Mar  1 00:53:13.311: SCTP: Instance 0 dest addr list:
*Mar  1 00:53:13.311: SCTP:                 addr 10.5.0.4
*Mar  1 00:53:13.311: SCTP:                 addr 10.6.0.4
*Mar  1 00:53:13.311:
*Mar  1 00:53:13.311: SCTP: Assoc 0: Send CookieAck
*Mar  1 00:53:13.311: SCTP:        COOKIE_ACK_CHUNK
```

Step 4 **debug ip sctp multihome**

The **debug ip sctp multihome** command shows the source and destination of datagrams in order to monitor the use of the multihome addresses. More than one IP address parameter can be included in an INIT chunk when the INIT sender

is multihomed. Datagrams should mostly be sent to the primary destination addresses unless the network is experiencing problems, in which case the datagrams can be sent to the secondary addresses.

Caution The **debug ip sctp multihome** command generates one debug line for each datagram sent or received. It should be used with extreme caution in a live network.

The following is sample output for this command:

Example:

```
Router# debug ip sctp multihome

SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 476
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP: Assoc 0: Send Data to dest 10.5.0.4
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 476
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 476
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP: Assoc 0: Send Data to dest 10.5.0.4
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 476
SCTP: Rcvd s=10.6.0.4  8787, d=10.2.0.2  8787, len 44
SCTP: Sent:  Assoc 0: s=10.2.0.2  8787, d=10.6.0.4  8787, len 44
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 476
```

Step 5 **debug ip sctp performance**

The **debug ip sctp performance** command reveals the average number of chunks and datagrams being sent and received per second. Once enabled, the **debug ip sctp performance** command displays this information once every 10 seconds. Note that the averages are cumulative since the last time the statistics were cleared and so may not accurately reflect the number of datagrams and chunks currently being sent and received.

In the following example, when the performance debug was first enabled, it showed a very low rate of traffic. However, it was expected that these numbers were not accurate, so a **clear ip sctp** command was executed. The average numbers adjusted quickly to reflect the accurate amount of flowing traffic.

Example:

```
Router# debug ip sctp performance

SCTP Sent: SCTP Dgrams 5, Chunks 28, Data Chunks 29, ULP Dgrams 29
SCTP Rcvd: SCTP Dgrams 7, Chunks 28, Data Chunks 29, ULP Dgrams 29
Chunks Discarded: 0,  Retransmitted 0
```

```
SCTP Sent: SCTP Dgrams 6, Chunks 29, Data Chunks 30, ULP Dgrams 30
SCTP Rcvd: SCTP Dgrams 7, Chunks 29, Data Chunks 30, ULP Dgrams 30
Chunks Discarded: 0,  Retransmitted 0
SCTP Sent: SCTP Dgrams 6, Chunks 29, Data Chunks 31, ULP Dgrams 31
SCTP Rcvd: SCTP Dgrams 7, Chunks 30, Data Chunks 31, ULP Dgrams 31
Chunks Discarded: 0,  Retransmitted 0
SCTP Sent: SCTP Dgrams 6, Chunks 30, Data Chunks 31, ULP Dgrams 31
SCTP Rcvd: SCTP Dgrams 7, Chunks 31, Data Chunks 32, ULP Dgrams 31
Chunks Discarded: 0,  Retransmitted 0
SCTP Sent: SCTP Dgrams 6, Chunks 31, Data Chunks 32, ULP Dgrams 32
SCTP Rcvd: SCTP Dgrams 7, Chunks 32, Data Chunks 32, ULP Dgrams 32
Chunks Discarded: 0,  Retransmitted 0

Router# clear ip sctp statistics

SCTP Sent: SCTP Dgrams 30, Chunks 210, Data Chunks 199, ULP Dgrams 201
SCTP Rcvd: SCTP Dgrams 30, Chunks 208, Data Chunks 198, ULP Dgrams 198
Chunks Discarded: 0,  Retransmitted 0
SCTP Sent: SCTP Dgrams 30, Chunks 210, Data Chunks 199, ULP Dgrams 200
SCTP Rcvd: SCTP Dgrams 30, Chunks 209, Data Chunks 199, ULP Dgrams 199
Chunks Discarded: 0,  Retransmitted 0
SCTP Sent: SCTP Dgrams 30, Chunks 211, Data Chunks 200, ULP Dgrams 199
SCTP Rcvd: SCTP Dgrams 30, Chunks 209, Data Chunks 198, ULP Dgrams 198
Chunks Discarded: 0,  Retransmitted 0
```

Step 6 **debug ip sctp rcvchunks**

The **debug ip sctp rcvchunks** command displays information about chunks that are received. It shows the stream number, sequence number, chunk length, and chunk transmission sequence number (TSN) for each chunk received, and whether the chunk is for a new datagram or is part of a datagram that is already being reassembled. The command output shows whether the datagram is complete after receiving this chunk or not and, if it is complete, whether it is in sequence within the specified stream and can be delivered to the ULP. It shows the SACKs that are sent back to the remote, indicating the cumulative TSN acknowledged, the number of fragments included, and that the datagram is received by the ULP.

Caution The **debug ip sctp rcvchunks** command generates multiple debug lines for each chunk received. It should be used with extreme caution in a live network.

In the following example, a segmented datagram is received in two chunks, for stream 0 and sequence number 0. The length of the first chunk is1452, and the second is 1 byte. The first chunk indicates that it is for a new datagram, but the second chunk indicates that it is part of an existing datagram that is already being reassembled. When the first chunk is processed, it is noted to be in sequence, but is not complete and so cannot be delivered yet. When the second chunk is received, the datagram is both in sequence and complete. The application receives the datagram, and a SACK is shown to acknowledge that both chunks were received with no missing chunks indicated (that is, with no fragments).

Example:

```
Router# debug ip sctp rcvchunks

SCTP: Assoc 0: New chunk (0/0/1452/2C33D822) for new dgram (0)
SCTP: Assoc 0: dgram (0) is in seq
SCTP: Assoc 0: Add Sack Chunk, CumTSN=2C33D822, numFrags=0
SCTP: Assoc 0: New chunk (0/0/1/2C33D823) for existing dgram (0)
SCTP: Assoc 0: dgram (0) is complete
SCTP: Assoc 0: ApplRecv chunk 0/0/1452/2C33D822
SCTP: Assoc 0: ApplRecv chunk 0/0/1/2C33D823
SCTP: Assoc 0: Add Sack Chunk, CumTSN=2C33D823, numFrags=0
```

Step 7 **debug ip sctp rto**

The **debug ip sctp rto** command shows any adjustments that are made to the retransmission (retrans) timeout value due either to retransmission of data chunks or to unacknowledged heartbeats.

Caution The **debug ip sctp rto** command can generate a great deal of output. It should be used with extreme caution in a live network.

In the following example, there is only one destination address available. Each time the chunk needs to be retransmitted, the retransmission timeout (RTO) value is doubled.

Example:

```
Router# debug ip sctp rto

SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 2000 ms
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 4000 ms
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 8000 ms
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 16000 ms
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 32000 ms
```

Step 8 debug ip sctp segments

The **debug ip sctp segments** output shows every datagram that is sent or received and the chunks that are contained in each. The segment debug command has two forms: simple and verbose. This is the simple form of the segment output, and it shows basic information for each chunk type. See the **debug ip sctp segmentv** command for the verbose form of this output.

Caution The **debug ip sctp segments** command generates several lines of output for each datagram sent or received. It should be used with extreme caution in a live network.

The following output shows an example in which an association is established, a few heartbeats are sent, the remote endpoint fails, and the association is restarted.

Example:

```
Router# debug ip sctp segments

SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 56
SCTP:        INIT_CHUNK, Tag: 3C72A02A, TSN: 3C72A02A
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 56
SCTP:        INIT_CHUNK, Tag: 13E5AD6C, TSN: 13E5AD6C
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 136
SCTP:        INIT_ACK_CHUNK, Tag: 3C72A02A, TSN: 3C72A02A
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 100
SCTP:        COOKIE_ECHO_CHUNK, len 88
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 16
SCTP:        COOKIE_ACK_CHUNK
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 52
SCTP:        HEARTBEAT_CHUNK
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 52
SCTP:        HEARTBEAT_CHUNK
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 52
SCTP:        HEARTBEAT_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 56
SCTP:        INIT_CHUNK, Tag: 4F2D8235, TSN: 4F2D8235
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 136
SCTP:        INIT_ACK_CHUNK, Tag: 7DD7E424, TSN: 7DD7E424
```

```
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 100
SCTP:        COOKIE_ECHO_CHUNK, len 88
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 16
SCTP:        COOKIE_ACK_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 144
SCTP:        SACK_CHUNK, TSN ack: 7DD7E423, rwnd 18000, num frags 0
SCTP:        DATA_CHUNK, 4/0/100/4F2D8235
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP:        SACK_CHUNK, TSN ack: 4F2D8235, rwnd 8900, num frags 0
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 128
SCTP:        DATA_CHUNK, 4/0/100/7DD7E424
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP:        SACK_CHUNK, TSN ack: 7DD7E424, rwnd 17900, num frags 0
SCTP: Recv:  Assoc 0: s=10.6.0.4  8787, d=10.2.0.2  8787, len 44
SCTP:        HEARTBEAT_CHUNK
SCTP: Sent:  Assoc 0: s=10.2.0.2  8787, d=10.6.0.4  8787, len 44
SCTP:        HEARTBEAT_ACK_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 128
SCTP:        DATA_CHUNK, 7/0/100/4F2D8236
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 144
SCTP:        SACK_CHUNK, TSN ack: 4F2D8236, rwnd 9000, num frags 0
SCTP:        DATA_CHUNK, 7/0/100/7DD7E425
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP:        SACK_CHUNK, TSN ack: 7DD7E424, rwnd 18000, num frags 0
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP:        SACK_CHUNK, TSN ack: 7DD7E425, rwnd 17900, num frags 0
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 128
SCTP:        DATA_CHUNK, 4/1/100/4F2D8237
```

Step 9 **debug ip sctp segmentv**

The **debug ip sctp segmentv** command output shows every datagram that is sent or received and the chunks that are contained in each. This is the verbose form of the output, and it shows detailed information for each chunk type (see the **debug ip sctp segments** command for the simple form output).

Caution The **debug ip sctp segmentv** command generates multiple lines of output for each datagram sent and received. It should be used with extreme caution in a live network.

The following output shows an example in which an association is established, a few heartbeats are sent, the remote endpoint fails, and the association is restarted.

Example:

```
Router# debug ip sctp segmentv

SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 56, ver tag 0
SCTP:        INIT_CHUNK, len 42
SCTP:        Initiate Tag: B131ED6A, Initial TSN: B131ED6A, rwnd 9000
SCTP:        Streams Inbound: 13, Outbound: 13
SCTP:        IP Addr: 10.1.0.2
SCTP:        IP Addr: 10.2.0.2
SCTP:        Supported addr types: 5
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 56, ver tag 0
SCTP:        INIT_CHUNK, len 42
SCTP:        Initiate Tag: 5516B2F3, Initial TSN: 5516B2F3, rwnd 18000
SCTP:        Streams Inbound: 13, Outbound: 13
SCTP:        IP Addr: 10.5.0.4
SCTP:        IP Addr: 10.6.0.4
SCTP:        Supported addr types: 5
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 136, ver tag 5516B2F3
SCTP:        INIT_ACK_CHUNK, len 124
SCTP:        Initiate Tag: B131ED6A, Initial TSN: B131ED6A, rwnd 9000
```

```
SCTP:        Streams Inbound: 13, Outbound: 13
SCTP:        Responder cookie len 88
SCTP:        IP Addr: 10.1.0.2
SCTP:        IP Addr: 10.2.0.2
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 100, ver tag B131ED6A
SCTP:        COOKIE_ECHO_CHUNK, len 88
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 16, ver tag 5516B2F3
SCTP:        COOKIE_ACK_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 144, ver tag B131ED6A
SCTP:        SACK_CHUNK, len 16
SCTP:        TSN ack: (0xB131ED69)
SCTP:        Rcv win credit: 18000
SCTP:        Num frags: 0
SCTP:        DATA_CHUNK, flags 3, chunkLen 116
SCTP:        DATA_CHUNK, 0/0/100/5516B2F3
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28, ver tag 5516B2F3
SCTP:        SACK_CHUNK, len 16
SCTP:        TSN ack: (0x5516B2F3)
SCTP:        Rcv win credit: 8900
SCTP:        Num frags: 0
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 128, ver tag 5516B2F3
SCTP:        DATA_CHUNK, flags 3, chunkLen 116
SCTP:        DATA_CHUNK, 0/0/100/B131ED6A
SCTP: Recv:  Assoc 0: s=10.6.0.4  8787, d=10.2.0.2  8787, len 44, ver tag B131ED6A
SCTP:        HEARTBEAT_CHUNK
SCTP: Sent:  Assoc 0: s=10.2.0.2  8787, d=10.6.0.4  8787, len 44, ver tag 5516B2F3
SCTP:        HEARTBEAT_ACK_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 28, ver tag B131ED6A
SCTP:        SACK_CHUNK, len 16
```

Step 10 debug ip sctp signal

The **debug ip sctp signal** command shows signals that are sent from SCTP to the application or ULP. These signals inform the ULP of state transitions for associations or destination addresses. There is also a signal sent to the ULP when new data is available to be received, but this signal is not shown in the example output below because it occurs infrequently. This debug command can be used to see if the current associations are stable. Because it does not generate output except on state transitions, it is safe to use in a live environment. It still should be used with caution, however, depending on the number of associations being handled by the system and the stability of the network.

Step 11 debug ip sctp state

The **debug ip sctp state** command is often used at the same time as the **debug ip sctp signal** command. Using the two commands together gives good insight into the stability of associations.

In the following example, a new association is requested and established. The peer then restarts the association and notes that the association failed and is being reestablished. The local peer then indicates that the association has failed because it has tried to retransmit the specified chunk more than the maximum number of times without success. As a result, the association fails (because of communication loss) and is terminated. The ULP requests that the association be attempted again, and this attempt succeeds. A shutdown is then received from the remote peer, and the local peer enters the shutdown acknowledge sent state, which is followed by the association being terminated. Again, another association attempt is made and succeeds.

Example:

```
Router# debug ip sctp signal
Router# debug ip sctp state

<new assoc attempt>
00:20:08: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:20:15: SCTP: Assoc 0:  state COOKIE_WAIT -> ESTABLISHED
00:20:15: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
```

```
00:21:03: SCTP: Assoc 0: Restart rcvd from peer
00:21:03: SCTP: Assoc 0: Sent ASSOC_RESTART signal
00:21:04: SCTP: Assoc 0: chunk 62EA7F40 retransmitted more than max times, failing assoc
00:21:04: SCTP: Assoc 0: Sent ASSOC_FAILED signal, reason: SCTP_COMM_LOST
00:21:04: SCTP: Assoc 0: Sent ASSOC_TERMINATE signal
00:21:04: SCTP: Assoc 0:  state ESTABLISHED -> CLOSED
<new assoc attempt>
00:21:04: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:21:04: SCTP: Assoc 0:  state COOKIE_WAIT -> COOKIE_ECHOED
00:21:04: SCTP: Assoc 0:  state COOKIE_ECHOED -> ESTABLISHED
00:21:04: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
00:21:04: SCTP: Assoc 0: Sent TERMINATE_PENDING signal
00:21:04: SCTP: Assoc 0:  state ESTABLISHED -> SHUTDOWN_ACKSENT
00:21:04: SCTP: Assoc 0: Sent ASSOC_TERMINATE signal
00:21:04: SCTP: Assoc 0:  state SHUTDOWN_ACKSENT -> CLOSED
<new assoc attempt>
00:21:04: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:21:04: SCTP: Assoc 0:  state COOKIE_WAIT -> COOKIE_ECHOED
00:21:04: SCTP: Assoc 0:  state COOKIE_ECHOED -> ESTABLISHED
00:21:04: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
```

Step 12 **debug ip sctp sndchunks**

The **debug ip sctp sndchunks** command shows the following types of information about all chunks that are being sent to remote SCTP peers:

- Application send requests from the local SCTP peer
- Chunks being bundled and sent to the remote peer
- Processing of the SACKs from the remote peer, indicating which chunks were successfully received
- Chunks that are marked for retransmission

Caution The **debug ip sctp sndchunks** command generates large amounts of data if there is any significant amount of traffic flowing. It should be used with extreme caution in live networks.

Example:

```
Router# debug ip sctp sndchunks

SCTP: Assoc 0: ApplSend, chunk: 0/10412/100/A23134F8 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 5/10443/100/A23134F9 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 5/10448/100/A231355C to 10.5.0.4
SCTP: Assoc 0: Set oldest chunk for dest 10.5.0.4 to TSN A23134F8
SCTP: Assoc 0: Bundling data, added 0/10412/100/A23134F8, outstanding 100
SCTP: Assoc 0: Bundling data, added 5/10443/100/A23134F9, outstanding 200
SCTP: Assoc 0: Bundling data, added 4/10545/100/A23134FA, outstanding 300
SCTP: Assoc 0: Bundling data, added 10/10371/100/A23134FB, outstanding 400
SCTP: Assoc 0: Bundling data, added 11/10382/100/A23134FC, outstanding 500
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A231350F, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A2313510
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A2313527, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A2313528
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A231353F, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A2313540
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A2313557, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A2313558
SCTP: Assoc 0: ApplSend, chunk: 10/10385/100/A23135BE to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 8/10230/100/A23135BF to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 5/10459/100/A23135C0 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 4/10558/100/A23135C1 to 10.5.0.4
```

```
SCTP: Assoc 0: Set oldest chunk for dest 10.5.0.4 to TSN A231355D
SCTP: Assoc 0: Bundling data, added 5/10449/100/A231355D, outstanding 100
SCTP: Assoc 0: Bundling data, added 3/10490/100/A231355E, outstanding 200
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A23135A4, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A23135A5
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A23135BC, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A23135BD
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A23135C1, numFrags=0
SCTP: Assoc 0: ApplSend, chunk: 5/10460/100/A23135C2 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 5/10461/100/A23135C3 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 11/10403/100/A2313626 to 10.5.0.4
SCTP: Assoc 0: Set oldest chunk for dest 10.5.0.4 to TSN A23135C2
SCTP: Assoc 0: Bundling data, added 5/10460/100/A23135C2, outstanding 100
SCTP: Assoc 0: Bundling data, added 5/10461/100/A23135C3, outstanding 200
SCTP: Assoc 0: Bundling data, added 5/10462/100/A23135C4, outstanding 300
SCTP: Assoc 0: Bundling data, added 4/10559/100/A23135C5, outstanding 400
SCTP: Assoc 0: Bundling data, added 4/10560/100/A23135C6, outstanding 500
SCTP: Assoc 0: Bundled 12 chunk(s) in next dgram to 10.5.0.4
SCTP: Assoc 0: Bundling data, added 1/10418/100/A2313622, outstanding 9700
SCTP: Assoc 0: Bundling data, added 3/10502/100/A2313623, outstanding 9800
SCTP: Assoc 0: Bundling data, added 7/10482/100/A2313624, outstanding 9900
SCTP: Assoc 0: Bundling data, added 3/10503/100/A2313625, outstanding 10000
SCTP: Assoc 0: Bundling data, added 11/10403/100/A2313626, outstanding 10100
SCTP: Assoc 0: Bundled 5 chunk(s) in next dgram to 10.5.0.4
SCTP: Assoc 0: Mark chunk A23135C2 for retrans
SCTP: Assoc 0: Mark chunk A23135C3 for retrans
SCTP: Assoc 0: Mark chunk A23135C4 for retrans
SCTP: Assoc 0: Mark chunk A23135C5 for retrans
SCTP: Assoc 0: Mark chunk A23135C6 for retrans
SCTP: Assoc 0: Mark chunk A23135C7 for retrans
SCTP: Assoc 0: Mark chunk A23135C8 for retrans
SCTP: Assoc 0: Mark chunk A23135C9 for retrans
SCTP: Assoc 0: Mark chunk A23135CA for retrans
SCTP: Assoc 0: Bundled 6 chunk(s) in next dgram to 10.6.0.4
SCTP: Assoc 0: Mark chunk A23135C2 for retrans
SCTP: Assoc 0: Mark chunk A23135C3 for retrans
SCTP: Assoc 0: Mark chunk A23135C4 for retrans
```

Step 13 debug ip sctp timer

The **debug ip sctp timer** command displays information about all started, stopped, and triggering SCTP timers. After they have been started, many SCTP timers are not restarted until they expire or are stopped. For these timers, the first call succeeds in starting the timer, and subsequent calls do nothing until the timer either expires or is stopped. For example, the retransmission timer is started when the first chunk is sent, but then is not started again for subsequent chunks when there is outstanding data.

Caution The **debug ip sctp timer** command generates a significant amount of output. It should be used with extreme caution in a live network.

The following example shows output from the **debug ip sctp timer** command:

Example:

```
Router# debug ip sctp timer

SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Timer BUNDLE triggered
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
```

```
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Stopping RETRANS timer for destaddr 10.5.0.4
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Assoc 0: Stopping RETRANS timer for destaddr 10.5.0.4
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Stopping CUMSACK timer
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
```

Step 14 **debug ip sctp warnings**

The **debug ip sctp warnings** command displays information on any unusual situation that is encountered. These situations may or may not indicate problems, depending on the particulars of the situation. Below are some examples of events or conditions that are flagged as warnings.

Example:

```
Router# debug ip sctp warnings

SCTP: Assoc 0: No cookie in InitAck, discarding
SCTP: Assoc 0: Incoming INIT_ACK: inbound streams reqd 15, allowed 13
SCTP: Assoc 0: Incoming INIT_ACK request: outbound streams req'd 13, allowed 1
SCTP: Assoc 0: Remote verification tag in init ack is zero, discarding
SCTP: Remote verification tag in init is zero, discarding
SCTP: Assoc 0: Rwnd less than min allowed (1500) in incoming INITACK, rcvd 0
SCTP: Assoc 0: Rwnd less than min allowed (1500) in incoming INITACK, rcvd 1499
SCTP: Rwnd in INIT too small (0), discarding
SCTP: Rwnd in INIT too small (1499), discarding
SCTP: Unknown INIT param 16537 (0x4099), length 8
SCTP: Assoc 0: Unknown INITACK param 153 (0x99), length 8
SCTP: Assoc 0: No cookie in InitAck, discarding
SCTP: Assoc 0: No cookie in InitAck, discarding
SCTP: Processing INIT, invalid param len 0, discarding...
SCTP: Assoc 0: Processing INITACK, invalid param len 0, discarding...
```

Configuration Examples for SCTP

Example: Defining SCTP Authentication Parameters

The following example shows how to define SCTP data chunks that the client requires be authenticated and to configure SCTP to automatically send ASCONF chunks in response to IP address changes on the router:

```
Router# configure terminal
Router(config)# ip sctp authenticate data
Router(config)# ip sctp asconf auto
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Applications that make use of SCTP	• PRI Backhaul Using the Stream Control Transmission Protocol and the ISDN Q.921 User Adaptation Layer • Netflow Reliable Export with SCTP section in the *Cisco IOS Netflow Configuration Guide*
SCTP commands	*Cisco IOS IP Application Services Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 2960	Stream Control Transmission Protocol (SCTP)

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for SCTP

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2: Feature Information for SCTP

Feature Name	Releases	Feature Information
SCTP, Release 1	12.2(4)T	Stream Control Transmission Protocol (SCTP) is a reliable datagram-oriented IP transport protocol specified by RFC 2960. In 12.2(4)T, this feature was introduced. The following commands were introduced or modified by this feature: **clear ip sctp statistics**, **debug ip sctp api**, **debug ip sctp congestion**, **debug ip sctp init**, **debug ip sctp multihome**, **debug ip sctp performance**, **debug ip sctp rcvchunks**,**debug ip sctp rto**, **debug ip sctp segments**, **debug ip sctp segmentv**, **debug ip sctp signal**, **debug ip sctp sndchunks**,**debug ip sctp state**,**debug ip sctp timer**, **debug ip sctp warnings**, **show ip sctp association list**, **show ip sctp association parameters**, **show ip sctp association statistics**, **show ip sctp errors**, **show ip sctp instances**, **show ip sctp statistics**.
SCTP Release 2	12.2(8)T	SCTP Release 2 introduced updated output for SCTP commands. The following commands were modified by this feature: **show ip sctp association paramters**, **show ip sctp association statistics**.

Feature Name	Releases	Feature Information
SCTP Show/Clear CLI Enhancements	12.4(11)T	The SCTP Show/Clear CLI Enhancements feature provides access to additional SCTP information that can help with troubleshooting potential problems. These enhancements also make the updated SCTP show and clear commands consistent with the CLI of other transport protocols. The following commands were replaced by this feature: **clear ip sctp statistics**, **show ip sctp association list**, **show ip sctp association parameters**, **show ip sctp association statistics**, **show ip sctp errors**, **show ip sctp instances**, **show ip sctp statistics**. The following commands were introduced by this feature: **clear sctp statistics**, **show sctp association**, **show sctp association list**, **show sctp association parameters**, **show sctp association statistics**, **show sctp errors**, **show sctp instance**,**show sctp instances**,**show sctp instances**, **show sctp statistics**.
SCTP Release 4, Phase 1	12.4(15)T	SCTP Release 4 introduced the SCTP Stream Reset and Authentication features. SCTP Release 4 introduced the SCTP Stream Reset and Authentication features. SCTP stream reset feature enables SCTP to reset stream transport sequence numbers and all stream sequence numbers. SCTP stream reset enables SCTP to: • Dynamically reset a peer's outbound streams • Dynamically reset a local host's outbound stream • Dynamically reset specific numbered streams SCTP Authentication enables SCTP to: • Set up a dynamic shared association key with no shared secret • Allow a shared secret to be combined with an association key • Use the shared association secret to authenticate chunks • Negotiate which chunk types must be authenticated
SCTP Release 4, Phase 2	12.4(20)T	As of Cisco IOS Release 12.4(20)T, SCTP Release 4 introduced the SCTP Add-IP feature. The SCTP Add-IP feature enables the ability to add or delete an IP address for an endpoint of an existing SCTP association and to communicate this change to the remote end. An ADD-IP chunk is sent to the remote end adding or removing the redundant server addresses of the association. The ADD-IP chunk also deletes all addresses of a failed host from an association. The SCTP Add-IP feature also enables an application to programmatically set the primary address for an SCTP association. The following commands are new or modified in this release: **ip sctp authenticate**, **ip sctp asconf**.