

# debug redundancy as5850

To enable specific redundancy-related debug options, use the **debug redundancy as5850** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug redundancy as5850 { fsm | lines | master | mode | rf-client }
```

```
no debug redundancy as5850
```

## Syntax Description

<b>fsm</b>	Finite-state-machine events.
<b>lines</b>	Hardware lines.
<b>master</b>	Master (active rather than standby) route-switch-controller (RSC).
<b>mode</b>	RSC's mode: classic-split or handover-split.
<b>rf-client</b>	Redundancy-related client-application information.

## Defaults

This command is disabled

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.2(2)XB1	This command was introduced.
12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T.

## Usage Guidelines

Use the master form of the command to view redundancy-related debug entries. All debug entries continue to be logged even if you do not specify an option here, and you can always use the **show redundancy debug-log** command to view them.

## Examples

The output from this command consists of event announcements that can be used by authorized troubleshooting personnel.

## Related Commands

Command	Description
<b>show redundancy debug-log</b>	Displays up to 256 debug entries.

# debug resource-pool

To see and trace resource pool management activity, use the **debug resource-pool** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug resource-pool**

**no debug resource-pool**

**Syntax Description** This command has no arguments or keywords.

**Defaults** Disabled

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.0(4)XI	This command was introduced.

**Usage Guidelines** Enter the **debug resource-pool** command to see and trace resource pool management activity. [Table 257](#) describes the resource pooling states.

**Table 257 Resource Pooling States**

State	Description
RM_IDLE	No call activity.
RM_RES_AUTHOR	Call waiting for authorization, message sent to authentication, authorization, and accounting (AAA).
RM_RES_ALLOCATING	Call authorized, resource-grp-mgr allocating.
RM_RES_ALLOCATED	Resource allocated, connection acknowledgment sent to signalling state. Call should get connected and become active.
RM_AUTH_REQ_IDLE	Signalling module disconnected call while in RM_RES_AUTHOR. Waiting for authorization response from AAA.
RM_RES_REQ_IDLE	Signalling module disconnected call while in RM_RES_ALLOCATING. Waiting for resource allocation response from resource-group manager.
RM_DNIS_AUTHOR	An intermediate state before proceeding with Route Processor Module (RPM) authorization.
RM_DNIS_AUTH_SUCCEEDED	Dialed number identification service (DNIS) authorization succeeded.
RM_DNIS_RES_ALLOCATED	DNIS resource allocated.

**Table 257 Resource Pooling States (continued)**

State	Description
RM_DNIS_AUTH_REQ_IDLE	DNIS authorization request idle.
RM_DNIS_AUTHOR_FAIL	DNIS authorization failed.
RM_DNIS_RES_ALLOC_SUCCESS	DNIS resource allocation succeeded.
RM_DNIS_RES_ALLOC_FAIL	DNIS resource allocation failed.
RM_DNIS_RPM_REQUEST	DNIS resource pool management requested.

You can use the resource pool state to isolate problems. For example, if a call fails authorization in the RM\_RES\_AUTHOR state, investigate further with AAA authorization debugs to determine whether the problem lies in the resource-pool manager, AAA, or dispatcher.

### Examples

The following example shows different instances where you can use the **debug resource-pool** command:

```
Router# debug resource-pool

RM general debugging is on

Router# show debug

General OS:
  AAA Authorization debugging is on
Resource Pool:
  resource-pool general debugging is on
Router #
Router #ping 21.1.1.10
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 21.1.1.10, timeout is 2 seconds:
*Jan  8 00:10:30.358: RM state:RM_IDLE event:DIALER_INCALL DS0:0:0:0:1
*Jan  8 00:10:30.358: RM: event incoming call

/* An incoming call is received by RM */

*Jan  8 00:10:30.358: RM state:RM_DNIS_AUTHOR event:RM_DNIS_RPM_REQUEST
DS0:0:0:0:1

/* Receives an event notifying to proceed with RPM authorization while
in DNIS authorization state */

*Jan  8 00:10:30.358: RM:RPM event incoming call
*Jan  8 00:10:30.358: RPM profile cp1 found

/* A customer profile "cp1" is found matching for the incoming call, in
the local database */

*Jan  8 00:10:30.358: RM state:RM_RPM_RES_AUTHOR
event:RM_RPM_RES_AUTHOR_SUCCESS DS0:0:0:0:1

/* Resource authorization success event received while in resource
authorization state*/

*Jan  8 00:10:30.358: Allocated resource from res_group isdn1
*Jan  8 00:10:30.358: RM:RPM profile "cp1", allocated resource "isdn1"
successfully
```

```

*Jan  8 00:10:30.358: RM state:RM_RPM_RES_ALLOCATING
event:RM_RPM_RES_ALLOC_SUCCESS DS0:0:0:0:1

/* Resource allocation success event received while attempting to
allocate a resource */
*Jan  8 00:10:30.358: Se0:1 AAA/ACCT/RM: doing resource-allocated
(local) (nothing to do)
*Jan  8 00:10:30.366: %LINK-3-UPDOWN: Interface Serial0:1, changed state
to up
*Jan  8 00:10:30.370: %LINK-3-UPDOWN: Interface Serial0:1, changed state
to down
*Jan  8 00:10:30.570: Se0:1 AAA/ACCT/RM: doing resource-update (local)
cpl (nothing to do)
*Jan  8 00:10:30.578: %LINK-3-UPDOWN: Interface Serial0:0, changed
state to up
*Jan  8 00:10:30.582: %DIALER-6-BIND: Interface Serial0:0 bound to
profile Dialer0...
Success rate is 0 percent (0/5)
Router #
*Jan  8 00:10:36.662: %ISDN-6-CONNECT: Interface Serial0:0 is now
connected to 71017
*Jan  8 00:10:52.990: %DIALER-6-UNBIND: Interface Serial0:0 unbound from
profile Dialer0
*Jan  8 00:10:52.990: %ISDN-6-DISCONNECT: Interface Serial0:0
disconnected from 71017 , call lasted 22 seconds
*Jan  8 00:10:53.206: %LINK-3-UPDOWN: Interface Serial0:0, changed state
to down
*Jan  8 00:10:53.206: %ISDN-6-DISCONNECT: Interface Serial0:1
disconnected from unknown , call lasted 22 seconds
*Jan  8 00:10:53.626: RM state:RM_RPM_RES_ALLOCATED event:DIALER_DISCON
DS0:0:0:0:1

/* Received Disconnect event from signalling stack for a call which
has a resource allocated. */

*Jan  8 00:10:53.626: RM:RPM event call drop

/* RM processing the disconnect event */

*Jan  8 00:10:53.626: Deallocated resource from res_group isdn1
*Jan  8 00:10:53.626: RM state:RM_RPM_DISCONNECTING
event:RM_RPM_DISC_ACK DS0:0:0:0:1

/* An intermediate state while the DISCONNECT event is being processed
by external servers, before RM goes back into IDLE state.
*/

```

Table 258 describes the significant fields shown in the display.

**Table 258 debug resource-pool Field Descriptions**

Field	Description
RM state:RM_IDLE	Resource manager state that displays no active calls.
RM state:RM_RES_AUTHOR	Resource authorization state.
RES_AUTHOR_SUCCESS DS0: shelf:slot:port:channel	Actual physical resource that is used
Allocated resource from res_group	Physical resource group that accepts the call.

**Table 258** *debug resource-pool Field Descriptions (continued)*

<b>Field</b>	<b>Description</b>
RM profile <x>, allocated resource <x>	Specific customer profile and resource group names used to accept the call.
RM state: RM_RES_ALLOCATING	Resource manager state that unifies a call with a physical resource.

# debug rif

To display information on entries entering and leaving the routing information field (RIF) cache, use the **debug rif** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rif**

**no debug rif**

## Syntax Description

This command has no arguments or keywords.

## Command Modes

Privileged EXEC

## Usage Guidelines

In order to use the **debug rif** command to display traffic source-routed through an interface, fast switching of source route bridging (SRB) frames must first be disabled with the **no source-bridge route-cache** interface configuration command.

## Examples

The following is sample output from the **debug rif** command:

```

router# debug rif
SDLLC or Local-Ack entry — RIF: U chk da=9000.5a59.04f9,sa=0110.2222.33c1 [4880.3201.00A1.0050] type 8 on
                             static/remote/0
                             RIF: U chk da=0000.3080.4aed,sa=0000.0000.0000 [] type 8 on TokenRing0/0
                             RIF: U add 1000.5a59.04f9 [4880.3201.00A1.0050] type 8
Non-SDLLC or non-Local-Ack entry / RIF: L checking da=0000.3080.4aed, sa=0000.0000.0000
RIF: rcvd TEST response from 9000.5a59.04f9
RIF: U upd da=1000.5a59.04f9,sa=0110.2222.33c1 [4880.3201.00A1.0050]
RIF: rcvd XID response from 9000.5a59.04f9
SR1: sent XID response to 9000.5a59.04f9

```

S2559

The first line of output is an example of a RIF entry for an interface configured for SDLC Logical Link Control (SDLLC) or Local-Ack. [Table 259](#) describes significant fields shown in the display.

**Table 259** debug rif Field Descriptions

Field	Description
RIF:	This message describes RIF debugging output.
U chk	Update checking. The entry is being updated; the timer is set to zero (0).
da=9000.5a59.04f9	Destination MAC address.
sa=0110.2222.33c1	Source MAC address. This field contains values of zero (0000.0000.0000) in a non-SDLLC or non-Local-Ack entry.
[4880.3201.00A1.0050]	RIF string. This field is blank (null RIF) in a non-SDLLC or non-Local-Ack entry.

**Table 259** *debug rif Field Descriptions (continued)*

Field	Description
type 8	Possible values follow: <ul style="list-style-type: none"> <li>• 0—Null entry</li> <li>• 1—This entry was learned from a particular Token Ring port (interface)</li> <li>• 2—Statically configured</li> <li>• 4—Statically configured for a remote interface</li> <li>• 8—This entry is to be aged</li> <li>• 16—This entry (which has been learned from a remote interface) is to be aged</li> <li>• 32—This entry is not to be aged</li> <li>• 64—This interface is to be used by LAN Network Manager (and is not to be aged)</li> </ul>
on static/remote/0	This route was learned from a real Token Ring port, in contrast to a virtual ring.

The following line of output is an example of a RIF entry for an interface that is not configured for SDLLC or Local-Ack:

```
RIF: U chk da=0000.3080.4aed,sa=0000.0000.0000 [] type 8 on TokenRing0/0
```

Notice that the source address contains only zero values (0000.0000.0000), and that the RIF string is null ([ ]). The last element in the entry indicates that this route was learned from a virtual ring, rather than a real Token Ring port.

The following line shows that a new entry has been added to the RIF cache:

```
RIF: U add 1000.5a59.04f9 [4880.3201.00A1.0050] type 8
```

The following line shows that a RIF cache lookup operation has taken place:

```
RIF: L checking da=0000.3080.4aed, sa=0000.0000.0000
```

The following line shows that a TEST response from address 9000.5a59.04f9 was inserted into the RIF cache:

```
RIF: rcvd TEST response from 9000.5a59.04f9
```

The following line shows that the RIF entry for this route has been found and updated:

```
RIF: U upd da=1000.5a59.04f9,sa=0110.2222.33c1 [4880.3201.00A1.0050]
```

The following line shows that an XID response from this address was inserted into the RIF cache:

```
RIF: rcvd XID response from 9000.5a59.04f9
```

The following line shows that the router sent an XID response to this address:

```
SR1: sent XID response to 9000.5a59.04f9
```

Table 260 explains the other possible lines of **debug rif** command output.

**Table 260 Additional debug rif Field Descriptions**

Field	Description
RIF: L Sending XID for <address>	Router/bridge wanted to send a packet to <i>address</i> but did not find it in the RIF cache. It sent an XID explorer packet to determine which RIF it should use. The attempted packet is dropped.
RIF: L No buffer for XID to <address>	Similar to the previous description; however, a buffer in which to build the XID packet could not be obtained.
RIF: U remote rif too small <rif>	Packet's RIF was too short to be valid.
RIF: U rej <address> too big <rif>	Packet's RIF exceeded the maximum size allowed and was rejected. The maximum size is 18 bytes.
RIF: U upd interface <address>	RIF entry for this router/bridge's interface has been updated.
RIF: U ign <address> interface update	RIF entry that would have updated an interface corresponding to one of this router's interfaces.
RIF: U add <address> <rif>	RIF entry for <i>address</i> has been added to the RIF cache.
RIF: U no memory to add rif for <address>	No memory to add a RIF entry for <i>address</i> .
RIF: removing rif entry for <address, type code>	RIF entry for <i>address</i> has been forcibly removed.
RIF: flushed <address>	RIF entry for <i>address</i> has been removed because of a RIF cache flush.
RIF: expired <address>	RIF entry for <i>address</i> has been aged out of the RIF cache.

#### Related Commands

Command	Description
<b>debug list</b>	Filters debugging information on a per-interface or per-access list basis.

# debug route-map ipc

To display a summary of the one-way Inter-process Communications (IPC) messages set from the route processor (RP) to the Versatile Interface Processor (VIP) about NetFlow policy routing when distributed Cisco Express Forwarding (dCEF) is enabled, use the **debug route-map ipc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug route-map ipc**

**no debug route-map ipc**

**Syntax Description** This command has no arguments or keywords.

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.0(3)T	This command was introduced.

**Usage Guidelines** This command is especially helpful for policy routing with dCEF switching.

This command displays a summary of one-way IPC messages from the RP to the VIP about NetFlow policy routing. If you execute this command on the RP, the messages are shown as “Sent.” If you execute this command on the VIP console, the IPC messages are shown as “Received.”

**Examples** The following is sample output from the **debug route-map ipc** command executed at the RP:

```
Router# debug route-map ipc

Routemap related IPC debugging is on

Router# configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)# ip cef distributed

Router(config)#^Z
Router#

RM-IPC: Clean routemap config in slot 0
RM-IPC: Sent clean-all-routemaps; len 12
RM-IPC: Download all policy-routing related routemap config to slot 0
RM-IPC: Sent add routemap test(seq:10); n_len 5; len 17
RM-IPC: Sent add acl 1 of routemap test(seq:10); len 21
RM-IPC: Sent add min 10 max 300 of routemap test(seq:10); len 24
RM-IPC: Sent add preced 1 of routemap test(seq:10); len 17
RM-IPC: Sent add tos 4 of routemap test(seq:10); len 17
RM-IPC: Sent add nexthop 50.0.0.8 of routemap test(seq:10); len 20
RM-IPC: Sent add default nexthop 50.0.0.9 of routemap test(seq:10); len 20
RM-IPC: Sent add interface Ethernet0/0/3(5) of routemap test(seq:10); len 20
```

RM-IPC: Sent add default interface Ethernet0/0/2(4) of routemap test(seq:10); len 20

The following is sample output from the **debug route-map ipc** command executed at the VIP:

VIP-Slot0# **debug route-map ipc**

Routemap related IPC debugging is on

VIP-Slot0#

RM-IPC: Rcvd clean-all-routemaps; len 12

RM-IPC: Rcvd add routemap test(seq:10); n\_len 5; len 17

RM-IPC: Rcvd add acl 1 of routemap test(seq:10); len 21

RM-IPC: Rcvd add min 10 max 300 of routemap test(seq:10); len 24

RM-IPC: Rcvd add preced 1 of routemap test(seq:10); len 17

RM-IPC: Rcvd add tos 4 of routemap test(seq:10); len 17

RP-IPC: Rcvd add nexthop 50.0.0.8 of routemap test(seq:10); len 20

RP-IPC: Rcvd add default nexthop 50.0.0.9 of routemap test(seq:10); len 20

RM-IPC: Rcvd add interface Ethernet0/3 of routemap tes; len 20

RM-IPC: Rcvd add default interface Ethernet0/2 of routemap test(seq:10); len 20

# debug rpms-proc preauth

To enable diagnostic reporting of preauthentication information, use the **debug rpms-proc preauth** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug rpms-proc preauth {all | h323 | sip}
```

```
no debug rpms-proc preauth {all | h323 | sip}
```

Syntax Description		
	<b>all</b>	Provides information for all calls.
	<b>h323</b>	Provides information for H.323 calls.
	<b>sip</b>	Provides information for Session Initiation Protocol (SIP) calls.

**Defaults** No default behavior or values

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.2(11)T	This command was introduced.

**Examples** The following example shows debugging output for two calls. The first is a leg 3 SIP call, and the second is a leg 3 H.323 call:

```
Router# debug rpms-proc preauth all

All RPMS Process preauth tracing is enabled
Feb 10 14:00:07.236: Entering rpms_proc_print_preauth_req

Feb 10 14:00:07.236: Request = 0
Feb 10 14:00:07.236: Preauth id = 8
Feb 10 14:00:07.236: EndPt Type = 1
Feb 10 14:00:07.236: EndPt = 192.168.80.70
Feb 10 14:00:07.236: Resource Service = 1
Feb 10 14:00:07.236: Call_origin = answer
Feb 10 14:00:07.236: Call_type = voip
Feb 10 14:00:07.236: Calling_num = 2220001
Feb 10 14:00:07.236: Called_num = 1120001
Feb 10 14:00:07.236: Protocol = 1
Feb 10 14:00:07.236:rpms_proc_create_node:Created node with preauth_id = 8
Feb 10 14:00:07.236:rpms_proc_send_aaa_req:uid got is 19
Feb 10 14:00:07.240:rpms_proc_preauth_response:Context is for preauth_id 8, aaa_uid 19
Feb 10 14:00:07.240:rpms_proc_preauth_response:Deleting Tree node for preauth id 8 uid 19
Feb 10 14:00:07.284: Entering rpms_proc_print_preauth_req

Feb 10 14:00:07.284: Request = 0
Feb 10 14:00:07.284: Preauth id = 9
Feb 10 14:00:07.284: EndPt Type = 1
Feb 10 14:00:07.284: EndPt = 192.168.81.102
Feb 10 14:00:07.284: Resource Service = 1
```

```

Feb 10 14:00:07.284: Call_origin = answer
Feb 10 14:00:07.284: Call_type = voip
Feb 10 14:00:07.284: Calling_num = 2210001
Feb 10 14:00:07.284: Called_num = 1#1110001
Feb 10 14:00:07.284: Protocol = 0
Feb 10 14:00:07.288:rpms_proc_create_node:Created node with preauth_id = 9
Feb 10 14:00:07.288:rpms_proc_send_aaa_req:uid got is 21
Feb 10 14:00:07.300:rpms_proc_preauth_response:Context is for preauth_id 9, aaa_uid 21
Feb 10 14:00:07.300:rpms_proc_preauth_response:Deleting Tree node for preauth id 9 uid 21

```

The following example shows the output for a single leg 3 H.323 call:

```
Router# debug rpms-proc preauth h323
```

```

RPMS Process H323 preauth tracing is enabled
Feb 10 14:04:57.867: Entering rpms_proc_print_preauth_req

Feb 10 14:04:57.867: Request = 0
Feb 10 14:04:57.867: Preauth id = 10
Feb 10 14:04:57.867: EndPt Type = 1
Feb 10 14:04:57.867: EndPt = 192.168.81.102
Feb 10 14:04:57.867: Resource Service = 1
Feb 10 14:04:57.867: Call_origin = answer
Feb 10 14:04:57.867: Call_type = voip
Feb 10 14:04:57.867: Calling_num = 2210001
Feb 10 14:04:57.867: Called_num = 1#1110001
Feb 10 14:04:57.867: Protocol = 0
Feb 10 14:04:57.867:rpms_proc_create_node:Created node with preauth_id = 10
Feb 10 14:04:57.867:rpms_proc_send_aaa_req:uid got is 25
Feb 10 14:04:57.875:rpms_proc_preauth_response:Context is for preauth_id 10, aaa_uid 25
Feb 10 14:04:57.875:rpms_proc_preauth_response:Deleting Tree node for preauth id 10 uid 25

```

The following example shows output for a single leg 3 SIP call:

```
Router# debug rpms-proc preauth sip
```

```

RPMS Process SIP preauth tracing is enabled
Feb 10 14:08:02.880: Entering rpms_proc_print_preauth_req

Feb 10 14:08:02.880: Request = 0
Feb 10 14:08:02.880: Preauth id = 11
Feb 10 14:08:02.880: EndPt Type = 1
Feb 10 14:08:02.880: EndPt = 192.168.80.70
Feb 10 14:08:02.880: Resource Service = 1
Feb 10 14:08:02.880: Call_origin = answer
Feb 10 14:08:02.880: Call_type = voip
Feb 10 14:08:02.880: Calling_num = 2220001
Feb 10 14:08:02.880: Called_num = 1120001
Feb 10 14:08:02.880: Protocol = 1
Feb 10 14:08:02.880:rpms_proc_create_node:Created node with preauth_id = 11
Feb 10 14:08:02.880:rpms_proc_send_aaa_req:uid got is 28
Feb 10 14:08:02.888:rpms_proc_preauth_response:Context is for preauth_id 11, aaa_uid 28
Feb 10 14:08:02.888:rpms_proc_preauth_response:Deleting Tree node for preauth id 11 uid 28

```

Table 261 describes the significant fields shown in the display.

**Table 261** *debug rpms-proc preauth Field Descriptions*

Field	Description
Request	Request Type—0 for preauthentication, 1 for disconnect.
Preauth id	Identifier for the preauthentication request.
EndPt Type	Call Origin End Point Type—1 for IP address, 2 for Interzone ClearToken (IZCT) value.
EndPt	Call Origin End Point Value—An IP address or IZCT value.
Resource Service	Resource Service Type—1 for Reservation, 2 for Query.
Call_origin	Answer.
Call_type	Voice over IP (VoIP).
Calling_num	Calling party number (calling line identification, or CLID).
Called_num	Called party number (dialed number identification service, or DNIS).
Protocol	0 for H.323, 1 for SIP.
function reports	Various identifiers and status reports for executed functions.

# debug rtpspi all

To debug all Routing Table Protocol (RTP) security parameter index (SPI) errors, sessions, and in/out functions, use the **debug rtpspi all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtpspi all**

**no debug rtpspi all**

**Syntax Description** This command has no arguments or keywords.

**Defaults** No default behavior or values

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.0(7)XK	This command was introduced on the Cisco MC3810 and Cisco 3600 series routers (except the Cisco 3620).

## Usage Guidelines



### Caution

Be careful when you use this command because it can result in console flooding and reduced voice quality.

## Examples

The following example shows a debug trace for RTP SPI errors, sessions, and in/out functions on a gateway:

```
Router# debug rtpspi all

RTP SPI Error, Session and function in/out tracings are enabled.

*Mar 1 00:38:59.381:rtpspi_allocate_rtp_port:Entered.
*Mar 1 00:38:59.381:rtpspi_allocate_rtp_port:allocated RTP port 16544
*Mar 1 00:38:59.381:rtpspi_allocate_rtp_port:Success. port = 16544. Leaving.
*Mar 1 00:38:59.381:rtpspi_call_setup_request:entered.
    Call Id = 5, dest = 0.0.0.0;   callInfo:
    final dest flag = 0,
    rtp_session_mode = 0x2,
    local_ip_addr = 0x5000001,remote_ip_addr = 0x0,
    local rtp port = 16544, remote rtp port = 0
*Mar 1 00:38:59.381:rtpspi_call_setup_request:spi_info copied for rtpspi_app_data_t.
*Mar 1 00:38:59.385:rtpspi_call_setup_request:leaving
*Mar 1 00:38:59.385:rtpspi_call_setup() entered
*Mar 1 00:38:59.385:rtpspi_initialize_ccb:Entered
*Mar 1 00:38:59.385:rtpspi_initialize_ccb:leaving
```

```

*Mar 1 00:38:59.385:rtpspi_call_setup:rtp_session_mode = 0x2
*Mar 1 00:38:59.385:rtpspi_call_setup:mode = CC_CALL_NORMAL.
    destination number = 0.0.0.0
*Mar 1 00:38:59.385:rtpspi_call_setup:Passed local_ip_addrs=0x5000001
*Mar 1 00:38:59.385:rtpspi_call_setup:Passed local_rtp_port = 16544
*Mar 1 00:38:59.385:rtpspi_call_setup:Saved RTCP Session = 0x1AF57E0
*Mar 1 00:38:59.385:rtpspi_call_setup:Passed remote rtp port = 0.
*Mar 1 00:38:59.389:rtpspi_start_rtcp_session:entered. rtp session mode=0x2, rem rtp=0,
rem ip=0x0
*Mar 1 00:38:59.389:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar 1 00:38:59.389:rtpspi_start_rtcp_session:Starting RTCP session.
    Local IP addr = 0x5000001, Remote IP addr = 0x0,
    Local RTP port = 16544, Remote RTP port = 0, mode = 0x2
*Mar 1 00:38:59.389:rtpspi_start_rtcp_session:RTP Session creation Success.
*Mar 1 00:38:59.389:rtpspi_call_setup:RTP Session creation Success.
*Mar 1 00:38:59.389:rtpspi_call_setup:calling cc_api_call_connected()
*Mar 1 00:38:59.389:rtpspi_call_setup:Leaving.
*Mar 1 00:38:59.393:rtpspi_bridge:entered. conf id = 1, src i/f = 0x1859E88,
    dest i/f = 0x1964EEC, src call id = 5, dest call id = 4
    call info = 0x1919140, xmit fn = 0xDA7494, tag = 0
*Mar 1 00:38:59.393:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar 1 00:38:59.393:rtpspi_modify_rtcp_session_parameters():xmit fn=0xDA7494,
dstIF=0x1964EEC, dstCallID=4, voip_mode=0x2, rtp_mode=0x2, ssrc_status=0
*Mar 1 00:38:59.393:rtpspi_bridge:Calling cc_api_bridge_done() for 5(0x1AF5400) and
4(0x0).
*Mar 1 00:38:59.393:rtpspi_bridge:leaving.
*Mar 1 00:38:59.397:rtpspi_caps_ind:Entered. vdb = 0x1859E88 call id = 5, srcCallId = 4
*Mar 1 00:38:59.397:rtpspi_caps_ind:caps from VTSP:codec=0x83FB, codec_bytes=0x50,
    fax rate=0x7F, vad=0x3 modem=0x0
*Mar 1 00:38:59.397:rtpspi_get_rtcp_session_parameters():CURRENT VALUES:
dstIF=0x1964EEC, dstCallID=4, current_seq_num=0x0
*Mar 1 00:38:59.397:rtpspi_get_rtcp_session_parameters():NEW VALUES:
dstIF=0x1964EEC, dstCallID=4, current_seq_num=0x261C
*Mar 1 00:38:59.397:rtpspi_caps_ind:Caps Used:codec=0x1, codec bytes=80,
    fax rate=0x1, vad=0x1, modem=0x1, dtmf_relay=0x1, seq_num_start=0x261D
*Mar 1 00:38:59.397:rtpspi_caps_ind:calling cc_api_caps_ind().
*Mar 1 00:38:59.397:rtpspi_caps_ind:Returning success
*Mar 1 00:38:59.397:rtpspi_caps_ack:Entered. call id = 5, srcCallId = 4
*Mar 1 00:38:59.397:rtpspi_caps_ack:leaving.
*Mar 1 00:38:59.618:rtpspi_call_modify:entered. call-id=5, nominator=0x7,
params=0x18DD440
*Mar 1 00:38:59.618:rtpspi_call_modify:leaving
*Mar 1 00:38:59.618:rtpspi_do_call_modify:Entered. call-id = 5
*Mar 1 00:38:59.622:rtpspi_do_call_modify:Remote RTP port changed. New port=16432
*Mar 1 00:38:59.622:rtpspi_do_call_modify:Remote IP addrs changed. New IP addrs=0x6000001
*Mar 1 00:38:59.622:rtpspi_do_call_modify:new mode 2 is the same as the current mode
*Mar 1 00:38:59.622:rtpspi_do_call_modify:Starting new RTCP session.
*Mar 1 00:38:59.622:rtpspi_start_rtcp_session:entered. rtp session mode=0x2, rem
rtp=16432, rem ip=0x6000001
*Mar 1 00:38:59.622:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar 1 00:38:59.622:rtpspi_start_rtcp_session:Removing old RTCP session.
*Mar 1 00:38:59.622:rtpspi_start_rtcp_session:Starting RTCP session.
    Local IP addr = 0x5000001, Remote IP addr = 0x6000001,
    Local RTP port = 16544, Remote RTP port = 16432, mode = 0x2
*Mar 1 00:38:59.622:rtpspi_start_rtcp_session:RTCP Timer creation Success. (5)*(5000)
*Mar 1 00:38:59.622:rtpspi_start_rtcp_session:RTP Session creation Success.
*Mar 1 00:38:59.622:rtpspi_do_call_modify:RTP Session creation Success.
*Mar 1 00:38:59.622:rtpspi_do_call_modify:Calling cc_api_call_modify(), result=0x0
*Mar 1 00:38:59.626:rtpspi_do_call_modify:success. leaving
*Mar 1 00:39:05.019:rtpspi_call_modify:entered. call-id=5, nominator=0x7,
params=0x18DD440
*Mar 1 00:39:05.019:rtpspi_call_modify:leaving
*Mar 1 00:39:05.019:rtpspi_do_call_modify:Entered. call-id = 5
*Mar 1 00:39:05.019:rtpspi_do_call_modify:New remote RTP port = old rtp port = 16432

```

```

*Mar 1 00:39:05.019:rtpspi_do_call_modify:New remote IP addr = old IP addr = 0x6000001
*Mar 1 00:39:05.019:rtpspi_do_call_modify:Mode changed. new = 3, old = 2
*Mar 1 00:39:05.019:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x3
*Mar 1 00:39:05.023:rtpspi_modify_rtcp_session_parameters():xmit fn=0xDA7494,
dstIF=0x1964EEEC, dstCallID=4, voip_mode=0x3, rtp_mode=0x3, ssrc_status=2
*Mar 1 00:39:05.023:rtpspi_do_call_modify:RTCP Timer start.
*Mar 1 00:39:05.023:rtpspi_do_call_modify:Calling cc_api_call_modify(), result=0x0
*Mar 1 00:39:05.023:rtpspi_do_call_modify:sucess. leaving
*Mar 1 00:40:13.786:rtpspi_bridge_drop:entered. src call-id=5, dest call-id=4, tag=0
*Mar 1 00:40:13.786:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x3
*Mar 1 00:40:13.786:rtpspi_modify_rtcp_session_parameters():xmit fn=0x0,
dstIF=0x0, dstCallID=0, voip_mode=0x3, rtp_mode=0x3, ssrc_status=2
*Mar 1 00:40:13.786:rtpspi_bridge_drop:leaving
*Mar 1 00:40:13.790:rtpspi_call_disconnect:entered. call-id=5, cause=16, tag=0
*Mar 1 00:40:13.790:rtpspi_call_disconnect:leaving.
*Mar 1 00:40:13.790:rtpspi_do_call_disconnect:Entered. call-id = 5
*Mar 1 00:40:13.790:rtpspi_do_call_disconnect:calling rtpspi_call_cleanup(). call-id=5
*Mar 1 00:40:13.794:rtpspi_call_cleanup:entered. ccb = 0x1AF5400, call-id=5, rtp port =
16544
*Mar 1 00:40:13.794:rtpspi_call_cleanup:releasing ccb cache. RTP port=16544
*Mar 1 00:40:13.794:rtpspi_store_call_history_entry():Entered.
*Mar 1 00:40:13.794:rtpspi_store_call_history_entry():Leaving.
*Mar 1 00:40:13.794:rtpspi_call_cleanup:RTCP Timer Stop.
*Mar 1 00:40:13.794:rtpspi_call_cleanup:deallocating RTP port 16544.
*Mar 1 00:40:13.794:rtpspi_free_rtcp_session:Entered.
*Mar 1 00:40:13.794:rtpspi_free_rtcp_session:Success. Leaving
*Mar 1 00:40:13.794::rtpspi_call_cleanup freeing ccb (0x1AF5400)
*Mar 1 00:40:13.794:rtpspi_call_cleanup:leaving
*Mar 1 00:40:13.794:rtpspi_do_call_disconnect:leaving

```

**Related Commands**

Command	Description
<b>debug rtpspi errors</b>	Debugs RTP SPI errors.
<b>debug rtpspi inout</b>	Debugs RTP SPI in/out functions.
<b>debug rtpspi send-nse</b>	Triggers the RTP SPI to send a triple redundant NSE.
<b>debug sgcp errors</b>	Debugs SGCP errors.
<b>debug sgcp events</b>	Debugs SGCP events.
<b>debug sgcp packet</b>	Debugs SGCP packets.
<b>debug vtsp send-nse</b>	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

# debug rtpspi errors

To debug Routing Table Protocol (RTP) security parameter index (SPI) errors, use the **debug rtpspi errors** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtpspi errors**

**no debug rtpspi errors**

**Syntax Description** This command has no arguments or keywords.

**Defaults** No default behavior or values

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.0(7)XK	This command was introduced on the Cisco MC3810 device and Cisco 3600 series routers (except the Cisco 3620).

## Usage Guidelines



### Caution

Be careful when you use this command because it can result in console flooding and reduced voice quality.

## Examples

This example shows a debug trace for RTP SPI errors on two gateways. The following example shows the debug trace on the first gateway:

```
Router# debug rtpspi errors

00:54:13.272:rtpspi_do_call_modify:new mode 2 is the same as the current mode
00:54:18.738:rtpspi_do_call_modify:New remote RTP port = old rtp port = 16452
00:54:18.738:rtpspi_do_call_modify:New remote IP addr = old IP addr = 0x6000001
```

The following example shows the debug trace on the second gateway:

```
Router# debug rtpspi errors

00:54:08:rtpspi_process_timers:
00:54:08:rtpspi_process_timers:Timer 0x1A5AF9C expired.
00:54:08:rtpspi_process_timers:Timer expired for callID 0x3
00:54:08:rtpspi_process_timers:
00:54:08:rtpspi_process_timers:Timer 0x1A5AF9C expired.
00:54:08:rtpspi_process_timers:Timer expired for callID 0x3
00:54:08:rtpspi_process_timers:
00:54:08:rtpspi_process_timers:Timer 0x1A5AF9C expired.
```

```
00:54:08:rtpspi_process_timers:Timer expired for callID 0x3
00:54:09:rtpspi_process_timers:
00:54:09:rtpspi_process_timers:Timer 0x1A5AFBC expired.
00:54:09:rtpspi_process_timers:Timer expired for callID 0x3
00:54:09:rtpspi_process_timers:
00:54:09:rtpspi_process_timers:Timer 0x1A5B364 expired.
00:54:09:rtpspi_process_timers:Timer expired for callID 0x3
```

**Related Commands**

<b>Command</b>	<b>Description</b>
<b>debug rtpspi all</b>	Debugs all RTP SPI errors, sessions, and in/out functions.
<b>debug rtpspi inout</b>	Debugs RTP SPI in/out functions.
<b>debug rtpspi send-nse</b>	Triggers the RTP SPI to send a triple redundant NSE.
<b>debug sgcp errors</b>	Debugs SGCP errors.
<b>debug sgcp events</b>	Debugs SGCP events.
<b>debug sgcp packet</b>	Debugs SGCP packets.
<b>debug vtsp send-nse</b>	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

# debug rtpspi inout

To debug Routing Table Protocol (RTP) security parameter index (SPI) in/out functions, use the **debug rtpspi inout** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtpspi inout**

**no debug rtpspi inout**

**Syntax Description** This command has no arguments or keywords.

**Defaults** No default behavior or values

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.0(7)XK	This command was introduced on the Cisco MC3810 device and Cisco 3600 series routers (except the Cisco 3620 device).

## Usage Guidelines



### Caution

Be careful when you use this command because it can result in console flooding and reduced voice quality.

## Examples

The following example shows a debug trace for RTP SPI in/out functions on a gateway:

```
Router# debug rtpspi inout

*Mar 1 00:57:24.565:rtpspi_allocate_rtp_port:Entered.
*Mar 1 00:57:24.565:rtpspi_allocate_rtp_port:Success. port = 16520. Leaving.
*Mar 1 00:57:24.565:rtpspi_call_setup_request:entered.
    Call Id = 9, dest = 0.0.0.0;   callInfo:
    final dest flag = 0,
    rtp_session_mode = 0x2,
    local_ip_addr = 0x5000001,remote_ip_addr = 0x0,
    local rtp port = 16520, remote rtp port = 0
*Mar 1 00:57:24.565:rtpspi_call_setup_request:spi_info copied for rtpspi_app_data_t.
*Mar 1 00:57:24.565:rtpspi_call_setup_request:leaving
*Mar 1 00:57:24.569:rtpspi_call_setup() entered
*Mar 1 00:57:24.569:rtpspi_initialize_ccb:Entered
*Mar 1 00:57:24.569:rtpspi_initialize_ccb:leaving
*Mar 1 00:57:24.569:rtpspi_start_rtcp_session:entered. rtp session mode=0x2, rem rtp=0,
rem ip=0x0
*Mar 1 00:57:24.569:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar 1 00:57:24.569:rtpspi_call_setup:Leaving.
```

```
*Mar 1 00:57:24.573:rtpspi_bridge:entered. conf id = 3, src i/f = 0x1859E88,
  dest i/f = 0x1964EEC, src call id = 9, dest call id = 8
  call info = 0x1919140, xmit fn = 0xDA7494, tag = 0
*Mar 1 00:57:24.573:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar 1 00:57:24.573:rtpspi_bridge:leaving.
*Mar 1 00:57:24.573:rtpspi_caps_ind:Entered. vdb = 0x1859E88 call id = 9, srcCallId = 8
*Mar 1 00:57:24.577:rtpspi_caps_ind:Returning success
*Mar 1 00:57:24.577:rtpspi_caps_ack:Entered. call id = 9, srcCallId = 8
*Mar 1 00:57:24.577:rtpspi_caps_ack:leaving.
*Mar 1 00:57:24.818:rtpspi_call_modify:entered. call-id=9, nominator=0x7,
params=0x18DD440
*Mar 1 00:57:24.818:rtpspi_call_modify:leaving
*Mar 1 00:57:24.818:rtpspi_do_call_modify:Entered. call-id = 9
*Mar 1 00:57:24.818:rtpspi_start_rtcp_session:entered. rtp session mode=0x2, rem
rtp=16396, rem ip=0x6000001
*Mar 1 00:57:24.822:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x2
*Mar 1 00:57:24.822:rtpspi_do_call_modify:success. leaving
*Mar 1 00:57:30.296:rtpspi_call_modify:entered. call-id=9, nominator=0x7,
params=0x18DD440
*Mar 1 00:57:30.296:rtpspi_call_modify:leaving
*Mar 1 00:57:30.300:rtpspi_do_call_modify:Entered. call-id = 9
*Mar 1 00:57:30.300:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x3
*Mar 1 00:57:30.300:rtpspi_do_call_modify:success. leaving
*Mar 1 00:58:39.055:rtpspi_bridge_drop:entered. src call-id=9, dest call-id=8, tag=0
*Mar 1 00:58:39.055:rtpspi_get_rtcp_mode:entered. rtp_mode = 0x3
*Mar 1 00:58:39.055:rtpspi_bridge_drop:leaving
*Mar 1 00:58:39.059:rtpspi_call_disconnect:entered. call-id=9, cause=16, tag=0
*Mar 1 00:58:39.059:rtpspi_call_disconnect:leaving.
*Mar 1 00:58:39.059:rtpspi_do_call_disconnect:Entered. call-id = 9
*Mar 1 00:58:39.059:rtpspi_call_cleanup:entered. ccb = 0x1AF5400, call-id=9, rtp port =
16520
*Mar 1 00:58:39.059:rtpspi_store_call_history_entry():Entered.
*Mar 1 00:58:39.059:rtpspi_store_call_history_entry():Leaving.
*Mar 1 00:58:39.059:rtpspi_free_rtcp_session:Entered.
*Mar 1 00:58:39.059:rtpspi_free_rtcp_session:Success. Leaving
*Mar 1 00:58:39.063:rtpspi_call_cleanup:leaving
*Mar 1 00:58:39.063:rtpspi_do_call_disconnect:leaving
```

**Related Commands**

Command	Description
<b>debug rtpspi all</b>	Debugs all RTP SPI errors, sessions, and in/out functions.
<b>debug rtpspi errors</b>	Debugs RTP SPI errors.
<b>debug rtpspi send-nse</b>	Triggers the RTP SPI to send a triple redundant NSE.
<b>debug sgcp errors</b>	Debugs SGCP errors.
<b>debug sgcp events</b>	Debugs SGCP events.
<b>debug sgcp packet</b>	Debugs SGCP packets.
<b>debug vtsp send-nse</b>	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

# debug rtpspi send-nse

To trigger the Routing Table Protocol (RTP) security parameter index (SPI) software module to send a triple redundant NSE, use the **debug rtpspi send-nse** command in privileged EXEC mode. To disable this action, use the **no** form of the command.

**debug rtpspi send-nse** *call-ID NSE-event-ID*

**no debug rtpspi send-nse** *call-ID NSE-event-ID*

Syntax Description		
	<i>call-ID</i>	Specifies the call ID of the active call. The valid range is from 0 to 65535.
	<i>NSE-event-ID</i>	Specifies the NSE Event ID. The valid range is from 0 to 255.

**Defaults** No default behavior or values

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.0(7)XK	This command was introduced on the Cisco MC3810 device and Cisco 3600 series routers (except the Cisco 3620 router).

**Examples** The following example shows the RTP SPI software module set to send an NSE:

```
Router# debug rtpspi send-nse
```

Related Commands	Command	Description
	<b>debug rtpspi all</b>	Debugs all RTP SPI errors, sessions, and in/out functions.
	<b>debug rtpspi errors</b>	Debugs RTP SPI errors.
	<b>debug rtpspi inout</b>	Debugs RTP SPI in/out functions.
	<b>debug sgcp errors</b>	Debugs SGCP errors.
	<b>debug sgcp events</b>	Debugs SGCP events.
	<b>debug sgcp packet</b>	Debugs SGCP packets.
	<b>debug vtsp send-nse</b>	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

# debug rtpspi session

To debug all Routing Table Protocol (RTP) security parameter index (SPI) sessions, use the **debug rtpspi session** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

**debug rtpspi session**

**no debug rtpspi session**

**Syntax Description** This command has no arguments or keywords.

**Defaults** No default behavior or values

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.0(7)XK	This command was introduced on the Cisco MC3810 device and Cisco 3600 series routers (except the Cisco 3620 router).

**Examples** The following example shows a debug trace for RTP SPI sessions on a gateway:

```
Router# debug rtpspi session

*Mar  1 01:01:51.593:rtpspi_allocate_rtp_port:allocated RTP port 16406
*Mar  1 01:01:51.593:rtpspi_call_setup:rtp_session_mode = 0x2
*Mar  1 01:01:51.593:rtpspi_call_setup:mode = CC_CALL_NORMAL.
      destination number = 0.0.0.0
*Mar  1 01:01:51.593:rtpspi_call_setup:Passed local_ip_addrs=0x5000001
*Mar  1 01:01:51.593:rtpspi_call_setup:Passed local_rtp_port = 16406
*Mar  1 01:01:51.593:rtpspi_call_setup:Saved RTCP Session = 0x1AFDFBC
*Mar  1 01:01:51.593:rtpspi_call_setup:Passed remote rtp port = 0.
*Mar  1 01:01:51.598:rtpspi_start_rtcp_session:Starting RTCP session.
      Local IP addr = 0x5000001, Remote IP addr = 0x0,
      Local RTP port = 16406, Remote RTP port = 0, mode = 0x2
*Mar  1 01:01:51.598:rtpspi_start_rtcp_session:RTP Session creation Success.
*Mar  1 01:01:51.598:rtpspi_call_setup:RTP Session creation Success.
*Mar  1 01:01:51.598:rtpspi_call_setup:calling cc_api_call_connected()
*Mar  1 01:01:51.598:rtpspi_modify_rtcp_session_parameters():xmit fn=0xDA7494,
dstIF=0x1964EEC, dstCallID=10, voip_mode=0x2, rtp_mode=0x2, ssrc_status=0
*Mar  1 01:01:51.598:rtpspi_bridge:Calling cc_api_bridge_done() for 11(0x1AF5400) and
10(0x0).
*Mar  1 01:01:51.602:rtpspi_caps_ind:caps from VTSP:codec=0x83FB, codec_bytes=0x50,
      fax rate=0x7F, vad=0x3 modem=0x0
*Mar  1 01:01:51.602:rtpspi_get_rtcp_session_parameters():CURRENT VALUES:
dstIF=0x1964EEC, dstCallID=10, current_seq_num=0x0
*Mar  1 01:01:51.602:rtpspi_get_rtcp_session_parameters():NEW VALUES:
dstIF=0x1964EEC, dstCallID=10, current_seq_num=0xF1E
*Mar  1 01:01:51.602:rtpspi_caps_ind:Caps Used:codec=0x1, codec bytes=80,
      fax rate=0x1, vad=0x1, modem=0x1, dtmf_relay=0x1, seq_num_start=0xF1F
*Mar  1 01:01:51.602:rtpspi_caps_ind:calling cc_api_caps_ind().
```

```

*Mar 1 01:01:51.822:rtpspi_do_call_modify:Remote RTP port changed. New port=16498
*Mar 1 01:01:51.822:rtpspi_do_call_modify:Remote IP addr changed. New IP addr=0x6000001
*Mar 1 01:01:51.822:rtpspi_do_call_modify:Starting new RTCP session.
*Mar 1 01:01:51.822:rtpspi_start_rtcp_session:Removing old RTCP session.
*Mar 1 01:01:51.822:rtpspi_start_rtcp_session:Starting RTCP session.
    Local IP addr = 0x5000001, Remote IP addr = 0x6000001,
    Local RTP port = 16406, Remote RTP port = 16498, mode = 0x2
*Mar 1 01:01:51.822:rtpspi_start_rtcp_session:RTCP Timer creation Success. (5)*(5000)
*Mar 1 01:01:51.826:rtpspi_start_rtcp_session:RTP Session creation Success.
*Mar 1 01:01:51.826:rtpspi_do_call_modify:RTP Session creation Success.
*Mar 1 01:01:51.826:rtpspi_do_call_modify:Calling cc_api_call_modify(), result=0x0
*Mar 1 01:01:57.296:rtpspi_do_call_modify:Mode changed. new = 3, old = 2
*Mar 1 01:01:57.296:rtpspi_modify_rtcp_session_parameters():xmit fn=0xDA7494,
dstIF=0x1964EEC, dstCallID=10, voip_mode=0x3, rtp_mode=0x3, ssrc_status=2
*Mar 1 01:01:57.296:rtpspi_do_call_modify:RTCP Timer start.
*Mar 1 01:01:57.296:rtpspi_do_call_modify:Calling cc_api_call_modify(), result=0x0
*Mar 1 01:03:06.108:rtpspi_modify_rtcp_session_parameters():xmit fn=0x0,
dstIF=0x0, dstCallID=0, voip_mode=0x3, rtp_mode=0x3, ssrc_status=2
*Mar 1 01:03:06.112:rtpspi_do_call_disconnect:calling rtpspi_call_cleanup(). call-id=11
*Mar 1 01:03:06.112:rtpspi_call_cleanup:releasing ccb cache. RTP port=16406
*Mar 1 01:03:06.112:rtpspi_call_cleanup:RTCP Timer Stop.
*Mar 1 01:03:06.112:rtpspi_call_cleanup:deallocating RTP port 16406.
*Mar 1 01:03:06.112::rtpspi_call_cleanup freeing ccb (0x1AF5400)

```

**Related Commands**

Command	Description
<b>debug rtpspi all</b>	Debugs all RTP SPI errors, sessions, and in/out functions.
<b>debug rtpspi errors</b>	Debugs RTP SPI errors.
<b>debug rtpspi inout</b>	Debugs RTP SPI in/out functions.
<b>debug rtpspi send-nse</b>	Triggers the RTP SPI to send a triple redundant NSE.
<b>debug sgcp errors</b>	Debugs SGCP errors.
<b>debug sgcp events</b>	Debugs SGCP events.
<b>debug sgcp packet</b>	Debugs SGCP packets.
<b>sgcp</b>	Starts and allocates resources for the SCGP daemon.
<b>debug vtsp send-nse</b>	Sends and debugs a triple redundant NSE from the DSP to a remote gateway.

# debug rtr error

To enable logging of Service Assurance (SA) Agent run-time errors, use the **debug rtr error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtr error** [*probe*]

**no debug rtr error** [*probe*]

## Syntax Description

*probe* (Optional) Number of the probe in the range from 0 to 31.

## Defaults

Logging is off.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
11.2	This command was introduced.
12.0(5)T	This command was modified.

## Usage Guidelines

The **debug rtr error** command displays run-time errors. When a probe number other than 0 is specified, all run-time errors for that probe are displayed when the probe is active. When the probe number is 0 all run-time errors relating to the Response Time Reporter scheduler process are displayed. When no probe number is specified, all run-time errors for all active probes configured on the router and probe control are displayed.



### Note

Use the **debug rtr error** command before using the **debug rtr trace** command because the **debug rtr error** command generates a lesser amount of debugging output.

## Examples

The following example shows output from the **debug rtr error** command. The output indicates failure because the target is not there or because the responder is not enabled on the target. All debugging output for the Response Time Reporter (including the **debug rtr trace** command) has the format shown in [Table 262](#).

```
Router# debug rtr error

May  5 05:00:35.483: control message failure:1
May  5 05:01:35.003: control message failure:1
May  5 05:02:34.527: control message failure:1
May  5 05:03:34.039: control message failure:1
May  5 05:04:33.563: control message failure:1
May  5 05:05:33.099: control message failure:1
May  5 05:06:32.596: control message failure:1
May  5 05:07:32.119: control message failure:1
May  5 05:08:31.643: control message failure:1
```

```
May 5 05:09:31.167: control message failure:1
May 5 05:10:30.683: control message failure:1
```

Table 262 describes the significant fields shown in the display.

**Table 262** *debug rtr error Field Descriptions*

Field	Description
RTR 1	Number of the probe generating the message.
Error Return Code	Message identifier indicating the error type (or error itself).
LU0 RTR Probe 1	Name of the process generating the message.
in echoTarget on call luReceive LuApiReturnCode of InvalidHandle - invalid host name or API handle	Supplemental messages that pertain to the message identifier.

#### Related Commands

Command	Description
<b>debug rtr trace</b>	Traces the execution of an SA Agent operation.

# debug rtr trace

To trace the execution of a Service Assurance (SA) Agent operation, use the **debug rtr trace** command in privileged EXEC mode. To disable trace debugging output (but not **debug rtr error** output), use the **no** form of this command.

**debug rtr trace** [*probe*]

**no debug rtr trace** [*probe*]

## Syntax Description:

*probe* (Optional) Number of the probe in the range from 0 to 31.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
11.2	This command was introduced.
12.0(5)T	This command was modified.

## Usage Guidelines

When a probe number other than 0 is specified, execution for that probe is traced. When the probe number is 0, the Response Time Reporter scheduler process is traced. When no probe number is specified, all active probes and every probe control is traced.

The **debug rtr trace** command also enables **debug rtr error** command for the specified probe. However, the **no debug rtr trace** command does not disable the **debug rtr error** command. You must manually disable the command by using the **no debug rtr error** command.

All debug output (including **debug rtr error** command output) has the format shown in the **debug rtr error** command output example.



### Note

The **debug rtr trace** command can generate a large number of debug messages. First use the **debug rtr error** command, and then use the **debug rtr trace** on a per-probe basis.

## Examples

The following output is from the **debug rtr trace** command. In this example, a probe is traced through a single operation attempt: the setup of a connection to the target, and the attempt at an echo to calculate UDP packet response time.

```
Router# debug rtr trace

Router# RTR 1:Starting An Echo Operation - IP RTR Probe 1

May  5 05:25:08.584:rtr hash insert :3.0.0.3 3383
May  5 05:25:08.584:source=3.0.0.3(3383)  dest-ip=5.0.0.1(9)
May  5 05:25:08.588:sending control msg:
May  5 05:25:08.588: Ver:1 ID:51 Len:52
May  5 05:25:08.592:cmd:command:RTT_CMD_UDP_PORT_ENABLE, ip:5.0.0.1, port:9, duration:5000
May  5 05:25:08.607:receiving reply
May  5 05:25:08.607: Ver:1 ID:51 Len:8
```

```

May 5 05:25:08.623:local delta:8
May 5 05:25:08.627:delta from responder:1
May 5 05:25:08.627:received <16> bytes and responseTime = 3 (ms)
May 5 05:25:08.631:rtr hash remove:3.0.0.3 3383RTR 1:Starting An Echo Operation - IP RTR
Probe 1

May 5 05:26:08.104:rtr hash insert :3.0.0.3 2974
May 5 05:26:08.104:source=3.0.0.3(2974) dest-ip=5.0.0.1(9)
May 5 05:26:08.108:sending control msg:
May 5 05:26:08.108: Ver:1 ID:52 Len:52
May 5 05:26:08.112:cmd:command:RTT_CMD_UDP_PORT_ENABLE, ip:5.0.0.1, port:9, duration:5000
May 5 05:26:08.127:receiving reply
May 5 05:26:08.127: Ver:1 ID:52 Len:8
May 5 05:26:08.143:local delta:8
May 5 05:26:08.147:delta from responder:1
May 5 05:26:08.147:received <16> bytes and responseTime = 3 (ms)
May 5 05:26:08.151:rtr hash remove:3.0.0.3 2974RTR 1:Starting An Echo Operation - IP RTR
Probe 1

```

**Related Commands**

<b>Command</b>	<b>Description</b>
<b>debug rtpspi all</b>	Enables logging of SA Agent run-time errors.

# debug rtsp all

To display all related information about the Real Time Streaming Protocol (RTSP) data, use the **debug rtsp all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtsp all**

**no debug rtsp all**

**Syntax Description** This command has no arguments or keywords.

**Defaults** Debug is not enabled.

**Command Modes** Privileged EXEC

## Command History

Release	Modification
12.1(3)T	This command was introduced.
12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.

## Usage Guidelines

We recommend that you log output from the **debug rtsp all** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

## Examples

The following example shows debugging output for the **debug rtsp all** command. The **show debug** command shows which RTSP modules are traced.

```
Router# debug rtsp all

All RTSP client debugging is on

Router# show debug

RTSP:
  RTSP client Protocol Error debugging is on
  RTSP client Protocol Message Handler debugging is on
  RTSP client API debugging is on
  RTSP client socket debugging is on
  RTSP client session debugging is on
Router#
Router#!call initiated
Router#
*Mar 11 03:14:23.471: //-1//RTSP:/rtsp_get_new_scb:
```

```

*Mar 11 03:14:23.471: //-1//RTSP:/rtsp_initialize_scb:
*Mar 11 03:14:23.471: //-1//RTSP:/rtsplib_init_svr_session: 0x63A5FE6C
*Mar 11 03:14:23.471: //-1//RTSP:/rtsp_api_create_session: evh=0x6155F0D4
context=0x6345042C
*Mar 11 03:14:23.471: //-1//RTSP:/rtsp_get_new_scb:
*Mar 11 03:14:23.471: //-1//RTSP:/rtsp_initialize_scb:
*Mar 11 03:14:23.471: //-1//RTSP:/rtsplib_init_svr_session: 0x63A5D874
*Mar 11 03:14:23.471: //-1//RTSP:/rtsp_api_create_session: evh=0x6155F204
context=0x6345046C
*Mar 11 03:14:23.471: //-1//RTSP:RS45:/rtsp_api_request: msg=0x63A59FB8
*Mar 11 03:14:23.471: //-1//RTSP:RS45:/rtsp_api_handle_req_set_params: msg=0x63A59FB8
*Mar 11 03:14:23.475: //-1//RTSP:RS45:/rtsp_api_free_msg_buffer: msg=0x63A59FB8
*Mar 11 03:14:23.475: //-1//RTSP:RS45:/rtsp_api_request: msg=0x63A5A304
*Mar 11 03:14:23.475: //-1//RTSP:RS45:/rtsp_api_handle_req_set_params: msg=0x63A5A304
*Mar 11 03:14:23.475: //-1//RTSP:RS45:/rtsp_api_free_msg_buffer: msg=0x63A5A304
*Mar 11 03:14:23.475: //-1//RTSP:RS45:/rtsp_api_request: msg=0x63A5A650
*Mar 11 03:14:23.475: //-1//RTSP:RS45:/rtsp_api_handle_req_set_params: msg=0x63A5A650
*Mar 11 03:14:23.475: //166//RTSP:LP:RS45:/rtsp_api_handle_req_set_params:
*Mar 11 03:14:23.475: //-1//RTSP:RS45:/rtsp_api_free_msg_buffer: msg=0x63A5A650
*Mar 11 03:14:23.475: //-1//RTSP:RS46:/rtsp_api_request: msg=0x63A5A99C
*Mar 11 03:14:23.475: //-1//RTSP:RS46:/rtsp_api_handle_req_set_params: msg=0x63A5A99C
*Mar 11 03:14:23.475: //166//RTSP:LP:RS46:/rtsp_api_handle_req_set_params:
*Mar 11 03:14:23.475: //-1//RTSP:RS46:/rtsp_api_free_msg_buffer: msg=0x63A5A99C
Router#
Router#!call answered
Router#
Router#!digits dialed
Router#
Router#!call terminated
Router#
*Mar 11 03:14:51.603: //-1//RTSP:RS45:/rtsp_api_request: msg=0x63A5ACE8
*Mar 11 03:14:51.603: //-1//RTSP:RS46:/rtsp_api_request: msg=0x63A5B034
*Mar 11 03:14:51.607: //-1//RTSP:RS45:/rtsp_control_process_msg:
*Mar 11 03:14:51.607: //166//RTSP:/rtsp_control_process_msg: received MSG request of TYPE
0
*Mar 11 03:14:51.607: //166//RTSP:/rtsp_set_event: api_req_msg_type=RTSP_API_REQ_DESTROY
*Mar 11 03:14:51.607: //166//RTSP:/rtsp_session_cleanup:
*Mar 11 03:14:51.607: //-1//RTSP:/rtsplib_free_svr_session:
*Mar 11 03:14:51.607: //-1//RTSP:/rtsplib_stop_timer: timer(0x638D5DDC) stops
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_create_session_history: scb=0x63A5FE6C,
callID=0xA6
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_create_session_history: No streams in session
control block
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_session_cleanup: deleting session: scb=0x63A5FE6C
*Mar 11 03:14:51.611: //-1//RTSP:RS45:/rtsp_api_free_msg_buffer: msg=0x63A5ACE8
*Mar 11 03:14:51.611: //-1//RTSP:RS46:/rtsp_control_process_msg:
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_control_process_msg: received MSG request of TYPE
0
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_set_event: api_req_msg_type=RTSP_API_REQ_DESTROY
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_session_cleanup:
*Mar 11 03:14:51.611: //-1//RTSP:/rtsplib_free_svr_session:
*Mar 11 03:14:51.611: //-1//RTSP:/rtsplib_stop_timer: timer(0x63A60110) stops
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_create_session_history: scb=0x63A5D874,
callID=0xA6
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_create_session_history: No streams in session
control block
*Mar 11 03:14:51.611: //166//RTSP:/rtsp_session_cleanup: deleting session: scb=0x63A5D874
*Mar 11 03:14:51.611: //-1//RTSP:RS46:/rtsp_api_free_msg_buffer: msg=0x63A5B034

```

Table 263 describes the significant fields shown in the display.

**Table 263** *debug rtsp all* Field Descriptions

Field	Description
//-1/	Indicates that the CallEntry ID for the module is unavailable.
//166/	Identifies the CallEntry ID.
RTSP:	Identifies the RTSP module.
rtsp_ <i>function name</i>	Identifies the function name.

**Related Commands**

Command	Description
<b>debug rtsp api</b>	Displays debugging output for the RTSP client API.
<b>debug rtsp error</b>	Displays error message for RTSP data.
<b>debug rtsp pmh</b>	Displays debugging messages for the PMH.
<b>debug rtsp socket</b>	Displays debugging output for the RTSP client socket data.
<b>voice call debug</b>	Allows configuration of the voice call debugging output.

# debug rtsp api

To display information about the Real Time Streaming Protocol (RTSP) application programming interface (API) messages passed down to the RTSP client, use the **debug rtsp api** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtsp api**

**no debug rtsp api**

**Syntax Description** This command has no arguments or keywords.

**Defaults** Debug is not enabled.

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.1(3)T	This command was introduced.
	12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.

**Usage Guidelines** We recommend that you log output from the **debug rtsp api** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

**Examples** The following example shows output from the **debug rtsp api** command:

```
Router# debug rtsp api

RTSP client API debugging is on

Router# !call initiated

*Mar 11 03:04:41.699: //-1//RTSP:/rtsp_api_create_session: evh=0x6155F0D4
context=0x6345088C
*Mar 11 03:04:41.699: //-1//RTSP:/rtsp_api_create_session: evh=0x6155F204
context=0x634508CC
*Mar 11 03:04:41.699: //-1//RTSP:RS35:/rtsp_api_request: msg=0x63A59FB8
*Mar 11 03:04:41.699: //-1//RTSP:RS35:/rtsp_api_handle_req_set_params: msg=0x63A59FB8
*Mar 11 03:04:41.699: //-1//RTSP:RS35:/rtsp_api_free_msg_buffer: msg=0x63A59FB8
*Mar 11 03:04:41.699: //-1//RTSP:RS35:/rtsp_api_request: msg=0x63A5A304
*Mar 11 03:04:41.699: //-1//RTSP:RS35:/rtsp_api_handle_req_set_params: msg=0x63A5A304
*Mar 11 03:04:41.699: //-1//RTSP:RS35:/rtsp_api_free_msg_buffer: msg=0x63A5A304
*Mar 11 03:04:41.703: //-1//RTSP:RS35:/rtsp_api_request: msg=0x63A5A650
*Mar 11 03:04:41.703: //-1//RTSP:RS35:/rtsp_api_handle_req_set_params: msg=0x63A5A650
```

```
*Mar 11 03:04:41.703: //146//RTSP:LP:RS35:/rtsp_api_handle_req_set_params:
*Mar 11 03:04:41.703: //-1//RTSP:RS35:/rtsp_api_free_msg_buffer: msg=0x63A5A650
*Mar 11 03:04:41.703: //-1//RTSP:RS36:/rtsp_api_request: msg=0x63A5A99C
*Mar 11 03:04:41.703: //-1//RTSP:RS36:/rtsp_api_handle_req_set_params: msg=0x63A5A99C
*Mar 11 03:04:41.703: //146//RTSP:LP:RS36:/rtsp_api_handle_req_set_params:
*Mar 11 03:04:41.703: //-1//RTSP:RS36:/rtsp_api_free_msg_buffer: msg=0x63A5A99C

Router!call answered

Router#!digits dialed

Router#!call terminated

*Mar 11 03:05:15.367: //-1//RTSP:RS35:/rtsp_api_request: msg=0x63A5ACE8
*Mar 11 03:05:15.367: //-1//RTSP:RS36:/rtsp_api_request: msg=0x63A5B034
*Mar 11 03:05:15.367: //-1//RTSP:RS35:/rtsp_api_free_msg_buffer: msg=0x63A5ACE8
*Mar 11 03:05:15.367: //-1//RTSP:RS36:/rtsp_api_free_msg_buffer: msg=0x63A5B034
```

Table 264 describes the significant fields shown in the display.

**Table 264 debug rtsp api Field Descriptions**

Field	Description
//-1/	Indicates that the CallEntry ID for the module is unavailable.
//146/	Identifies the CallEntry ID.
RTSP:	Identifies the RTSP module.
rtsp_ <i>function name</i>	Identifies the function name.

**Related Commands**

Command	Description
<b>debug rtsp error</b>	Displays error message for RTSP data.
<b>debug rtsp pmh</b>	Displays debugging messages for the PMH.
<b>debug rtsp socket</b>	Displays debugging output for the RTSP client socket data.
<b>voice call debug</b>	Allows configuration of the voice call debugging output.

# debug rtsp client



## Note

Effective with Release 12.3(4), the **debug rtsp cleint** command is replaced by the **debug rtsp session** command. See the **debug rtsp session** command for more information.

To display client information and stream information for the stream that is currently active for the Real Time Streaming Protocol (RTSP) client, use the **debug rtsp client** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtsp client**

**no debug rtsp client**

## Syntax Description

This command has no arguments or keywords.

## Defaults

Debug is not enabled.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.1(3)T	This command was introduced.
12.3(4)T	This command was replaced by the <b>debug rtsp session</b> command.

## Usage Guidelines

We recommend that you log output from the **debug rtsp client** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

## Related Commands

Command	Description
<b>debug rtsp api</b>	Displays debugging output for the RTSP client API.
<b>debug rtsp error</b>	Displays error message for RTSP data.
<b>debug rtsp pmh</b>	Displays debugging messages for the PMH.
<b>debug rtsp socket</b>	Displays debugging output for the RTSP client socket data.
<b>voice call debug</b>	Allows configuration of the voice call debugging output.

# debug rtsp client session



**Note**

Effective with Release 12.3(4), the **debug rtsp cleint session** command is replaced by the **debug rtsp session** command. See the **debug rtsp session** command for more information.

To display debug messages about the Real Time Streaming Protocol (RTSP) client or the current session, use the **debug rtsp** command. To disable debugging output, use the **no** form of this command.

**debug rtsp [client | session]**

**no debug rtsp [client | session]**

**Syntax Description**

<b>client</b>	(Optional) Displays client information and stream information for the stream that is currently active.
<b>session</b>	(Optional) Displays cumulative information about the session, packet statistics, and general call information such as call ID, session ID, individual RTSP stream URLs, packet statistics, and play duration.

**Defaults**

Debug is not enabled.

**Command History**

Release	Modification
12.1(3)T	This command was introduced.
12.3(4)T	This command was replaced by the <b>debug rtsp session</b> command.

**Examples**

The following example displays the debug messages of the RTSP session:

```
Router# debug rtsp session

RTSP client session debugging is on
router#
Jan  1 00:08:36.099:rtsp_get_new_scb:
Jan  1 00:08:36.099:rtsp_initialize_scb:
Jan  1 00:08:36.099:rtsp_control_process_msg:
Jan  1 00:08:36.099:rtsp_control_process_msg:received MSG request of TYPE 0
Jan  1 00:08:36.099:rtsp_set_event:
Jan  1 00:08:36.099:rtsp_set_event:api_req_msg_type=RTSP_API_REQ_PLAY
Jan  1 00:08:36.103:rtsp_set_event:url:[rtsp://rtsp-cisco.cisco.com:554/en_welcome.au]
Jan  1 00:08:36.103:rtsp_process_async_event:SCB=0x62128F08
Jan  1 00:08:36.103:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_IDLE
                    rtsp_event = RTSP_EV_PLAY_OR_REC
Jan  1 00:08:36.103:act_idle_event_play_or_rec_req:
Jan  1 00:08:36.103:rtsp_resolve_dns:
Jan  1 00:08:36.103:rtsp_resolve_dns:IP Addr = 1.13.79.6:
Jan  1 00:08:36.103:rtsp_connect_to_svr:
Jan  1 00:08:36.103:rtsp_connect_to_svr:socket=0, connection_state = 2
Jan  1 00:08:36.103:rtsp_start_timer:timer (0x62128FD0)starts - delay (10000)
Jan  1 00:08:36.107:rtsp_control_main:SOCK= 0 Event=0x1
```

```

Jan 1 00:08:36.107:rtsp_stop_timer:timer(0x62128FD0) stops
Jan 1 00:08:36.107:rtsp_process_async_event:SCB=0x62128F08
Jan 1 00:08:36.107:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_IDLE
      rtsp_event = RTSP_EV_SVR_CONNECTED
Jan 1 00:08:36.107:act_idle_event_svr_connected:
Jan 1 00:08:36.107:rtsp_control_main:SOCK= 0 Event=0x1
Jan 1 00:08:36.783:rtsp_control_main:SOCK= 0 Event=0x1
Jan 1 00:08:36.783:rtsp_process_async_event:SCB=0x62128F08
Jan 1 00:08:36.783:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_READY
      rtsp_event = RTSP_EV_SVR_DESC_OR_ANNOUNCE_RESP
Jan 1 00:08:36.783:act_ready_event_desc_or_announce_resp:
Jan 1
00:08:36.783:act_ready_event_desc_or_announce_resp:RTSP_STATUS_DESC_OR_ANNOUNCE_RESP_OK
Jan 1 00:08:37.287:rtsp_control_main:SOCK= 0 Event=0x1
Jan 1 00:08:37.287:rtsp_process_async_event:SCB=0x62128F08
Jan 1 00:08:37.287:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_READY
      rtsp_event = RTSP_EV_SVR_SETUP_RESP
Jan 1 00:08:37.287:act_ready_event_setup_resp:
Jan 1 00:08:37.287:act_ready_event_setup_resp:Remote RTP Port=13344
Jan 1 00:08:37.287:rtsp_rtp_stream_setup:scb=0x62128F08, callID=0x7 record=0
Jan 1 00:08:37.287:rtsp_rtp_stream_setup:Starting RTCP session.
      Local IP addr = 1.13.79.45, Remote IP addr = 1.13.79.6,
      Local RTP port = 18748, Remote RTP port = 13344 CallID=8
Jan 1 00:08:37.291:xmit_func = 0x0 vdbptr = 0x61A0FC98
Jan 1 00:08:37.291:rtsp_control_main:CCAPI Queue Event
Jan 1 00:08:37.291:rtsp_rtp_associate_done:ev=0x62070E08, callID=0x7
Jan 1 00:08:37.291:rtsp_rtp_associate_done:scb=0x62128F08
Jan 1 00:08:37.291:rtsp_rtp_associate_done:callID=0x7, pVdb=0x61F4FBC8,
Jan 1 00:08:37.291:      spi_context=0x6214145C
Jan 1 00:08:37.291:      disposition=0, playFunc=0x60CA2238,
Jan 1 00:08:37.291:      codec=0x5, vad=0, mediaType=6,
Jan 1 00:08:37.291:      stream_assoc_id=1
Jan 1 00:08:37.291:rtsp_rtp_modify_session:scb=0x62128F08, callID=0x7
Jan 1 00:08:37.291:rtsp_process_async_event:SCB=0x62128F08
Jan 1 00:08:37.291:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_READY
      rtsp_event = RTSP_EV_ASSOCIATE_DONE
Jan 1 00:08:37.291:act_ready_event_associate_done:
Jan 1 00:08:37.291:rtsp_get_stream:
Jan 1 00:08:37.783:rtsp_control_main:SOCK= 0 Event=0x1
Jan 1 00:08:37.783:rtsp_process_async_event:SCB=0x62128F08
Jan 1 00:08:37.783:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_READY
      rtsp_event = RTSP_EV_SVR_PLAY_OR_REC_RESP
Jan 1 00:08:37.783:act_ready_event_play_or_rec_resp:
Jan 1 00:08:37.783:rtsp_start_timer:timer(0x62128FB0)starts - delay (4249)
rtsp-5#
Jan 1 00:08:42.035:rtsp_process_timer_events:
Jan 1 00:08:42.035:rtsp_process_timer_events:PLAY OR RECORD completed
Jan 1 00:08:42.035:rtsp_process_async_event:SCB=0x62128F08
Jan 1 00:08:42.035:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_PLAY_OR_REC
      rtsp_event = RTSP_EV_PLAY_OR_REC_TIMER_EXPIRED
Jan 1 00:08:42.035:act_play_event_play_done:
Jan 1 00:08:42.035:act_play_event_play_done:elapsed play time = 4249 total play time =
4249
Jan 1 00:08:42.035:rtsp_send_teardown_to_svr:
Jan 1 00:08:42.487:rtsp_control_main:SOCK= 0 Event=0x1
Jan 1 00:08:42.487:rtsp_process_async_event:SCB=0x62128F08
Jan 1 00:08:42.487:rtsp_process_async_event:rtsp_state = RTSP_SES_STATE_PLAY_OR_REC
      rtsp_event = RTSP_EV_SVR_TEARDOWN_RESP
Jan 1 00:08:42.487:act_play_event_teardown_resp:
Jan 1 00:08:42.487:rtsp_server_closed:
Jan 1 00:08:42.487:rtsp_send_resp_to_api:
Jan 1 00:08:42.487:rtsp_send_resp_to_api:sending RESP=RTSP_STATUS_PLAY_COMPLETE
Jan 1 00:08:42.491:rtsp_rtp_teardown_stream:scb=0x62128F08, callID=0x7
Jan 1 00:08:42.491:rtsp_rtp_stream_cleanup:scb=0x62128F08, callID=0x7

```

```

Jan 1 00:08:42.491:rtsp_update_stream_stats:scb=0x62128F08, stream=0x61A43350,
Jan 1 00:08:42.491:call_info=0x6214C67C, callID=0x7
Jan 1 00:08:42.491:rtsp_update_stream_stats:rx_bytes = 25992
Jan 1 00:08:42.491:rtsp_update_stream_stats:rx_packetes = 82
Jan 1 00:08:42.491:rtsp_reinitialize_scb:
Jan 1 00:08:42.503:rtsp_control_process_msg:
Jan 1 00:08:42.503:rtsp_control_process_msg:received MSG request of TYPE 0
Jan 1 00:08:42.503:rtsp_set_event:
Jan 1 00:08:42.503:rtsp_set_event:api_req_msg_type=RTSP_API_REQ_DESTROY
Jan 1 00:08:42.503:rtsp_session_cleanup:
Jan 1 00:08:42.503:rtsp_create_session_history:scb=0x62128F08, callID=0x7
Jan 1 00:08:42.503:rtsp_insert_session_history_record:current=0x6214BDC8, callID=0x7
Jan 1 00:08:42.503:rtsp_insert_session_history_record:count = 3
Jan 1 00:08:42.503:rtsp_insert_session_history_record:starting history record
deletion_timer of10 minutes
Jan 1 00:08:42.503:rtsp_session_cleanup:deleting session:scb=0x62128F08
Router#

```

**Related Commands**

Command	Description
<b>debug rtsp all</b>	Displays debugging output for the RTSP client API.
<b>debug rtsp pmh</b>	Displays debugging messages for the PMH.
<b>debug rtsp socket</b>	Displays debugging output for the RTSP client socket data.

# debug rtsp error

To display error information about the Real-Time Streaming Protocol (RTSP) client, use the **debug rtsp error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtsp error**

**no debug rtsp error**

**Syntax Description** This command has no arguments or keywords.

**Defaults** Debug is not enabled.

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.1(3)T	This command was introduced.
	12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.

**Usage Guidelines** We recommend that you log output from the **debug rtsp error** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Related Commands	Command	Description
	<b>debug rtsp api</b>	Displays debugging output for the RTSP client API.
	<b>debug rtsp pmh</b>	Displays debugging messages for the PMH.
	<b>debug rtsp socket</b>	Displays debugging output for the RTSP client socket data.
	<b>voice call debug</b>	Allows configuration of the voice call debugging output.

# debug rtsp pmh

To display debugging information about the Protocol Message Handler (PMH), use the **debug rtsp pmh** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtsp pmh**

**no debug rtsp pmh**

**Syntax Description** This command has no arguments or keywords.

**Defaults** Debug is not enabled.

**Command Modes** Privileged EXEC

## Command History

Release	Modification
12.1(3)T	This command was introduced.
12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.

## Usage Guidelines

We recommend that you log output from the **debug rtsp pmh** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

## Related Commands

Command	Description
<b>debug rtsp api</b>	Displays debugging output for the RTSP client API.
<b>debug rtsp error</b>	Displays error message for RTSP data.
<b>debug rtsp socket</b>	Displays debugging output for the RTSP client socket data.
<b>voice call debug</b>	Allows configuration of the voice call debugging output.

# debug rtsp session

To display client information and stream information for the stream that is currently active for the Real Time Streaming Protocol (RTSP) client, use the **debug rtsp session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtsp session**

**no debug rtsp session**

**Syntax Description** This command has no arguments or keywords.

**Defaults** Debug is not enabled.

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.1(3)T	This command was introduced.
	12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.
	12.3(4)T	This command replaces the <b>debug rtsp client</b> command and the <b>debug rtsp client session</b> command.

**Usage Guidelines** We recommend that you log output from the **debug rtsp session** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

**Examples** The following example shows the display of the debugging messages of the RTSP session:

```
Router# debug rtsp session

RTSP client session debugging is on
Router#
Router#!call initiated
Router#
*Mar 11 03:09:58.123: //-1//RTSP:/rtsp_get_new_scb:
*Mar 11 03:09:58.123: //-1//RTSP:/rtsp_initialize_scb:
*Mar 11 03:09:58.123: //-1//RTSP:/rtsplib_init_svr_session: 0x63A5FE6C
*Mar 11 03:09:58.123: //-1//RTSP:/rtsp_get_new_scb:
*Mar 11 03:09:58.123: //-1//RTSP:/rtsp_initialize_scb:
*Mar 11 03:09:58.123: //-1//RTSP:/rtsplib_init_svr_session: 0x63A5D874
Router#
Router#!call answered
```

```

Router#
Router#!digits dialed
Router#
Router#!call terminated
Router#
*Mar 11 03:10:38.139: //-1//RTSP:RS41:/rtsp_control_process_msg:
*Mar 11 03:10:38.139: //158//RTSP:/rtsp_control_process_msg: received MSG request of TYPE
0
*Mar 11 03:10:38.139: //158//RTSP:/rtsp_set_event: api_req_msg_type=RTSP_API_REQ_DESTROY
*Mar 11 03:10:38.139: //158//RTSP:/rtsp_session_cleanup:
*Mar 11 03:10:38.139: //-1//RTSP:/rtsplib_free_svr_session:
*Mar 11 03:10:38.139: //-1//RTSP:/rtsplib_stop_timer: timer(0x638D5DDC) stops
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_create_session_history: scb=0x63A5FE6C,
callID=0x9E
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_create_session_history: No streams in session
control block
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_session_cleanup: deleting session: scb=0x63A5FE6C
*Mar 11 03:10:38.143: //-1//RTSP:RS42:/rtsp_control_process_msg:
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_control_process_msg: received MSG request of TYPE
0
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_set_event: api_req_msg_type=RTSP_API_REQ_DESTROY
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_session_cleanup:
*Mar 11 03:10:38.143: //-1//RTSP:/rtsplib_free_svr_session:
*Mar 11 03:10:38.143: //-1//RTSP:/rtsplib_stop_timer: timer(0x63A60110) stops
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_create_session_history: scb=0x63A5D874,
callID=0x9E
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_create_session_history: No streams in session
control block
*Mar 11 03:10:38.143: //158//RTSP:/rtsp_session_cleanup: deleting session: scb=0x63A5D874

```

Table 265 describes the significant fields shown in the display.

**Table 265 debug rtsp session Field Descriptions**

Field	Description
//-1/	Indicates that the CallEntry ID for the module is unavailable.
//158/	Identifies the CallEntry ID.
RTSP:	Identifies the RTSP module.
rtsp_ function name	Identifies the function name.

**Related Commands**

Command	Description
<b>debug rtsp api</b>	Displays debugging output for the RTSP client API.
<b>debug rtsp error</b>	Displays error message for RTSP data.
<b>debug rtsp pmh</b>	Displays debugging messages for the PMH.
<b>debug rtsp socket</b>	Displays debugging output for the RTSP client socket data.
<b>voice call debug</b>	Allows configuration of the voice call debugging output.

# debug rtsp socket

To display debugging messages about the packets received or sent on the TCP or User Datagram Protocol (UDP) sockets, use the **debug rtsp socket** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug rtsp socket**

**no debug rtsp socket**

## Syntax Description

This command has no arguments or keywords.

## Defaults

Debug is not enabled.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.1(3)T	This command was introduced.
12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.

## Usage Guidelines

Each Real-Time Streaming Protocol (RTSP) session has a TCP port for control and a UDP (RTP) port for delivery of data. The control connection (TCP socket) is used to exchange a set of messages (request from the RTSP client and the response from the server) for displaying a prompt. The **debug rtsp socket** command enables the user to debug the message exchanges being done on the TCP control connection.



### Note

We recommend that you log output from the **debug rtsp socket** command to a buffer rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

## Related Commands

Command	Description
<b>debug rtsp api</b>	Displays debugging output for the RTSP client API.
<b>debug rtsp error</b>	Displays error message for RTSP data.
<b>debug rtsp pmh</b>	Displays debugging messages for the PMH.
<b>voice call debug</b>	Allows configuration of the voice call debugging output.

# debug rudpv1

For debug information for Reliable User Datagram Protocol (RUDP), use the **debug rudpv1** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug rudpv1 {application | performance | retransmit | segment | signal | state | timer |
transfer}
```

```
no debug rudpv1 {application | performance | retransmit | segment | signal | state | timer |
transfer}
```

## Syntax Description

<b>application</b>	Application debugging.
<b>performance</b>	Performance debugging.
<b>retransmit</b>	Retransmit/soft reset debugging.
<b>segment</b>	Segment debugging.
<b>signal</b>	Signals sent to applications.
<b>state</b>	State transitions.
<b>timer</b>	Timer debugging.
<b>transfer</b>	Transfer state information.

## Defaults

Debugging for rudpv1 is not enabled.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.1(1)T	This command was introduced.
12.2(4)T	This command was implemented on the Cisco 2600 series, Cisco 3600 series, and Cisco MC3810.
12.2(2)XB	This command was implemented on the Cisco AS5350 and Cisco AS5400 universal gateways.
12.2(2)XB1	This command was implemented on the Cisco AS5850 platform.
12.2(8)T	This command was implemented on Cisco IAD2420 series integrated access devices (IADs).
12.2(11)T	This command was implemented on the Cisco AS5350, Cisco AS5400, and Cisco AS5850 platforms.

## Usage Guidelines

Use this command only during times of low traffic.

**Examples**

The following is sample output from the **debug rudpv1 application** command:

```
Router# debug rudpv1 application

Rudpvl:Turning application debugging on
*Jan 1 00:20:38.271:Send to appl (61F72B6C), seq 12
*Jan 1 00:20:48.271:Send to appl (61F72B6C), seq 13
*Jan 1 00:20:58.271:Send to appl (61F72B6C), seq 14
*Jan 1 00:21:08.271:Send to appl (61F72B6C), seq 15
*Jan 1 00:21:18.271:Send to appl (61F72B6C), seq 16
*Jan 1 00:21:28.271:Send to appl (61F72B6C), seq 17
*Jan 1 00:21:38.271:Send to appl (61F72B6C), seq 18
*Jan 1 00:21:48.275:Send to appl (61F72B6C), seq 19
*Jan 1 00:21:58.275:Send to appl (61F72B6C), seq 20
*Jan 1 00:22:08.275:Send to appl (61F72B6C), seq 21
*Jan 1 00:22:18.275:Send to appl (61F72B6C), seq 22
*Jan 1 00:22:28.275:Send to appl (61F72B6C), seq 23
*Jan 1 00:22:38.275:Send to appl (61F72B6C), seq 24
*Jan 1 00:22:48.279:Send to appl (61F72B6C), seq 25
*Jan 1 00:22:58.279:Send to appl (61F72B6C), seq 26
*Jan 1 00:23:08.279:Send to appl (61F72B6C), seq 27
*Jan 1 00:23:18.279:Send to appl (61F72B6C), seq 28
*Jan 1 00:23:28.279:Send to appl (61F72B6C), seq 29
```

The following is sample output from the **debug rudpv1 performance** command:

```
Router# debug rudpv1 performance

Rudpvl:Turning performance debugging on
corsair-f#
*Jan 1 00:44:27.299:
*Jan 1 00:44:27.299:Rudpvl Sent:Pkts 11, Data Bytes 236, Data Pkts 9
*Jan 1 00:44:27.299:Rudpvl Rcvd:Pkts 10, Data Bytes 237, Data Pkts 9
*Jan 1 00:44:27.299:Rudpvl Discarded:0, Retransmitted 0
*Jan 1 00:44:27.299:
*Jan 1 00:44:37.299:
*Jan 1 00:44:37.299:Rudpvl Sent:Pkts 11, Data Bytes 236, Data Pkts 9
*Jan 1 00:44:37.299:Rudpvl Rcvd:Pkts 10, Data Bytes 237, Data Pkts 9
*Jan 1 00:44:37.299:Rudpvl Discarded:0, Retransmitted 0
*Jan 1 00:44:37.299:
*Jan 1 00:44:47.299:
*Jan 1 00:44:47.299:Rudpvl Sent:Pkts 11, Data Bytes 236, Data Pkts 9
*Jan 1 00:44:47.299:Rudpvl Rcvd:Pkts 11, Data Bytes 236, Data Pkts 9
*Jan 1 00:44:47.299:Rudpvl Discarded:0, Retransmitted 0
*Jan 1 00:44:47.299:
```

The following is sample output from the **debug rudpv1 retransmit** command:

```
Router# debug rudpv1 retransmit

Rudpvl:Turning retransmit/softreset debugging on
*Jan 1 00:52:59.799:Retrans timer, set to ack 199
*Jan 1 00:52:59.903:Retrans timer, set to ack 200
*Jan 1 00:53:00.003:Retrans timer, set to ack 201
*Jan 1 00:53:00.103:Retrans timer, set to ack 202
*Jan 1 00:53:00.203:Retrans timer, set to ack 203
*Jan 1 00:53:00.419:Retrans timer, set to ack 97
*Jan 1 00:53:00.503:Retrans handler fired, 203
*Jan 1 00:53:00.503:Retrans:203:205:
*Jan 1 00:53:00.503:
*Jan 1 00:53:00.607:Retrans timer, set to ack 207
*Jan 1 00:53:00.907:Retrans timer, set to ack 210
*Jan 1 00:53:01.207:Retrans handler fired, 210
*Jan 1 00:53:01.207:Retrans:210:211:212:
```

```

*Jan 1 00:53:01.207:
*Jan 1 00:53:01.207:Retrans timer, set to ack 213
*Jan 1 00:53:01.311:Retrans timer, set to ack 214
*Jan 1 00:53:01.419:Retrans timer, set to ack 98
*Jan 1 00:53:01.611:Retrans timer, set to ack 215
*Jan 1 00:53:01.711:Retrans timer, set to ack 218
*Jan 1 00:53:01.811:Retrans timer, set to ack 219
*Jan 1 00:53:01.911:Retrans timer, set to ack 220
*Jan 1 00:53:02.011:Retrans timer, set to ack 221
*Jan 1 00:53:02.311:Retrans handler fired, 221
*Jan 1 00:53:02.311:Retrans:221:
*Jan 1 00:53:02.311:
*Jan 1 00:53:02.311:Retrans timer, set to ack 222
*Jan 1 00:53:02.415:Retrans timer, set to ack 225

```

The following is sample output from the **debug rudpv1 segment** command:

```
Router# debug rudpv1 segment
```

```

Rudpv1:Turning segment debugging on
*Jan 1 00:41:36.359:Rudpv1: (61F72DAC) Rcvd ACK 61..198 (32)
*Jan 1 00:41:36.359:Rudpv1: (61F72DAC) Send ACK 199..61 (32)
*Jan 1 00:41:36.459:Rudpv1: (61F72DAC) Rcvd ACK 62..199 (8)
*Jan 1 00:41:36.459:Rudpv1: (61F72DAC) Rcvd ACK 62..199 (32)
*Jan 1 00:41:36.459:Rudpv1: (61F72DAC) Send ACK 200..62 (32)
*Jan 1 00:41:36.559:Rudpv1: (61F72DAC) Rcvd ACK 63..200 (32)
*Jan 1 00:41:36.559:Rudpv1: (61F72DAC) Send ACK 201..63 (32)
*Jan 1 00:41:36.659:Rudpv1: (61F72DAC) Rcvd ACK 64..201 (32)
*Jan 1 00:41:36.659:Rudpv1: (61F72DAC) Send ACK 202..64 (32)
*Jan 1 00:41:36.759:Rudpv1: (61F72DAC) Rcvd ACK 65..202 (32)
*Jan 1 00:41:36.759:Rudpv1: (61F72DAC) Send ACK 203..65 (32)
*Jan 1 00:41:36.859:Rudpv1: (61F72DAC) Rcvd ACK 66..202 (32)
*Jan 1 00:41:36.859:Rudpv1: (61F72DAC) Send ACK 204..66 (32)
*Jan 1 00:41:36.959:Rudpv1: (61F72DAC) Rcvd ACK 67..202 (32)
*Jan 1 00:41:36.959:Rudpv1: (61F72DAC) Rcvd ACK EAK 68..202 (9)
*Jan 1 00:41:36.959:Rudpv1: (61F72DAC) Send ACK 203..67 (32)
*Jan 1 00:41:36.963:Rudpv1: (61F72DAC) Send ACK 205..67 (32)
*Jan 1 00:41:36.963:Rudpv1: (61F72DAC) Rcvd ACK 68..204 (8)
*Jan 1 00:41:37.051:Rudpv1: (61F72B6C) Send ACK NUL 118..96 (8)
*Jan 1 00:41:37.051:Rudpv1: (61F72B6C) Rcvd ACK 97..118 (8)
*Jan 1 00:41:37.059:Rudpv1: (61F72DAC) Rcvd ACK 68..205 (32)
*Jan 1 00:41:37.063:Rudpv1: (61F72DAC) Send ACK 206..68 (32)
*Jan 1 00:41:37.263:Rudpv1: (61F72DAC) Rcvd ACK 70..206 (32)
*Jan 1 00:41:37.363:Rudpv1: (61F72DAC) Send ACK EAK 207..68 (9)
*Jan 1 00:41:37.363:Rudpv1: (61F72DAC) Rcvd ACK 71..206 (32)
*Jan 1 00:41:37.363:Rudpv1: (61F72DAC) Rcvd ACK 69..206 (32)
*Jan 1 00:41:37.363:Rudpv1: (61F72DAC) Send ACK 207..71 (8)
*Jan 1 00:41:37.363:Rudpv1: (61F72DAC) Send ACK 207..71 (32)
*Jan 1 00:41:37.363:Rudpv1: (61F72DAC) Send ACK 208..71 (32)
*Jan 1 00:41:37.363:Rudpv1: (61F72DAC) Send ACK 209..71 (32)
*Jan 1 00:41:37.367:Rudpv1: (61F72DAC) Rcvd ACK 72..209 (8)
*Jan 1 00:41:37.463:Rudpv1: (61F72DAC) Rcvd ACK 72..209 (32)
*Jan 1 00:41:37.463:Rudpv1: (61F72DAC) Send ACK 210..72 (32)
*Jan 1 00:41:37.563:Rudpv1: (61F72DAC) Rcvd ACK 73..210 (32)
*Jan 1 00:41:37.563:Rudpv1: (61F72DAC) Send ACK 211..73 (32)

```

The following is sample output from the **debug rudpv1 signal** command:

```
Router# debug rudpv1 signal
```

```
Rudpv1:Turning signal debugging on
*Jan 1 00:39:59.551:Rudpv1:Sent CONN_FAILED to connID 61F72DAC, sess 33
*Jan 1 00:39:59.551:
*Jan 1 00:39:59.551:Rudpv1:Sent CONN_TRANS_STATE to connID 61F72B6C, sess 34
*Jan 1 00:39:59.551:
*Jan 1 00:39:59.551:Rudpv1:Sent CONN_TRANS_STATE to connID 61F72DAC, sess 33
*Jan 1 00:39:59.551:
*Jan 1 00:39:59.551:Rudpv1:Sent CONN_OPEN to connID 61F72B6C, sess 34

*Jan 1 00:39:59.551:Rudpv1:Sent AUTO_RESET to connID 61F72DAC, sess 33
*Jan 1 00:39:59.551:
*Jan 1 00:40:00.739:%LINK-5-CHANGED:Interface FastEthernet0, changed state
to administratively down
*Jan 1 00:40:01.739:%LINEPROTO-5-UPDOWN:Line protocol on Interface
FastEthernet0, changed state to down
*Jan 1 00:40:04.551:Rudpv1:Sent CONN_RESET to connID 61F72DAC, sess 33
*Jan 1 00:40:04.551:
*Jan 1 00:40:05.051:Rudpv1:Clearing conn rec values, index 2, connid
61F72DAC
*Jan 1 00:40:10.051:Rudpv1:Sent CONN_RESET to connID 61F72DAC, sess 33
*Jan 1 00:40:10.051:
*Jan 1 00:40:10.551:Rudpv1:Clearing conn rec values, index 2, connid
61F72DAC
*Jan 1 00:40:15.551:Rudpv1:Sent CONN_RESET to connID 61F72DAC, sess 33
*Jan 1 00:40:15.551:
*Jan 1 00:40:16.051:Rudpv1:Clearing conn rec values, index 2, connid
61F72DAC

*Jan 1 00:40:21.051:Rudpv1:Sent CONN_RESET to connID 61F72DAC, sess 33
*Jan 1 00:40:21.051:
*Jan 1 00:40:21.551:Rudpv1:Clearing conn rec values, index 2, connid
61F72DAC
*Jan 1 00:40:25.587:%LINK-3-UPDOWN:Interface FastEthernet0, changed state
to up
*Jan 1 00:40:26.551:Rudpv1:Sent CONN_RESET to connID 61F72DAC, sess 33
*Jan 1 00:40:26.551:
*Jan 1 00:40:26.587:%LINEPROTO-5-UPDOWN:Line protocol on Interface
FastEthernet0, changed state to up
*Jan 1 00:40:27.051:Rudpv1:Clearing conn rec values, index 2, connid
61F72DAC
*Jan 1 00:40:28.051:Rudpv1:Sent CONN_OPEN to connID 61F72DAC, sess 33
```

The following is sample output from the **debug rudpv1 state** command:

```
Router# debug rudpv1 state
```

```
Rudpv1:Turning state debugging on

*Jan 1 00:38:37.323:Rudpv1: (61F72DAC) State Change:OPEN -> CONN_FAILURE
*Jan 1 00:38:37.323:Rudpv1: (61F72B6C) State Change:OPEN -> TRANS_STATE
*Jan 1 00:38:37.323:Rudpv1: (61F72DAC) State Change:CONN_FAILURE ->
TRANS_STATE
*Jan 1 00:38:37.323:Rudpv1: (61F72B6C) State Change:TRANS_STATE -> OPEN
*Jan 1 00:38:37.323:Rudpv1: (61F72DAC) State Change:TRANS_STATE -> SYN_SENT
*Jan 1 00:38:37.455:%LINK-5-CHANGED:Interface FastEthernet0, changed state
to administratively down
*Jan 1 00:38:38.451:%LINEPROTO-5-UPDOWN:Line protocol on Interface
FastEthernet0, changed state to down
*Jan 1 00:38:42.323:Rudpv1: (61F72DAC) State Change:SYN_SENT -> CLOSED
*Jan 1 00:38:42.823:Rudpv1: (61F72DAC) State Change:INACTIVE -> SYN_SENT
```

```

*Jan 1 00:38:47.823:Rudpv1: (61F72DAC) State Change:SYN_SENT -> CLOSED
*Jan 1 00:38:48.323:Rudpv1: (61F72DAC) State Change:INACTIVE -> SYN_SENT
*Jan 1 00:38:53.323:Rudpv1: (61F72DAC) State Change:SYN_SENT -> CLOSED
*Jan 1 00:38:53.823:Rudpv1: (61F72DAC) State Change:INACTIVE -> SYN_SENT
*Jan 1 00:38:56.411:%LINK-3-UPDOWN:Interface FastEthernet0, changed state
to up
*Jan 1 00:38:57.411:%LINEPROTO-5-UPDOWN:Line protocol on Interface
FastEthernet0, changed state to up
*Jan 1 00:38:57.823:Rudpv1: (61F72DAC) State Change:SYN_SENT -> OPEN

```

The following is sample output from the **debug rudpv1 timer** command:

```
Router# debug rudpv1 timer
```

```

Rudpv1:Turning timer debugging on
*Jan 1 00:53:40.647:Starting Retrans timer for connP = 61F72B6C, delay = 300
*Jan 1 00:53:40.647:Stopping SentList timer for connP = 61F72B6C
*Jan 1 00:53:40.747:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:40.747:Stopping Retrans timer for connP = 61F72B6C
*Jan 1 00:53:40.747:Starting Retrans timer for connP = 61F72B6C, delay = 300
*Jan 1 00:53:40.747:Stopping SentList timer for connP = 61F72B6C
*Jan 1 00:53:40.847:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:40.847:Stopping Retrans timer for connP = 61F72B6C
*Jan 1 00:53:40.847:Starting Retrans timer for connP = 61F72B6C, delay = 300
*Jan 1 00:53:40.847:Stopping SentList timer for connP = 61F72B6C
*Jan 1 00:53:40.947:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:40.947:Stopping Retrans timer for connP = 61F72B6C
*Jan 1 00:53:40.947:Starting Retrans timer for connP = 61F72B6C, delay = 300
*Jan 1 00:53:40.947:Stopping SentList timer for connP = 61F72B6C
*Jan 1 00:53:41.047:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:41.147:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:41.151:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:41.151:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:41.151:Stopping Retrans timer for connP = 61F72B6C
*Jan 1 00:53:41.151:Starting SentList timer for connP = 61F72B6C, delay = 300
*Jan 1 00:53:41.419:Timer Keepalive (NullSeg) triggered for conn = 61F72DAC
*Jan 1 00:53:41.419:Starting Retrans timer for connP = 61F72DAC, delay = 300
*Jan 1 00:53:41.419:Stopping SentList timer for connP = 61F72DAC
*Jan 1 00:53:41.419:Starting NullSeg timer for connP = 61F72DAC, delay = 1000
*Jan 1 00:53:41.419:Stopping Retrans timer for connP = 61F72DAC
*Jan 1 00:53:41.451:Timer SentList triggered for conn = 61F72B6C
*Jan 1 00:53:41.451:Starting SentList timer for connP = 61F72B6C, delay = 300
*Jan 1 00:53:41.451:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:41.451:Stopping SentList timer for connP = 61F72B6C
*Jan 1 00:53:41.551:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:41.551:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:41.551:Starting NullSeg timer for connP = 61F72B6C, delay = 1000
*Jan 1 00:53:41.551:Starting NullSeg timer for connP = 61F72B6C, delay = 1000

```

The following is sample output from the **debug rudpv1 transfer** command:

Router# **debug rudpv1 transfer**

```
Rudpv1:Turning transfer debugging on
*Jan 1 00:37:30.567:Rudpv1:Send TCS, connId 61F72B6C, old connId 61F72DAC
*Jan 1 00:37:30.567:Rudpv1:Initiate transfer state, old conn 61F72DAC to
new conn 61F72B6C
*Jan 1 00:37:30.567:Rudpv1:Old conn send window 51 .. 52
*Jan 1 00:37:30.567:Rudpv1:New conn send window 255 .. 2
*Jan 1 00:37:30.567:Rudpv1:Rcvd TCS 142, next seq 142
*Jan 1 00:37:30.567:Rudpv1:Rcv'ing trans state, old conn 61F72DAC to new
conn 61F72B6C
*Jan 1 00:37:30.567:Rudpv1:Seq adjust factor 148
*Jan 1 00:37:30.567:Rudpv1:New rcvCur 142
*Jan 1 00:37:30.567:Rudpv1:Send transfer state, old conn 61F72DAC to new
conn 61F72B6C
*Jan 1 00:37:30.567:Rudpv1:Send TCS, connId 61F72B6C, old connId 61F72DAC,
seq adjust 208, indication 0
*Jan 1 00:37:30.567:Rudpv1:Transfer seg 51 to seg 3 on new conn
*Jan 1 00:37:30.567:Rudpv1:Finishing transfer state, old conn 61F72DAC to
new conn 61F72B6C
*Jan 1 00:37:30.567:Rudpv1:Send window 2 .. 4
```

#### Related Commands

Command	Description
<b>clear rudpv1 statistics</b>	Clears RUDP statistics and failure counters.
<b>show rudpv1</b>	Displays RUDP failures, parameters, and statistics.